Panasonic

CONTROL FPWIN PRO

Befehlssatz

für alle SPS-Typen

Panasonic

Panasonic Electric Works Europe AG, im Folgenden kurz PEW genannt, weist darauf hin, dass Informationen und Hinweise in diesem Handbuch technischen Änderungen unterliegen können, da die Produkte von PEW ständig weiterentwickelt werden. PEW übernimmt keine Haftung für die in diesem Handbuch enthaltenen Druckfehler oder sonstige Ungenauigkeiten, es sei denn, dass PEW die Fehler oder Ungenauigkeiten nachweislich bekannt sind oder diese PEW aufgrund grober Fahrlässigkeit unbekannt sind und PEW von einer Behebung der Fehler oder Ungenauigkeiten aus diesen Gründen abgesehen hat. PEW weist den Anwender ausdrücklich darauf hin, dass dieses Handbuch nur eine allgemeine Beschreibung technischer Vorgänge und Hinweise enthält, deren Umsetzung nicht in jedem Einzelfall in der vorliegenden Form sinnvoll sein kann. In Zweifelsfällen ist daher unbedingt mit PEW Rücksprache zu nehmen.

Verbesserungsvorschläge zu diesem Handbuch werden gerne entgegengenommen unter:
techdoc.peweu@eu.panasonic.com

Panasonic Electric Works Europe AG
Rudolf-Diesel-Ring 2
D-83607 Holzkirchen

Wichtige Symbole

Die folgenden Symbole werden in diesem Handbuch verwendet:

GEFAHR!

Unter dem Warndreieck werden im Handbuch besondere Sicherheitshinweise und Warnungen gegeben. Bei deren Nichteinhaltung können je nach speziellem Fall Personenschäden und/oder erhebliche Sachschäden auftreten.

VORSICHT

Verfahren Sie mit Vorsicht! Bei Nichtbeachtung besteht Verletzungsgefahr oder die Gefahr von Geräteschäden bzw. Datenverlust.

HINWEIS

Enthält wichtige zusätzliche Informationen.

Beispiel

Enthält ein Beispiel zur Veranschaulichung des vorhergehenden Textabschnitts.

Vorgehensweise

Kennzeichnet eine Schritt-für-Schrittanleitung.

REFERENZ

Weist auf eine zusätzliche Informationsquelle hin.

Inhaltsverzeichnis

Teil I Grundlagen

Teil III FP-Befehle

15. Arithmetikbefehle 335

16. Bitweise Boolsche Befehle 495

17. Bit-Schiebefunktionen 537

Kapitel 1

Grundlagen

1.1 Operanden

In FPWIN Pro werden als Operanden bezeichnet:

- Ein- und Ausgänge sowie interne Speicher
- Merker
- Sondermerker
- Zeitgeber und Zähler
- Datenregister
- Sonderdatenregister
- File-Register
- Koppeldatenregister und -merker

Die Anzahl der zur Verfügung stehenden Operanden hängt vom SPS-Typ und ihrem Ausbau ab. Wieviele von den jeweiligen Operanden zur Verfügung stehen, entnehmen Sie daher bitte Ihrer Hardware-Beschreibung.

1.1.1 Ein-/Ausgänge (X/Y)

Je nach SPS- und Modultyp stehen Ihnen unterschiedlich viele Ein- bzw. Ausgänge zur Verfügung. Jeder Eingangskontakt entspricht einem Eingang **X** und jeder Ausgangskontakt einem Ausgang **Y**.

In Systemregister 20 stellen Sie ein, ob ein Ausgang im kompletten Programm einmal oder mehrmals belegt werden darf.

☞ **Ausgänge, die physikalisch nicht vorhanden sind, können wie Merker verwendet werden. Diese Merker sind nicht selbsthaltend, das heißt ihre Inhalte gehen z.B. bei einem Stromausfall verloren.**

1.1.2 Merker (R)

Merker sind Speicherbereiche, in die Sie z.B. Zwischenergebnisse schreiben können. Programmtechnisch werden Merker wie interne Ausgänge behandelt.

Im Systemregister 7 stellen Sie ein, ob ein Merker selbsthaltend ist oder nicht. Das heißt, ob er seine Werte nach einem Spannungsausfall hält oder nicht.

Wie viele Merker Ihnen zur Verfügung stehen, hängt vom SPS-Typ ab. Entnehmen Sie die Anzahl daher bitte Ihrer Hardware-Beschreibung.

1.1.3 Sondermerker

Sondermerker sind Speicher für festgelegte Systemfunktionen. Sie werden von der SPS automatisch gesetzt bzw. zurückgesetzt und dienen:

- zur Anzeige bestimmter Systemzustände, z.B. Fehler
- als Impulsgeber
- zur Initialisierung oder werden zum Beispiel
- als EIN/AUS-Kontrollkontakt in einer Bedingung verwendet.

Einige Merker haben z.B. einen bestimmten Status, wenn in einem SPS-Netz Daten für eine Übertragung anliegen.

Wie viele Merker Ihnen zur Verfügung stehen, hängt vom SPS-Typ ab. Entnehmen Sie die Anzahl daher bitte Ihrer Hardware-Beschreibung.

☞ Sondermerker können nur gelesen werden.

1.1.4 Zeitgeber und Zähler

Zeitgeber (Timer) haben zusammen mit den Zählern (Counter) einen gemeinsamen Speicher- und Adressbereich.

Definieren Sie in den Systemregistern 5 und 6, wie der Speicherbereich zwischen Zeitgeber und Zähler aufgeteilt werden soll, und welche Zeitgeber/Zähler selbsthaltend oder nicht selbsthaltend sein sollen. Selbsthaltend bedeutet, dass die Daten auch nach einem Spannungsausfall nicht verloren gehen. Dies ist bei nicht selbsthaltenden Registern nicht der Fall.

Die Eingabe einer Zahl im Systemregister 5 definiert den ersten Zähler. Alle Zahlen, die kleiner als die eingegebene Zahl sind, definieren Zeitgeber.

Zum Beispiel: Wenn Sie Null eingeben, definieren Sie ausschließlich Zähler. Wenn Sie den höchstmöglichen Wert eingeben, definieren Sie nur Zeitgeber.

In der Standardeinstellung wird der selbsthaltende Bereich durch die Startadresse des Zählerbereichs definiert. Das heißt: Alle Zähler sind selbsthaltend und alle Zeitgeber nicht selbsthaltend. Sie können diese Einstellung jedoch anpassen und einen anderen Wert für den selbsthaltenden Bereich setzen. Dadurch lassen sich einige Zeitgeber (bei Bedarf auch alle) als selbsthaltend definieren.

Neben dem Speicherbereich für Zeitgeber/Zähler, gibt es einen Speicherbereich, der für den Sollwert (set value - SV) und den Istwert (elapsed value - EV) jedes einzelnen Zeitgeber/Zählerkontakts reserviert ist. Die Größe beider Bereiche beträgt 16 Bits (WORD). Im SV- und EV-Bereich lässt sich ein Ganzzahlwert von 0 bis 32.767 speichern.

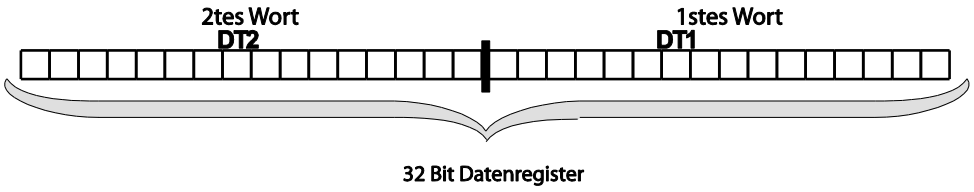
Zeitgeber/Zähler-Nr.	SV	EV	Merker
TM0	SV0	EV0	T0
. . .	. . .	. . .	. . .
TM99	SV99	EV99	T99
CT100	SV100	EV100	C100
. . .	. . .	. . .	. . .

Während ein Zeitgeber oder Zähler verarbeitet wird, lässt sich der zugehörige Wert lesen und unter bestimmten Umständen auch bearbeiten.

☞ **Wenn Sie die Einstellungen im Systemregister 5 geändert haben, vergessen Sie nicht, die Adressen der Zeitgeber/Zähler im SPS-Programm anzupassen, denn diese entsprechen den TM/CT-Nummern.**

1.1.5 Datenregister (DT)

Datenregister haben eine Breite von 16 Bits. Sie lassen sich beispielsweise dazu verwenden, Konstanten/Parameter zu schreiben und zu lesen. Wenn ein Befehl 32 Bit erfordert, werden zwei 16-Bit-Datenregister verwendet. Ist dies der Fall, geben Sie die Adresse des ersten Datenregisters mit dem Präfix DDT anstelle von DT ein. Das nächste Datenregister (Word) wird automatisch verwendet.



Datenregister können selbsthaltend oder nicht selbsthaltend sein. Selbsthaltend bedeutet, dass die Daten auch nach einem Spannungsausfall nicht verloren gehen. Stellen Sie die selbsthaltenden/nicht selbsthaltenden

Bereiche im Systemregister 8 ein, indem Sie die Startadresse des selbsthaltenden Bereichs festlegen.

Die Anzahl der verfügbaren Datenregister hängt vom SPS-Typ ab (siehe Hardware-Beschreibung):

1.1.6 Sonderdatenregister (DT)

Sonderdatenregister werden wie Sondermerker für bestimmte Aufgaben von der SPS belegt und größtenteils von der SPS gesetzt und zurückgesetzt.

Die Registerbreite beträgt 16 Bit (Datentyp = WORD). Die Anzahl der verfügbaren Datenregister hängt vom SPS-Typ ab (siehe Hardware-Beschreibung):

Sonderdatenregister können Sie im allgemeinen nur lesen. Ausnahmen sind:

- Interrupts und Zykluszeit (DT9027, DT9023-DT9024; FP0-T32CP: DT90027, DT90023-DT90024)...
- Istwerte des schnellen Zählers (DT9044 und DT9045 bzw. DT90044 und DT90045 für FP0)
- Kontrollmerker des schnellen Zählers DT9052 (DT90053 für FP0-T32CP)
- Echtzeituhr (DT90054 bis DT90058; für FP0-T32CP: DT90044 und DT90045)
- Echtzeituhr (DT90054 bis DT90058; FP0-T32CP: DT90054 bis DT90058)

1.1.7 File-Register (FL)

Bei bestimmten SPS-Typen (siehe Hardware-Beschreibung) stehen Ihnen zusätzlich Datenregister zur Auswahl, mit denen Sie den Datenregisterbereich erweitern können. File-Register werden genau wie Datenregister verwendet. Im Systemregister 9 können Sie den selbsthaltenden/nicht selbsthaltenden Bereich festlegen. Selbsthaltend bedeutet, dass die Daten auch nach einem Spannungsausfall nicht verloren gehen.

1.1.8 Koppelmerker und -register (L/LD)

Koppelmerker haben eine Breite von 1 Bit (BOOL). In den Systemregistern 10-13 und 40-55, definieren Sie:

- den Sendebereich
- die Anzahl der zu sendenden Koppelmerkerworte
- den selbsthaltenden und nicht selbsthaltenden Bereich

Koppeldatenregister haben eine Breite von 16 Bit (WORD). In den Systemregistern 10-13 und 40-55, definieren Sie:

- den Sendebereich
- die Anzahl der zu sendenden Koppelmerkerworte
- den selbsthaltenden und nicht selbsthaltenden Bereich

1.2 Adressen

Für jede Variable, die Sie in FPWIN Pro in der globalen Variablenliste definieren, wird im Feld "Adresse" die physikalische Adresse der Variablen eingetragen.

Zur Adresse gehören der Operand und die Adressnummer. In FPWIN Pro können Sie entweder FP-Adressen und/oder IEC-Adressen verwenden. Für diese beiden Adresstypen stehen folgende Operandenkürzel zur Auswahl:

Operand	FP-Adresse	IEC-Adresse
Eingang	X	I
Ausgang	Y	Q
Speicher (interner Speicherbereich)	R	M0
Zeitgeberkontakt	T	M1
Zählerkontakt	C	M2
Sollwert	SV	M3
Istwert	EV	M4
Datenregister	DT/DDT	M5
Koppelmerker	L	M6
Koppeldatenregister	LD	M7
File-Register	FL	M8

Bitte entnehmen Sie die verfügbaren Registernummern (z.B. DT9000/DT90000) Ihrer Hardware-Beschreibung. In den folgenden beiden Abschnitten werden die Adresstypen und ihre Zusammensetzung detailliert beschrieben.

Unter einer Adresse versteht man die direkte Hardware-Adresse eines Ein-/Ausgangs bzw. eines Registers oder Zählers.

Wenn Sie zum Beispiel die Hardware-Adresse des 1. Eingangs und des 4. Ausgangs einer SPS angeben möchten:

- X0 (X = Eingang, 0 = erster Merker)
- Y3 (Y = Ausgang, 3 = vierter Merker)

Für die Speicherbereiche verwenden Sie die folgenden Adressenkürzel. Die Registernummern finden Sie in der Hardware-Beschreibung.

Speicherbereich	Abk.	Beispiel
Speicher (interner Speicherbereich)	R	R9000: Selbstdiagnosefehler
Zeitgeberkontakt	T	T200: Zeitgeber Nr. 200 (Einstellungen im Systemregister 5 und 6)
Zählerkontakt	C	C100: Zähler Nr. 100 (Einstellungen im Systemregister 5 und 6)
Sollwert	SV	SV200 (Sollwert für Zählermerker 200)
Istwert	EV	EV100 (Istwert für Zeitgeber 100)
Datenregister	DT	DT9001/DT90001 (Signale für Spannungseinbruch)
Koppelmerker	L	L1270
Koppeldatenregister	LD	LD255
File-Register	FL	FL8188

1.2.1 IEC-Adressen

Die Zusammensetzung einer IEC-1131-Adresse hängt ab von:

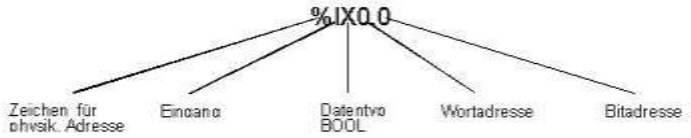
- Operandentyp
- Datentyp
- Steckplatz der Einheit (Word-Adresse)
- Merkernr. (Bit-Adresse)
- SPS-Typ

Die Ein- und Ausgänge sind die wichtigsten Komponenten einer Speicherprogrammierbaren Steuerung (SPS). Die SPS erhält von den Eingängen Signale und verarbeitet diese im SPS-Programm. Die Ergebnisse werden entweder gespeichert oder zum Ausgang übertragen, d.h. die SPS steuert die Ausgänge.

Eine SPS hat auch Sonderspeicherbereiche, Kürzel "M", um Zwischenergebnisse zu speichern. Zum Beispiel:

Wenn Sie den Status von Eingang 1 des ersten Moduls lesen und Ausgang 4 des zweiten Moduls steuern möchten, benötigen Sie die physikalische Adresse jedes Ein- bzw. Ausgangs. Physikalische FPWIN-Pro-Adressen bestehen aus einem Prozentzeichen, einer Abkürzung der Ein-/Ausgänge, einer Abkürzung des Datentyps und der Wort- und Bitadressen:

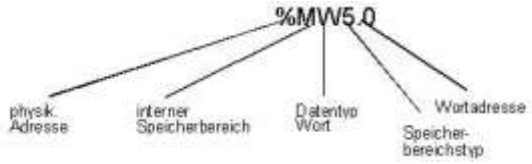
Beispiel: IEC-Adresse für einen Eingang



Das Prozentzeichen ist ein Indikator der physikalischen Adresse. “I” bedeutet Eingang, “X” bezeichnet den Datentyp BOOL. Die erste Null steht für die Wortadresse (Steckplatznr.) und die zweite Null für die Bit-Adresse. Beachten Sie, dass die Zählung mit Null beginnt und sich die Zählung von Wort- und Bit-Adressen je nach SPS-Typ unterscheidet.

Eine SPS hat interne Speicherbereiche (M), um Zwischenergebnisse zu speichern. Zum Beispiel: Wenn Sie einen internen Speicherbereich nutzen, z.B. Datenregister, beachten Sie die zusätzliche Zahl (hier 5) für den Speichertyp:

Beispiel: IEC-Adresse für internen Speicherbereich



Die Bit-Adressen müssen bei Datenregistern, Zählern, Zeitgebern oder Soll- und Istwerten nicht angegeben werden.

Gemäß IEC 1131 lauten die Abkürzungen für **Ein- und Ausgänge** “I” und “Q”. Für die **Speicherbereiche** sind folgende Abkürzungen vorgesehen:

Speichertyp	Nr.	Beispiel
Interne Merker (R)	0	%MX0.900.0 = interner Merker R9000
Zeitgeber (T)	1	%MX1.200 = Zählernr. 200
Zähler (C)	2	%MX2.100 = Zählernr. 100
Sollwert Zähler/Zeitgeber (SV)	3	%MW3.200 für Sollwert von Zähler Nr. 200
Istwert Zähler/Zeitgeber (EV)	4	%MW4.100 für Istwert von Zähler Nr. 100
Datenregister (DT, DDT)	5	%MW5.9001 = Datenregister DT9001 %MD5.90001 = 32-Bit Datenregister DDT90001
Koppelmerker (WL)	7	%MW7.63 = Koppelmerker 63

Speichertyp	Nr.	Beispiel
Koppeldatenregister (LD)	8	%MW8.127 = Koppeldatenregister 127
File-Register (FL)	9	%MW9.800 = File-Register 800

☞ **Tabellen mit Speicherbereichs-Adressen finden Sie in der Hardware-Beschreibung der SPS.**

Sie können folgende Datentypen verwenden:

Datentyp	Abkürzung	Wertebereich	Datenbreite
BOOL	BOOL	0 (FALSE), 1 (TRUE)	1 Bit
INTEGER	INT	-32.768 bis 32.768	16 Bit
DOUBLE INTEGER	DINT	-2.147.438.648 bis 2.147.438.647	32 Bit
WORD	WORD	0 bis 65.535	16 Bit
DOUBLE WORD	DWORD	0 bis 4.294.987.295	32 Bit
TIME (16 Bits):	TIME	T#0,00s bis T#327,67s	16 Bit*
TIME (32 Bit):	TIME	T#0,00s bis T#21 474 836,47s	32 Bit*
STRING	STRING	1 bis 255 Byte (ASCII)	8 Bit pro Byte
TIME (32 Bit):	TIME	T#0,00s bis T#21 474 836,47s	32 Bit
REAL	REAL	$-1.175494 \times 10^{-38}$ bis $-3.402823 \times 10^{-38}$ und 1.175494×10^{-38} bis 3.402823×10^{-38}	32 Bit

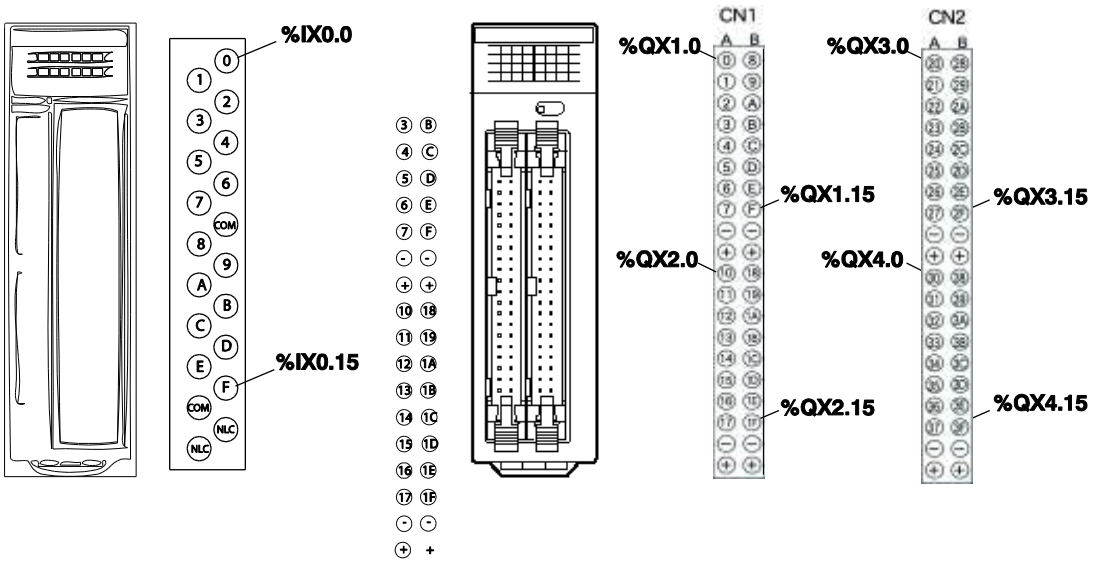
*abhängig von der SPS

☞ **Bitte beachten Sie, dass nicht alle Datentypen für jeden IEC-Befehl verwendet werden können.**

Die Wortadressen werden je nach Steuerungstyp unterschiedlich bestimmt (siehe jeweilige Hardware-Beschreibung). Für FP0/FP1/FP-M werden Adressen **NICHT durchgezählt**. Bei den Ausgängen fangen sie sowohl für die Wort- als auch für die Bitadressen wieder bei Null an. Angenommen, Sie haben eine FP1 C24 mit 16 Eingängen und 8 Ausgängen, dann lauten die Eingangsadressen: Eingänge: %IX0.0 - %IX0.15, Ausgänge: %QX0.0 - %QX0.7.

Adressen werden **fortlaufend** numeriert. Wenn Sie z.B. ein Eingangsmodul mit 16 Eingängen in den ersten Steckplatz und ein Ausgangsmodul mit 32 Ausgängen in den zweiten Steckplatz stecken, belegt das Eingangsmodul die Adressen: %IX0.0 - %IX0.15 und das Ausgangsmodul die Adressen: %QX1.0 - %QX2.15. Die physikalische Adresse setzt sich aus Modultyp (I/Q), der Steckplatznummer (Wortadresse) und der Nummer des Merkers (Bitadresse) zusammen.

Input module **Output module**



Daraus ist zu erkennen, wie die hexadezimale Zählung (0-F für 0-15) konvertiert wird. Die Adressenzuweisung finden Sie in Ihrer Hardware-Beschreibung.

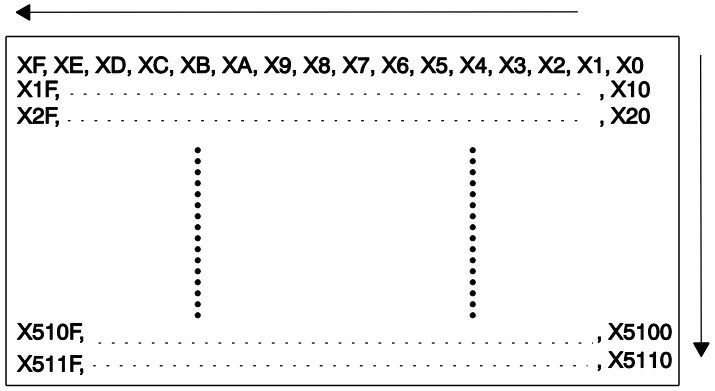
◆HINWEIS

- **Weitere Tabellen mit Speicherbereichen finden Sie ebenfalls in der jeweiligen Hardware-Beschreibung.**
- **Bei Zeitgebern, Zählern, Soll-/Istwerten und Datenregistern muss die Bit-Adresse nicht angezeigt werden.**
- **Sie können auch die Registernummer (R9000, DT9001/90001) oder die FP-Adresse, z.B. "X0" (Eingang 0), anstelle der IEC-Adresse eingeben.**

1.2.2 Spezifizierung von Relais-Adressen

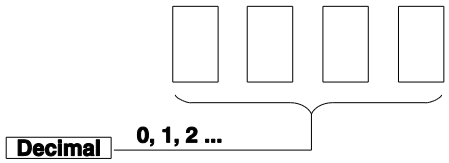
Zu den Relais zählen externe Eingangsrelais (X), externe Ausgangsrelais (Y), Merker (R), Koppelmerker (L) und Pulsrelais (P). Die niedrigste dieser Relaisadressen wird in Hexadezimalzahlen ausgedrückt und die zweiten und höheren Ziffern in Dezimalzahlen (siehe nachstehendes Beispiel).

Beispiel **Konfiguration des externen Eingangsrelais (X)**



1.2.3 Zeitgeber-Kontakte (T) und Zähler-Kontakte (C)

Adressen von Zeitgeber-Kontakten (T) und Zähler-Kontakten (C) entsprechen den **TM**- und **CT**-Befehlsnummern. Die Adressen sind vom jeweiligen SPS-Typ abhängig.



e.g. for FP2:
T0, T1 ..... T2999
C3000, C3001 ..... C3072

Da Adressen für Zeitgeber-Kontakte (T) und Zähler-Kontakte (C) den TM- und CT-Befehlsnummern entsprechen, ist folgendes zu beachten: Falls der von TM und CT gemeinsam benutzte Befehl durch das Systemregister 5 geändert wird, ändert sich auch der Zeitgeber- und Zähler-Kontakt.

1.2.4 Fehleralarmmerker

◆HINWEIS

- **Nur die SPS-Typen FP2SH und FP10SH verfügen über Fehleralarmmerker.**
- **Einschränkungen Fehleralarmmerker (siehe S. 34)**

Fehleralarmmerker (E) werden verwendet, um Fehler aufzuzeigen, die außerhalb der SPS auftreten. Aus diesem Grund wurde ein Sonderdatenregisterbereich definiert, der dem Benutzer Zugriff auf Fehlerinformationen wie die Anzahl der gesetzten Fehlermerker und eine Fehlerhistorie bietet.

Wenn ein Merker vom Fehleralarm-Programm auf TRUE gesetzt wird, weil eine Fehlersituation eingetreten ist, wird die Zahl der eingeschalteten Merker im Sonderdatenregister DT90400 gespeichert. Die Merkernummern werden der Reihe nach in den Sonderdatenregistern DT90401 bis DT90419 gespeichert. Wenn mindestens einer der Fehleralarmmerker E0 bis E2047 auf TRUE gesetzt wird, wird R9040 auf TRUE gesetzt. Der Zeitpunkt des ersten Fehleralarms wird in DT90420 bis DT90422 gespeichert.

Das Abbildung unten zeigt die interne Struktur und die Adresszuweisung im Sonderdatenregisterbereich dieses Fehlerspeichers.

GVL

Glob. Variablen

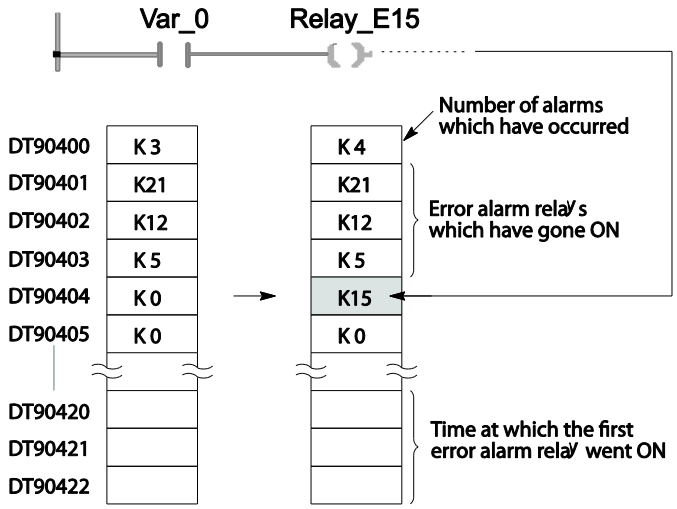
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	Relay_E15	E15	%MX10.15	BOOL	FALSE
1	VAR_GLOBAL					

POE-Kopf

Glob. Variablen | Program_1

	Klasse	Bezeichner	Typ	Initial
0	VAR	Var_0	BOOL	FALSE
1	VAR_EXTERNAL	Relay_E15	BOOL	FALSE

KOP



DT90400	Anzahl der aufgetretenen Alarme
DT90401–DT90419	Fehleralarmmerker (in chronologischer Reihenfolge)
DT90420	Sekunde und Minute

DT90421	Stunde und Tag
DT90422	Monat und Jahr
R9040	Wird auf TRUE gesetzt, sobald einer der Fehleralarmmerker E0 bis E2047 auf TRUE gesetzt wird.

Da in Control FPWIN Pro alle Schreiboperationen für Fehlermerker intern zu SET (s. S. 842)- und RST (s. S. 842)-Befehlen kompiliert werden, sind immer auch der Sondermerker R9040 und die Sonderdatenregister DT90400 bis DT90422 betroffen.

Wenn alle Fehleralarmmerker auf FALSE gesetzt werden, wird auch R9040 auf FALSE gesetzt.

Wenn Sie Alarmmerker in Control FPWIN Pro überwachen möchten: **Monitor → Sondermerker und -datenregister → Alarmmerker**.

1.2.4.1 Einschränkungen Fehleralarmmerker

Ein Fehleralarmmerker kann unbegrenzt oft in einem Programm verwendet werden. Wird jedoch ein Fehleralarmmerker für verschiedene Fehlerzustände in mehreren Fehleralarmprogrammen verwendet, ist eine genaue Identifizierung des Fehlers nicht möglich. Die CPU kontrolliert eine mehrfache Verwendung nicht.

Beim Abschalten der SPS oder beim Umschalten von PROG- in RUN-Modus werden die Fehleralarmmerker und die zugehörigen Sonderdatenregister gehalten. Zum Zurücksetzen der Fehleralarmmerker und der Sonderdatenregister müssen Sie im PROG-Modus den Initialisierungsschalter betätigen.

Systemregister 4 kann aber so eingestellt werden, dass die Daten auch dann nicht gelöscht werden. Dann wird erst der nächste Übertragung des Programms auf die SPS die Fehlermerker und die entsprechenden Sonderdatenregister zurücksetzen.

1.2.5 Pulsmerker (P)

Ein Pulsmerker (P) schaltet für nur einen Scan ein. Der EIN/AUS-Status wird nicht extern ausgegeben, sondern nur im Programm ausgeführt.

Ein Pulsmerker schaltet sich nur bei einem Befehl für eine steigende oder fallende Flanke ein.

Wenn ein Pulsmerker als Trigger verwendet wird, funktioniert er nur während eines Scans bei steigender oder fallender Flanke.

Beispiel: Global deklariert

GVL:

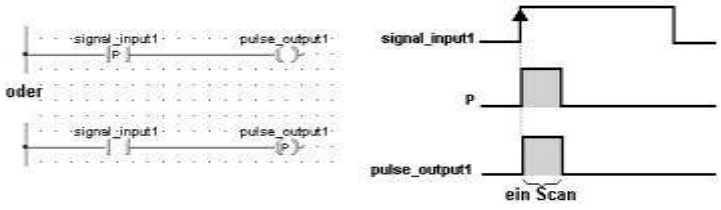
Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	pulse_output1	R0	%MX0.0.0	BOOL	FALSE

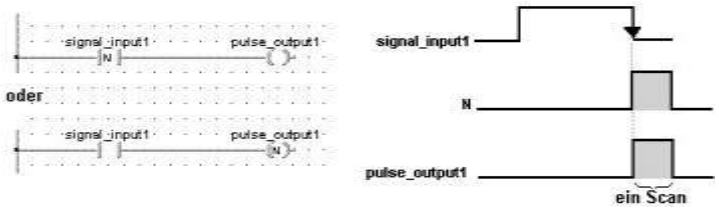
POE-Kopf

	Klasse	Bezeichner	Typ	Initial
0	VAR	signal_input1	BOOL	FALSE
1	VAR_EXTERNAL	pulse_output1	BOOL	FALSE

Ausführung mit steigender Flanke:



Ausführung mit fallender Flanke:



1.2.5.1 Einschränkungen Pulsrelais (P)

Ein Pulsmerker kann nur einmal im Programm als Ausgang verwendet werden, d.h. eine mehrfache Verwendung ist nicht erlaubt.

Ein Pulsmerker kann unbegrenzt als Kontakt verwendet werden.

Ein Pulsmerker kann nicht als Ausgang für die Befehle **KP**, **SET**, **RST** oder **ALT** verwendet werden.

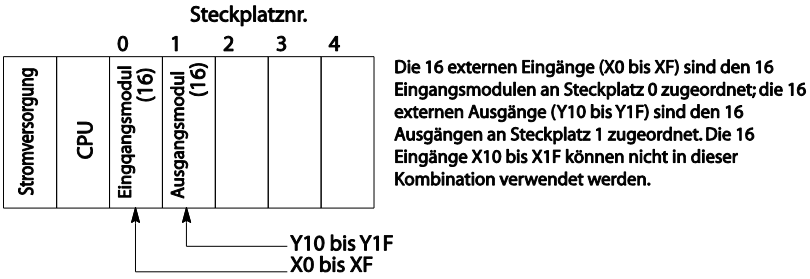
Ein WP-Merker lässt sich nicht als Speicherort für einen hochrangigen Befehl verwenden.

In Control FPWIN PRO können Pulsmerker nur in den oben beschriebenen Situationen oder zusammen mit einer DF- oder DFN-Anweisung benutzt werden. Es gibt zwar die Möglichkeit die Anzahl der Pulsmerker zu erhöhen, aber in Control FPWIN Pro meist keine Notwendigkeit dazu.

1.2.6 Externe Ein- (X) und Ausgangsrelais (Y)

- Die externen Eingänge sind tatsächlich als Eingänge zugeordnet.
- Die externen Ausgänge können dazu verwendet werden, externe Geräte ein- und auszuschalten. Die anderen externen Ausgänge können genauso wie die internen Ausgänge verwendet werden.
- Die E/A Zuweisung hängt von der Kombination der E/A- und Analogmodule ab. Für FP10SH und FP2SH können 8192 Ein-/Ausgänge verwendet werden, Ausgang und Eingang eingeschlossen. Für FP2 und FP3 können 2048 E/A verwendet werden.

Beispiel



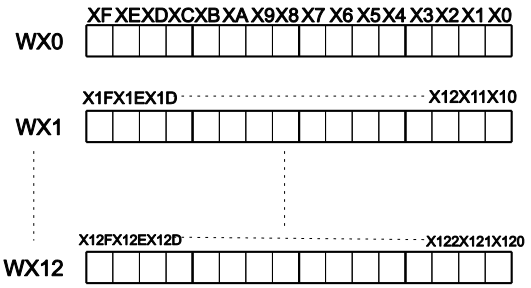
Die 16 externen Eingänge (X0 bis XF) sind den 16 Eingangsmodulen an Steckplatz 0 zugeordnet; die 16 externen Ausgänge (Y10 bis Y1F) sind den 16 Ausgängen an Steckplatz 1 zugeordnet. Die 16 Eingänge X10 bis X1F können nicht in dieser Kombination verwendet werden.

1.2.7 Darstellung der Merker in Worten (WX, WY, WR, und WL)

Der externe Eingang (X), der externe Ausgang (Y), der interne Merker (R) und das Koppelregister (L) können auch im Wort-Format dargestellt werden. Das Wort-Format behandelt Gruppen von 16-Bit Merker als ein Wort. Diese Merker können in Worten dargestellt werden als externer Eingang (WX), externer Ausgang (WY), Wortmerker (WR) und Koppelregister (WL).

◆Beispiel

Konfiguration des externen Eingangs in Worten (WX)



◆HINWEIS

Weil der Inhalt des Wortmerkers dem Status des Merkers bzw. Komponenten entspricht, falls einige Merker auf AN geschaltet werden, verändert sich der Inhalt der Worte.

1.3 Konstanten

Eine Konstante beinhaltet einen festen Wert. Je nach Anwendung lässt sich eine Konstante als Summand, Faktor, Adresse, Ein-/Ausgangsnummer, Sollwert etc. verwenden.

Es gibt drei Arten von Konstanten:

- Dezimalkonstanten
- Hexadezimalkonstanten
- BCD-Konstanten

1.3.1 Dezimalkonstanten

Dezimalkonstanten können eine Breite von 16 oder 32 Bit haben.

Wertebereich 16 Bit -32.768 bis 32.768

Wertebereich 32 Bit -2.147.483.648 bis 2.147.483.648

Konstanten werden intern in das 16-Bit-Binärformat mit Vorzeichen konvertiert und als solche verarbeitet. Geben Sie einfach die Dezimalzahl in das Programm ein.

1.3.2 Hexadezimalkonstante

Hexadezimalkonstanten beanspruchen weniger Stellen als Binärdaten. 16-Bit-Konstanten können als 4-stellige und 32 Bit breite Konstanten oder 8-stellige Hexadezimalkonstanten dargestellt werden.

Wertebereich 16 Bit: 8000 bis 7FFF

Wertebereich 32 Bit: 80000000 bis 7FFFFFFF

Beispiel: Im Programm geben Sie 16#7FFF für den Hexadezimalwert 7FFF ein.

1.3.3 BCD-Konstanten

BCD ist die Abkürzung für Binary Coded Decimal (binär codierte Dezimalzahl).

Wertebereich 16 Bit: 0 bis 9999

Wertebereich 32 Bit: 0 bis 99999999

Im Programm geben Sie BCD-Konstanten wie folgt ein:

binär: 2#0001110011100101 oder
hexadezimal: 16#9999

1.4 Datentypen

In Control FPWIN Pro muss jeder Variable ein bestimmter Datentyp zugeordnet werden. Wir unterscheiden zwischen Datentypen nach IEC 61131-3 (s. S. 38) und benutzerdefinierten Datentypen.

1.4.1 Elementare Datentypen

Schlüsselwort	Datentyp	Bereich	Reservierter Speicher	Initialwert
BOOL	Boolescher Wert	0 (FALSE) 1 (TRUE)	1 Bit	0
WORD	Bitzeichenfolge der Länge 16	0–65535	16 Bits	0
DWORD	Bitzeichenfolge der Länge 32	0–4294967295	32 Bits	0
INT	Ganze Zahl	-32768–32,767	16 Bits	0
DINT	Doppelwort für ganze Zahlen	-2147483648– 2147483647	32 Bits	0
UINT	Vorzeichenlose ganze Zahl	0–65,535	16 Bits	0
UDINT	Vorzeichenloses Doppelwort für ganze Zahlen	0–4294967295	32 Bits	0
REAL	Reelle Zahl	-3.402823466*E38– -1.175494351*E-38 0.0 +1.175494351*E-38– +3.402823466*E38	32 Bits	0.0
TIME	Zeitdauer	T#0s–T#327.67s	16 Bits [1)]	T#0s
		T#0s–T#21474836.47s	32 Bits [1)]	
DATE_AND_TIME	Datum und Uhrzeit	DT#2001-01-01-00:00:00– DT#2099-12-31-23:59:59	32 Bits	DT#2001-01-01-00:00:00
DATE	Datum	D#2001-01-01–D#2099-12-31	32 Bits	D#2001-01-01
TIME_OF_DAY	Uhrzeit	TOD#00:00:00–TOD#23:59:59	32 Bits	TOD#00:00:00
STRING	Zeichenfolge variabler Länge	1–32767 Bytes (ASCII), je nach Speichergröße der SPS	2 Worte für den Kopf + (n+1)/2 Worte für die Zeichen	''

[1)] Abhängig vom SPS-Typ

1.4.1.1 BOOL

Variablen vom Datentyp BOOL sind binäre Variablen. Sie können nur den Wert 0 oder 1 haben und haben immer eine Breite von 1 Bit.

Der Zustand 0 entspricht **FALSE** (z.B. Initialwert im POE-Kopf) und heißt, die Schaltvariable ist ausgeschaltet. Man spricht in diesem Fall auch davon, dass die Variable nicht gesetzt ist.

Der Zustand 1 entspricht **TRUE** (z.B. Initialwert im POE-Kopf) und heißt, die Schaltvariable ist eingeschaltet. Man spricht in diesem Fall auch davon, dass die Variable gesetzt ist.

Die Voreinstellung für den Initialwert, z.B. bei der Variablendeklaration im POE-Kopf bzw. in der Liste der globalen Variablen ist 0 (FALSE). Die Schaltvariable ist in diesem Fall beim SPS-Programmstart nicht gesetzt. Sollte dies nicht der Fall sein, können Sie den Initialwert auch auf TRUE einstellen.

1.4.1.2 INT

Variablenwerte vom Datentyp INTEGER sind natürliche Zahlen ohne Kommastellen. Für INTEGER-Werte beträgt der Wertebereich -32768 bis 32767.

Für eine Variable von diesem Datentyp ist der Initialwert 0 voreingestellt.

Zahlen können im Dezimal-, Hexadezimal- oder Binärformat eingegeben werden.

Dezimalzahl	Hexadezimalzahl	Binärzahl
1234	16#4D2	2#10011010010
-1234	16#FB2E	2#1111101100101110

1.4.1.3 UINT

Die Variablenwerte vom Datentyp vorzeichenloser INTEGER sind ganze natürliche Zahlen ohne Kommastellen. Für UINT-Werte beträgt der Wertebereich 0–65535.

1.4.1.4 DINT

Variablenwerte vom Datentyp DOUBLE INTEGER sind natürliche Zahlen ohne Kommastellen. Der Wertebereich für den Datentyp DOUBLE INTEGER umfasst -2147483648 bis 2147483647.

Für eine Variable von diesem Datentyp ist der Initialwert 0 voreingestellt.

Zahlen können im Dezimal-, Hexadezimal- oder Binärformat eingegeben werden.

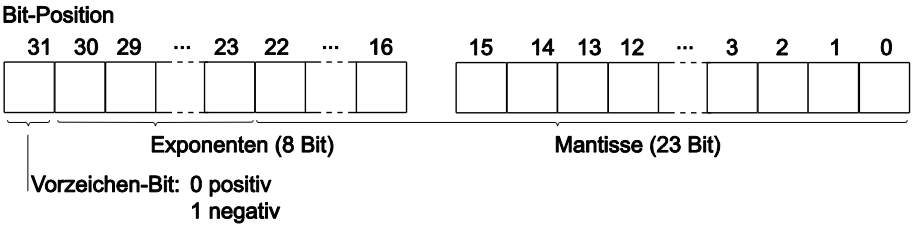
Dezimalzahl	Hexadezimalzahl	Binärzahl
123456789	16#75BCD15	2#11101011011110011010010101
-123456789	16#F8A432EB	2#11111000101001000011001011110

1.4.1.5 UDINT

Die Variablenwerte vom Datentyp vorzeichenlose DOUBLE INTEGER sind ganze natürliche Zahlen ohne Kommastellen. Für UDINT-Werte beträgt der Wertebereich 0–4294967295.

1.4.1.6 REAL

Die Variablen des Datentyps REAL sind 32-Bit-Zahlen, die auf der Norm IEEE754 basieren. Die Mantisse hat 23 Bit und der Exponent 8 Bit.



Der Wertebereich für REAL-Werte liegt zwischen -3,402823466*E38 bis -1,175494351*E-38, 0.0, +1,175494351*E-38 bis +3,402823466*E38.

Die Voreinstellung für den Initialwert, z.B. bei der Variablendeklaration im POE-Kopf bzw. in der Liste der globalen Variablen = 0,0.**nur für FP-e und FP0:** Verwenden Sie keine REAL-Befehle in Interrupt-Programmen.

Sie können REAL-Werte in folgendem Format eingeben:

[+-] Integer.Integer [(Ee) [+-] Integer]

Beispiele:

5.983e-7

-33.876e12

3.876e3

0.000123

123.0

☞ **Der REAL-Wert muss immer mit einem Punkt (z.B. 123.0) eingegeben werden.**

1.4.1.7 WORD

Eine Variable vom Datentyp WORD besteht aus 16 binären Zuständen. Die Schaltzustände von 16 Ein-/Ausgängen können als eine Einheit in einem Wort zusammengefasst werden.

Für eine Variable von diesem Datentyp ist der Initialwert 0 voreingestellt.

Zahlen können im Dezimal-, Hexadezimal- oder Binärformat eingegeben werden.

Dezimalzahl	Hexadezimalzahl	Binärzahl
1234	16#4D2	2#10011010010
64302	16#FB2E	2#1111101100101110

1.4.1.8 DWORD

Eine Variable vom Datentyp DOUBLE WORD besteht aus 32 binären Zuständen. In einem Doppelwort können die Schaltzustände von 32 Ein-/Ausgängen als Einheit zusammengefasst werden.

Für eine Variable von diesem Datentyp ist der Initialwert 0 voreingestellt.

Zahlen können im Dezimal-, Hexadezimal- oder Binärformat eingegeben werden.

Dezimalzahl	Hexadezimalzahl	Binärzahl
123456789	16#75BCD15	2#11101011011110011010010101
4171510507	16#F8A432EB	2#11111000101001000011001011110

1.4.1.9 TIME

TIME (16 Bits): FP1, FP-M, FP3, FP-C, FP5, FP10, FP10S

Bei Variablen vom Datentyp TIME (16 Bit) können Sie eine Dauer von 0,01 bis 327,67 Sekunden angeben. Die Auflösung beträgt 10ms.

TIME (32 Bits): FP-X, FP-Sigma, FP0, FP2/2SH, FP10SH

Bei Variablen vom Datentyp TIME (32 Bit) können Sie eine Dauer von 0,01 bis 21.474.836,47 Sekunden angeben. Die Auflösung beträgt 10ms.

Voreinstellung für 16- und 32-Bit-Werte = T#0 (entspricht 0 Sekunden)

◆HINWEIS

- Werte vom Datentyp TIME müssen mit dem Präfix T# oder TIME# beginnen.
- Die Einheiten der Zeitliterale können mit dem Zeichen "_" unterteilt werden.
- Die Einheiten, z.B. Sekunden, Millisekunden etc. können mit Groß- oder Kleinbuchstaben geschrieben werden.
- Das Überschreiten des Bereichs der größten Einheit wird unterstützt, z.B. wird T#25h_15m akzeptiert.

Beschreibung	Beispiele

Zeitliterale ohne Trennstriche:
mit kurzem Präfix
	`T#14ms  T#-14ms  T#14,7s  T#14,7m`
	`T#14,7h  T#14,7d  t#25h15m`
	`t#5d14h12m18s3,5ms`

mit langem Präfix	`TIME#14s  TIME#-14s  time#14,7s`
Zeitliterale mit Trennstrichen: mit kurzem Präfix	`T#25h_15m` `T#5d_14h_12m_18s_3,5ms`
mit langem Präfix	`TIME#25h_15m` `time#5d_14h_12m_18s_3,5ms`

1.4.1.10 DATE_AND_TIME

Die Variablenwerte vom Datentyp DATE_AND_TIME sind Datums- und Zeitliterale. Für DATE_AND_TIME-Werte beträgt der Wertebereich DT#2001-01-01-00:00:00– DT#2099-12-31-23:59:59.

Beschreibung	Beispiele
Mit kurzem Präfix	DT#2010-06-07-15:36:55 dt#2010-06-07-15:36:55
Mit langem Präfix	DATE_AND_TIME#2010-06-07-15:36:55 date_and_time#2010-06-07-15:36:55
Interne Darstellung	Sekunden nach DT#2001-01-01-00:00:00

Vorteile:

- Erleichtert jede Art von Berechnungen von Datum und Uhrzeit
- Geeignet für nachführbare Photovoltaikanwendungen
  - Sonnenstand, Sonnenaufgang, Sonnenuntergang
  - Konvertierungen zwischen Universalzeit und Ortszeit
- Gebäudeautomatisierung

- Feiertage (z.B. Ostern), Sommerzeit
- Verwendbar u.a. zum Stellen (SET_RTC_DT (s. S. 288)) oder Lesen (GET_RTC_DT (s. S. 283)) der Uhr-/Kalenderfunktion der SPS
- erleichtert die Anpassung und Integration von POEs, die nach IEC 61131-3 mit Programmiersoftware anderer Anbieter, wie z.B. OSCAT (Open Source Community for Automation Technology), erstellt wurden

1.4.1.11 DATE

Die Variablenwerte vom Datentyp DATE sind Datumsliterale. Für DATE-Werte beträgt der Wertebereich D#2001-01-01–D#2099-12-31.

Beschreibung	Beispiele
Mit kurzem Präfix	D#2010-06-07 d#2010-06-07
Mit langem Präfix	DATE#2010-06-07 date#2010-06-07
Interne Darstellung	Sekunden nach 2001-01-01

Vorteile:

- Erleichtert jede Art von Berechnungen von Datum und Uhrzeit
- Geeignet für nachführbare Photovoltaikanwendungen
  - Sonnenstand, Sonnenaufgang, Sonnenuntergang
  - Konvertierungen zwischen Universalzeit und Ortszeit
- Gebäudeautomatisierung
- Feiertage (z.B. Ostern), Sommerzeit
- Verwendbar u.a. zum Stellen (SET_RTC_DT (s. S. 288)) oder Lesen (GET_RTC_DT (s. S. 283)) der Uhr-/Kalenderfunktion der SPS

1.4.1.12 TIME_OF_DAY

Die Variablenwerte vom Datentyp TIME_OF_DAY sind Uhrzeitliterale. Für TIME_OF_DAY-Werte beträgt der Wertebereich TOD#00:00:00–TOD#23:59:59.

Beschreibung	Beispiele
Mit kurzem Präfix	TOD#15:36:55 tod#15:36:55
Mit langem Präfix	TIME_OF_DAY#15:36:55 time_of_day#15:36:55
Interne Darstellung	Sekunden nach TOD#00:00:00

Vorteile:

- Erleichtert jede Art von Berechnungen von Datum und Uhrzeit
- Geeignet für nachführbare Photovoltaikanwendungen
  - Sonnenstand, Sonnenaufgang, Sonnenuntergang
  - Konvertierungen zwischen Universalzeit und Ortszeit
- Gebäudeautomatisierung
- Feiertage (z.B. Ostern), Sommerzeit
- Verwendbar u.a. zum Stellen (SET_RTC_DT (s. S. 288)) oder Lesen (GET_RTC_DT (s. S. 283)) der Uhr-/Kalenderfunktion der SPS

1.4.1.13 STRING

Der Datentyp STRING besteht aus einer Folge von bis zu 32767 ASCII-Zeichen. Die maximale Zeichenzahl ist abhängig von der Speichergröße der SPS. Die Voreinstellung können Sie unter **Extras** → **Optionen** → **Compiler-Optionen** → **Code-Erzeugung** ändern.

Die Voreinstellung für den Initialwert, z.B. bei der Variablendeklaration im POE-Kopf bzw. in der globalen Variablenliste, ist ein leeres Feld, das durch '' begrenzt wird.

Deklaration

Verwenden Sie für die Deklaration von STRING-Variablen im POE-Kopf die folgende Syntax:

STRING[n], mit **n** = Zeichenzahl

Die Standardzeichenzahl für STRING ist 32.

Interne Speicherstruktur von Zeichenfolgen auf der Steuerung

Jedes Zeichen der Zeichenfolge wird in einem Byte gespeichert. Der Speicherbereich für eine Zeichenfolge umfasst einen Kopf (2 Worte) und jeweils ein Wort für zwei Zeichen.

- Das erste Wort enthält die Anzahl an Zeichen, die für diese Zeichenfolge im Speicher reserviert werden.
- Das zweite Wort enthält die tatsächliche Zeichenzahl in der Zeichenfolge.
- Nachfolgende Worte enthalten die ASCII-Zeichen (zwei pro Wort)

Geben Sie die Zeichenfolgenlänge mit folgender Formel an, um einen Bereich im Speicher zu reservieren: Speichergröße = 2 Worte (Kopf) + (n+1)/2 Worte (Zeichen)

Der Speicher ist wortweise organisiert. Deshalb wird immer auf die nächst höhere Ganzzahl gerundet.

Wort x	Für die Zeichenfolge reservierte Zeichen	
Wort x+1	Tatsächlich in der Zeichenfolge gespeicherte Zeichen	
Wort x+2	Zeichen 2	Zeichen 1
Wort x+3	Zeichen 4	Zeichen 3
Wort x+4	Zeichen 6	Zeichen 5
	...	...
Wort x+(n+1)/2+1	Zeichen n	Zeichen n-1
	Höherwertiges Byte	Niederwertiges Byte

Ein Programmierbeispiel finden Sie unter F159_MTRN (s. S. 703).

Zeichenfolgenliterale (nach IEC 61131-3)

Ein Zeichenfolgenliteral ist eine Folge von null oder mehr Zeichen, die durch einfache Anführungszeichen (') begrenzt wird.

Dabei wird das Dollarzeichen ($), auf das zwei hexadezimalen Ziffern folgen, als die hexadezimale Darstellung des 8-Bit-Zeichencodes interpretiert.

Folgen aus dem Dollarzeichen und einem weiteren Zeichen werden gemäß folgender Tabelle interpretiert:

Kombination	Gedrucktes Zeichen
$$	Dollarzeichen ($24)
$'	Einfache Anführungszeichen ($27)
$L oder $l	Zeilenvorschub ($0A)
$N oder $n	Neue Zeile ($0D$0A)
$P oder $p	Formularvorschub ($0C)
$R oder $r	Wagenrücklauf ($0D)
$T oder $t	Tabulator ($09)

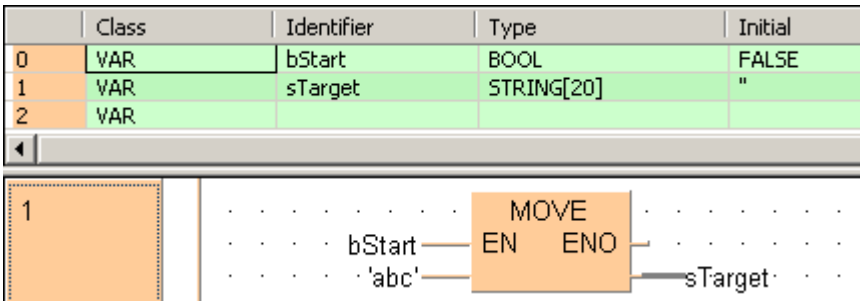
Beispiele für Zeichenfolgenliterale

Beispiel	Erklärung
''	Leere Zeichenfolge (Länge 0)
'A'	Zeichenfolge der Länge 1 mit dem Zeichen A
' '	Zeichenfolge der Länge 1 mit dem Leerzeichen
'$''	Zeichenfolge der Länge 1 mit einem einfachen Anführungszeichen
'RL'	Zeichenfolge der Länge 2 mit einem CR- und einem LF-Zeichen
'$$1.00'	Zeichenfolge der Länge 5 wird gedruckt als "$1.00"
'$02$03'	Zeichenfolge der Länge 2 mit einem STX- und einem ETX-Zeichen

Zeichenfolgen als Konstanten

Es ist möglich, Werte vom Datentyp STRING direkt als Konstanten in eine Funktion oder einen Funktionsbaustein einzugeben. Die Zeichenfolge muss von einfachen Anführungszeichen begrenzt sein.

Übergabe der Zeichenfolge 'abc' an die Zeichenfolgenvariable sTarget.



Übergabe von Zeichenfolgen an Funktionen oder Funktionsbausteine

Bei der Übergabe von Zeichenfolgen werden nur so viele Zeichen übergeben, wie maximal in die Zielzeichenfolge passen. Siehe hierzu die Beispiele in der Online-Hilfe unter dem Stichwort 'STRING':

1. Kopieren einer Ausgangszeichenfolge in eine kürzere Zielzeichenfolge
2. Kopieren einer Zeichenfolgen-Konstante in eine kürzere Zielzeichenfolgen-Konstante
3. Erzeugen einer Nachricht mit Hilfe einer Zeichenfolgenfunktion

◆HINWEIS

Die Umwandlungsfunktionen (E_)INT_TO_STRING (s. S. 217), (E_)DINT_TO_STRING (s. S. 220), (E_)REAL_TO_STRING (s. S. 225), (E_)TIME_TO_STRING (s. S. 227) usw. beanspruchen viele Systemressourcen (Programmschritte und Verarbeitungszeit). Wenn Sie diese Funktionen häufig verwenden, betten Sie sie in benutzerdefinierte Funktionen ein und verwenden Sie diese in Ihrem Projekt. Bei älteren Steuerungen (FP0, FP3, FP5, FP10) gilt dies auch für die Befehle CONCAT (s. S. 264) und FIND (s. S. 268).

Stringfunktionen mit EN/ENO

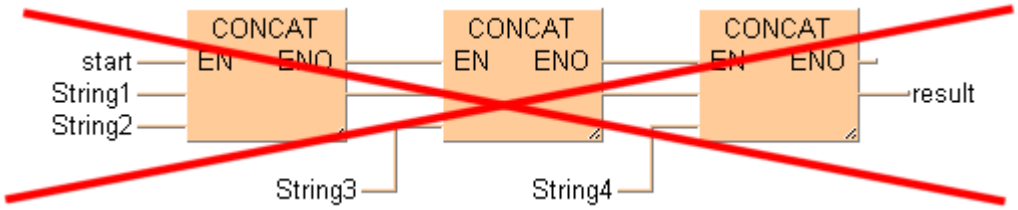
Verwendung von STRING-Funktionen mit Freigabe-Eingang (EN) und Freigabe-Ausgang (ENO) im Kontaktplan (KOP) und in der Funktionsbausteinsprache (FBS).

In KOP und in FBS können STRING-Befehle mit EN/ENO Kontakten nicht aneinander gehängt werden.

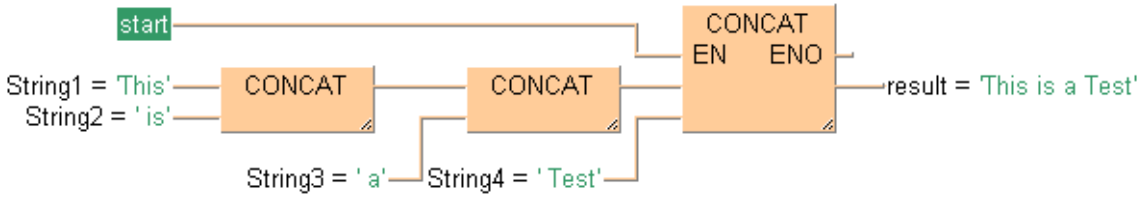
Sie können diese Konfiguration trotzdem nutzen, wenn zunächst die entsprechenden Funktionen ohne EN/ENO aneinander gehängt werden und abschließend die Funktion mit EN/ENO verwendet wird. Ihr EN-Freigabe-Eingang entscheidet dann über die Ausgabe des Gesamtergebnisses.

◆Beispiel

Diese Anordnung ist nicht möglich:



Diese Anordung ist möglich:



Verwendung von STRING-Funktionen in der Anweisungsliste (AWL)

STRING-Befehle mit EN/ENO können in der AWL aneinander gereiht werden. Um Zwischenvariablen zu vermeiden, empfiehlt es sich jedoch, einen bedingten Sprung statt einer Hintereinanderschaltung vieler Funktionen mit EN/ENO zu setzen.

◆Beispiel

POE-Kopf eines Programms mit Zwischenstring

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	String1	STRING[4]	'This'
2	VAR	String2	STRING[3]	' is'
3	VAR	String3	STRING[2]	' a'
4	VAR	String4	STRING[5]	' Test'
5	VAR	result	STRING[32]	''
6	VAR	help_string	STRING[32]	''

AWL-Rumpf

```
(* When start = TRUE then calculate
   result = String1 + String2 + String3 + String4 *)

LD          start
E_CONCAT    String1, String2, help_string
E_CONCAT    help_string, String3, help_string
E_CONCAT    help_string, String4, result
```

POE-Kopf eines Programms mit bedingtem Sprung

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	String1	STRING[4]	'This'
2	VAR	String2	STRING[3]	' is'
3	VAR	String3	STRING[2]	' a'
4	VAR	String4	STRING[5]	' Test'
5	VAR	result	STRING[32]	''

AWL-Rumpf

```
1       (* When start = TRUE then calculate
           result = String1 + String2 + String3 + String4 *)

        LDN         start
        JMPC        marker
        LD          String1
        CONTACT     String2
        CONTACT     String3
        CONTACT     String4
        ST          result
        ..........................................................
2
marker: (* Insert code for network 2 here *)
```

Bei der Programmierung mit Zwischenstring liegt die spezielle Schwierigkeit in der korrekten Wahl seiner Länge. Bei der Hintereinanderschaltung von unbedingten Stringfunktionen wird diese automatisch berechnet.

1.4.2 Generische Datentypen

Neben den elementaren Datentypen können generische Datentypen zur Definition der Ein- und Ausgänge von Standardfunktionen und Funktionsbausteinen verwendet werden. Generische Datentypen sind durch das Präfix ANY gekennzeichnet.

◆HINWEIS

In benutzerdefinierten POEs können keine generischen Datentypen verwendet werden.

ANY	
ANY_NUM	REAL, ANY_INT
ANY_INT	INT, DINT UINT, UDINT
ANY16	WORD INT, UINT
ANY32	DWORD DINT, UDINT DATE, TOD, DT
ANY_BIT	BOOL WORD, DWORD
ANY_DATE	DATE, TOD, DT

1.4.3 SDT

Ein SDT (strukturierter Datentyp) besteht aus mehreren, auch unterschiedlichen elementare Datentypen.

1.4.3.1 SDT-Typen

SDT mit nicht-überlapppenden Elementen

Im Dialogfeld "Eigenschaften" können Sie zwischen zwei Möglichkeiten der Speicherbelegung durch den SDT wählen:

1. mit überlappenden Elementen (s. S. 47)
2. mit nicht-überlapppenden Elementen

Speicherbelegung von SDTs mit nicht-überlappenden Elementen:

Alle Elemente vom Datentyp BOOL werden zu einem Block zusammengefasst und dem Bitspeicherbereich beginnend bei einer 16-Bit-Wortadresse der Reihe nach zugewiesen.

Alle Elemente vom Datentyp ARRAY OF BOOL werden zu einem Block zusammengefasst und dem Bitspeicherbereich beginnend bei einer 16-Bit-Wortadresse zugewiesen.

Alle anderen Elemente werden zu einem Block zusammengefasst und einem für 16-Bit-Worte reservierten Speicherbereich der Reihe nach zugewiesen.

SDTs mit überlappenden Elementen

Speicherbelegung von SDTs mit überlappenden Elementen:

Alle Elemente von gleichen Datentypen (BOOL, WORD, INT, DWORD, DINT, REAL und STRINGs mit einer gleichen, gemeinsamen Stringlänge) werden jeweils für sich zusammengefasst und hintereinander ab einer gemeinsamen Startadresse reserviert. Alle Arrays werden ebenfalls von dieser gemeinsamen Startadresse aus zugeordnet.

Für diese Startadresse gilt: Sind Elemente vom Typ BOOL oder ARRAY OF BOOL vorhanden, wird der SDT in den für Bits reservierten Speicher gelegt, ansonsten wird er in einem für 16-Bit-Worte reservierten Speicher gelegt.

Um Zweideutigkeiten bei der Initialisierung zu vermeiden, dürfen diese SDTs nicht initialisiert werden. Folgende Default-Initialisierungen werden verarbeitet:

- BOOL: FALSE
- WORD, INT, DWORD, DINT: 0
- REAL: 0.0
- STRING: '' (das heißt, die Adresse, die von der größtmöglichen Zeichenkette verwendet wird, wird initialisiert mit der größtmöglichen Länge derjenigen Zeichenkette, die größer oder gleich null ist. Der Rest der Zeichenkette wird mit Nullen initialisiert.)

Außerdem müssen alle Elementvariablen vom Typ String am Ende der Deklaration stehen.

◆HINWEIS

- Generell muss die jeweilige Speicherbelegung der verwendeten Datentypen genauestens beachtet werden.
- Insbesondere bei Verwendung von STRINGs ist deren spezielle Speicherbelegung zu berücksichtigen, die mit Hilfe der anderen Elemente beliebig überschrieben werden können.
- Außerdem müssen alle Elementvariablen vom Typ String am Ende der Deklaration stehen.

1.4.3.2 SDT verwenden

SDT anlegen

1. **Objekt → Neu → SDT oder** 
2. **STD-Name eingeben**

   Aktivieren Sie bei Bedarf das Kontrollfeld für SDTs mit überlappenden Elementen (s. S. 47).
3. **[OK]**
4. **Neuen SDT im Fenster "Projekt" öffnen**
5. **Variablen für den STD deklarieren**
6. **Objekt → Kontrolle oder** 

SDT in der globalen Variablenliste verwenden

1. **"Globale Variablen" in Fenster "Projekt" öffnen**
2. **Gegebenenfalls neue Zeile mit**  **oder**  **einfügen**
3. **Unter "Klasse" "VAR_GLOBAL" wählen**
4. **Unter "Bezeichner" symbolischen Name eingeben**
5. **FP- oder IEC-Adresse eingeben**

   Das erste Element des SDT bestimmt den Adresstyp: ist es vom Typ BOOL muss es eine Bit-Adresse sein (z.B. R10), ansonsten eine 16-Bit-Adresse (z.B. WR1). Wenn Sie eine Adresse angegeben, müssen bei SDTs mit nicht-überlappenden Elementen entweder alle oder kein Element vom Typ BOOL sein.
6. **Unter "Typ"**  **wählen, um das Dialogfeld "Typ-Auswahl" zu öffnen**
7. **Unter "Typklasse" "Strukt. Datentyp" wählen**
8. **Unter "Typ" den gewünschten SDT wählen**
9. **[OK]**
10. **Unter "Initial"**  **wählen, um das Dialogfeld "Vorgabewerte für strukturierten Datentyp" zu öffnen**

    Dieses Dialogfeld zeigt, wie die einzelnen Variablen im SDT definiert worden sind. Sie können nur die Initialisierungswerte für eine einzelne Variable (nicht für den SDT) ändern.
11. **Gegebenenfalls Initialisierungswert für gewünschte Variable ändern**
12. **[OK]**
13. **Unter "Kommentar" gegebenenfalls einen Text eingeben**
14. **Objekt → Speichern**

◆HINWEIS

Um einen in der globalen Variablenliste deklarierten SDT in einem POE-Rumpf verwenden zu können, muss er vorher in den Kopf der entsprechenden POE kopiert werden.

SDT im POE-Kopf verwenden

1. **POE-Kopf in Fenster "Projekt" öffnen**
2. **Gegebenenfalls neue Zeile mit**  **oder**  **einfügen**
3. **Unter "Klasse" "VAR" auswählen**
4. **Unter "Bezeichner" symbolischen Name eingeben**
5. **Unter "Typ"**  **wählen, um das Dialogfeld "Typ-Auswahl" zu öffnen**
6. **Unter "Typklasse" ”Strukt. Datentyp” wählen**
7. **Unter "Typ" den gewünschten SDT wählen**
8. **[OK]**
9. **Unter "Initial"**  **wählen, um das Dialogfeld "Vorgabewerte für strukturierten Datentyp" zu öffnen**

   Dieses Dialogfeld zeigt, wie die einzelnen Variablen im SDT definiert worden sind. Sie können nur die Initialisierungswerte für eine einzelne Variable (nicht für den SDT) ändern.
10. **Gegebenenfalls Initialisierungswert für gewünschte Variable ändern**
11. **[OK]**
12. **Unter "Kommentar" gegebenenfalls einen Text eingeben**
13. **Objekt → Speichern**

    Jetzt können Sie den SDT bzw. eine einzelne Variable des SDT im POE-Rumpf verwenden. Nehmen Sie die Zuweisung des SDT über das Fenster "Variablen" (**<F2>)** vor.

◆HINWEIS

Um einen in der globalen Variablenliste deklarierten SDT in einem POE-Rumpf verwenden zu können, muss er vorher in den Kopf der entsprechenden POE kopiert werden.

1.4.4 Array

Arrays

Ein Array ist eine Gruppe von Variablen, die alle den **gleichen** elementaren Datentyp haben und hintereinander in einem zusammenhängenden Block angeordnet sind. Diese Variablengruppe stellt selbst wieder eine Variable dar und wird daher auch deklariert. Im Programm können Sie dann das gesamte Array oder einzelne Array-Elemente verwenden.

Deklaration

Verwenden Sie für die Deklaration von ARRAY-Variablen im POE-Kopf die folgende Syntax:

ARRAY[A...B,C...D,E...F] OF <Datentyp>, wobei:

A=	erster Elementindex	erste Dimension (s. S. 51)
B=	letzter Elementindex	
C=	erster Elementindex	zweite Dimension (s. S. 52) (optional)
D=	letzter Elementindex	
E=	erster Elementindex	dritte Dimension (s. S. 53) (optional)
F=	letzter Elementindex	

Arrays können 1-, 2- oder 3-dimensional sein. In jeder Dimension kann ein Array mehrere Felder haben. Der

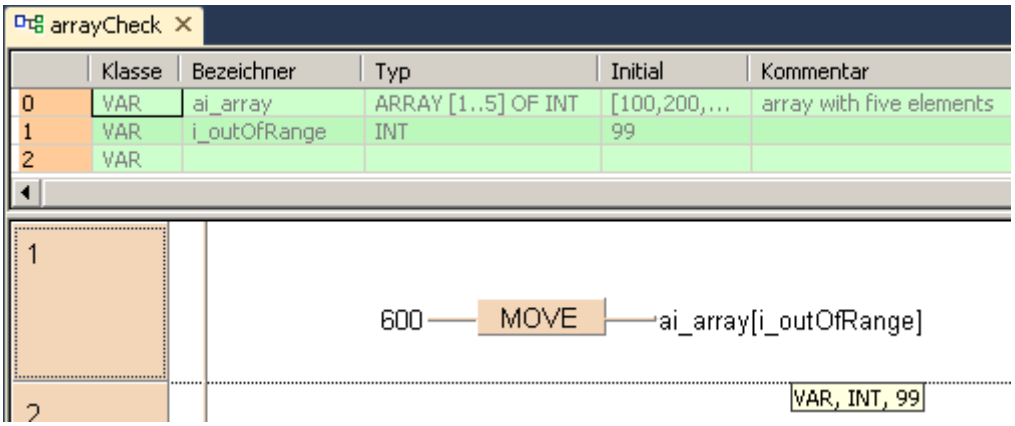
Elementindex kann eine positive oder negative Ganzzahl sein. Das erste Element muss kleiner sein als das letzte.

HINWEIS

- **Ein Array kann nicht als Variable eines anderen Arrays verwendet werden.**
- **Beim Zugriff auf einen Arrayindex wird von Control FPWIN Pro nicht geprüft, ob der Index in den Array-Grenzen liegt. Stellen Sie sicher, dass der Index nicht außerhalb des Bereichs liegt, der im POE-Kopf definiert ist.**

  Beispiel: ARRAY [1..5] OF INT

  ai_array[99] liegt hier außerhalb des Bereichs, es wird jedoch kein Fehler ausgegeben.



Gültige Datentypen für Arrays sind:

- BOOL
- DATE
- DATE_AND_TIME
- DINT
- DWORD
- INT
- REAL
- STRING
- TIME
- TIME_OF_DAY
- UDINT
- UINT
- WORD

Strukturierter Datentyp

Ein strukturierter Datentyp (SDT) besteht nur aus einer Gruppe von Variablen, die sich aus mehreren elementaren Datentypen (BOOL, WORD usw.) zusammensetzt. Solche Gruppen werden z. B. verwendet, wenn Tabellen verarbeitet werden wie für die Bitmusterausgabe beim Befehl F164_SPD0 (FP1, FP-M) der “FP-Library” (siehe Online-Hilfe). Die Bitmusterausgabe dieses Befehls können Sie z.B. zur Drehzahlregelung eines Motors über einen Drehzahlregler verwenden . Einen SDT definieren Sie zunächst im SDT-Pool. Sie können den SDT dann im Feld "Typ der globalen Variablenliste oder eines POE-Kopfs ähnlich verwenden wie die Datentypen Integer, BOOL etc. Im Programm können Sie dann entweder die gesamten SDT oder einzelne Variablen des SDT verwenden.

◆HINWEIS

Ein SDT kann nicht als Variable eines anderen SDT verwendet werden.

Weitere Informationen zu Arrays und SDTs finden Sie in der Online-Hilfe.

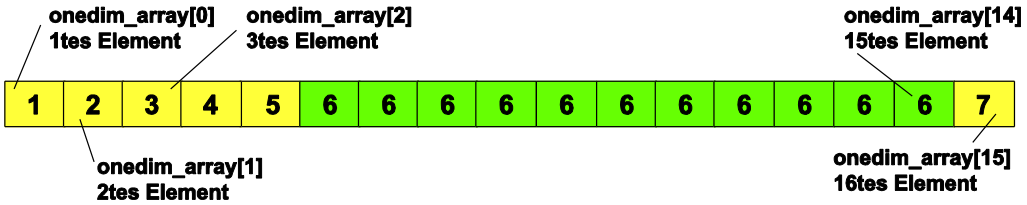
1.4.4.1 Eindimensionales ARRAY

Deklarieren in der Liste der globalen Variablen:

Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial	Kommentar
0	VAR_GLOBAL	onedim_array	DDT100	%MD5.100	DWORD	0	one-dimensional data field

Das deklarierte Array kann man sich folgendermaßen vorstellen:



Arrays mit Werten vorbelegen

Werden aufeinanderfolgende Array-Elemente mit den gleichen Werten initialisiert, ist die verkürzte Schreibweise **Anzahl(Wert)** möglich.

* **Anzahl** Anzahl der Array-Elemente

* **Wert** Initialisierungswert

Im Beispiel wurde das **Element 1** mit dem Wert 1, das **Element 2** mit dem Wert 2 usw. vorbelegt.

Eindimensionale Array-Elemente im Programm verwenden

Ein eindimensionales Array-Element verwenden Sie, indem Sie Bezeichner[Var1] eingeben.

* Bezeichner (Name des Arrays, siehe Feld "Bezeichner")

* **Var1** ist eine Variable vom Typ INT oder eine Konstante, die im Wertebereich der Array-Deklaration liegen muss. Für dieses Beispiel gilt Var1 im Bereich 0...15

Im Beispiel rufen Sie das dritte Array-Element (**Element 3**) mit **eindim_array[2]** auf. Wenn Sie z.B. in einem AWL-Programm diesem Element einen Wert zuweisen möchten, geben Sie folgendes ein:

```
LD  Isttemperatur

ST  eindim_array[2]
```

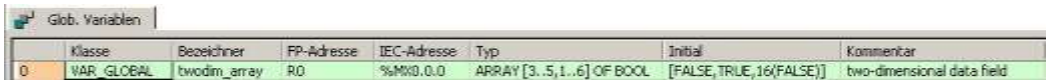
Adressen von Array-Elementen

Array-Elemente eines eindimensionalen Arrays werden nacheinander, beginnend mit Element 1 im Speicher der SPS abgelegt. Für das hier beschriebene Beispiel ergibt sich folgende Speicherbelegung:

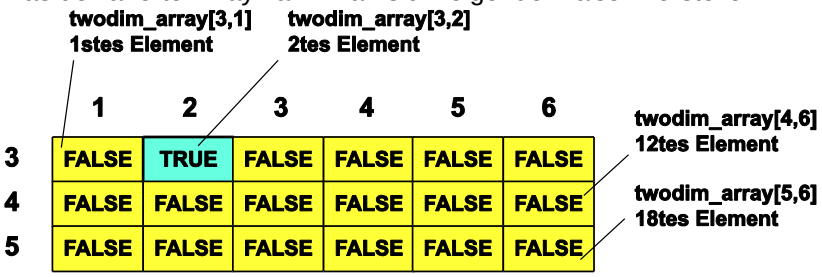
Array-Elementname	Array-Element	FP-Adresse	IEC-Adresse
eindim_array[0]	Element 1	DT0	%MW5.0
eindim_array[1]	Element 2	DT1	%MW5.1
eindim_array[2]	Element 3	DT2	%MW5.2
eindim_array[3]	Element 4	DT3	%MW5.3
eindim_array[4]	Element 5	DT4	%MW5.4
...	...	...	...
eindim_array[13]	Element 14	DT13	%MW5.13
eindim_array[14]	Element 15	DT14	%MW5.14
eindim_array[15]	Element 16	DT15	%MW5.15

1.4.4.2 Zweidimensionales ARRAY

Deklarieren in der Liste der globalen Variablen:



Das deklarierte Array kann man sich folgendermaßen vorstellen:



Arrays mit Werten vorbelegen

Die Initialisierung des Arrays mit Werten beginnt mit dem ersten Array-Element (Element 1) und endet mit dem letzten Array-Element (Element 18). Die Initialisierungswerte werden nacheinander in das Feld "Initial" eingetragen und durch Komma voneinander getrennt.

Werden aufeinanderfolgende Array-Elemente mit den gleichen Werten initialisiert, ist die verkürzte Schreibweise **Anzahl(Wert)** möglich.

* **Anzahl** Anzahl der Array-Elemente

* **Wert** Initialisierungswert

Im Beispiel wurde **Element 1** mit dem Wert FALSE, **Element 2** mit dem Wert TRUE und die restlichen Array-Elemente mit FALSE vorbelegt.

Zweidimensionale Array-Elemente im Programm verwenden

Ein zweidimensionales Array-Element verwenden Sie, indem Sie Bezeichner[Var1, Var2] eingeben.

* Bezeichner (Name des Arrays, siehe Feld "Bezeichner")

* **Var1** und **Var2** sind Variablen vom Typ INT oder Konstanten, die im Wertebereich der Array-Deklaration (siehe Feld "Typ") liegen müssen. Für dieses Beispiel liegt Var1 im Bereich 3...5 und Var2 im Bereich 1...6.

Im Beispiel rufen Sie mit **zweidim_array[4,6]** das Element 12 auf. Wenn Sie z.B. in einem AWL-Programm diesem Element einen Wert zuweisen möchten, geben Sie folgendes ein:

```
LD Motor_AN
```

```
ST  zweidim_array[4,6]
```

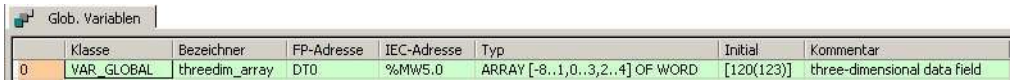
Adressen von zweidimensionalen Array-Elementen

Array-Elemente eines zweidimensionalen Arrays werden nacheinander, beginnend mit Element 1 im Speicher der SPS abgelegt. Für das hier beschriebene Beispiel ergibt sich folgende Speicherbelegung:

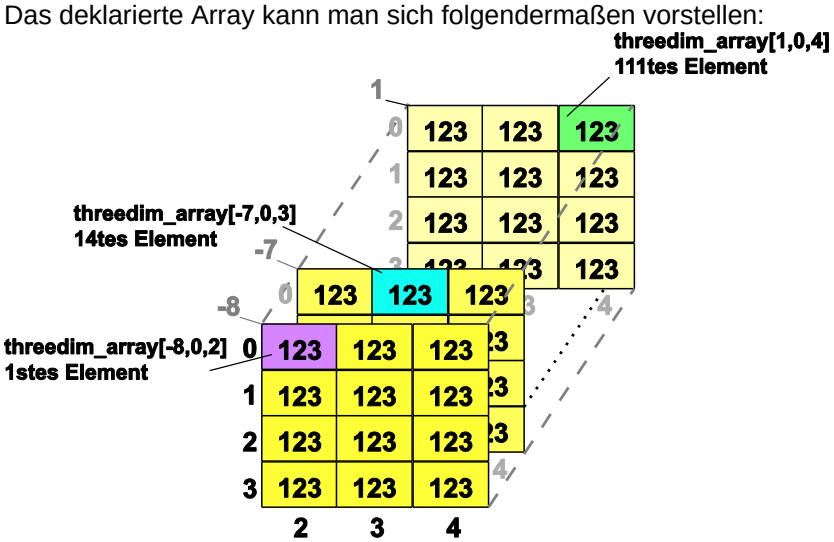
Array-Elementname	Array-Element	FP-Adresse	IEC-Adresse
zweidim_array[3,1]	Element 1	R0	%MX0.0.0
zweidim_array[3,2]	Element 2	R1	%MX0.0.1
zweidim_array[3,3]	Element 3	R2	%MX0.0.2
...	...	...	...
zweidim_array[3,6]	Element 6	R5	%MX0.0.5
zweidim_array[4,1]	Element 7	R6	%MX0.0.6
zweidim_array[4,2]	Element 8	R7	%MX0.0.7
...	...	...	...
zweidim_array[5,4]	Element 16	RF	%MX0.0.15
zweidim_array[5,5]	Element 17	R10	%MX0.1.0
zweidim_array[5,6]	Element 18	R11	%MX0.1.1

1.4.4.3 Dreidimensionales ARRAY

Deklarieren in der Liste der globalen Variablen:



Das deklarierte Array kann man sich folgendermaßen vorstellen:



Arrays mit Werten vorbelegen

Die Initialisierung des Arrays mit Werten beginnt mit dem ersten Array-Element (Element 1) und endet mit dem letzten Array-Element (Element 111). Die Initialisierungswerte werden nacheinander in das Feld "Initial" eingetragen und durch Komma voneinander getrennt.

Werden aufeinanderfolgende Array-Elemente mit den gleichen Werten initialisiert, ist die verkürzte Schreibweise **Anzahl(Wert)** möglich.

* **Anzahl** Anzahl der Array-Elemente

* **Wert** Initialisierungswert

Im Beispiel wurden alle Array-Elemente mit dem Wert 123 vorbelegt.

Dreidimensionale Array-Elemente im Programm verwenden

Der Zugriff auf ein dreidimensionales Array ist durch den Ausdruck Bezeichner[Var1,Var2,Var3, Var4].* möglich.
* Bezeichner ist der Name des Arrays (siehe Feld "Bezeichner").

* **Var1, Var2** und **Var3** sind Variablen von Typ INT oder Konstante, die im Wertebereich der Array-Deklaration (siehe Feld "Typ") liegen müssen. Für dieses Beispiel gilt Var1 im Bereich -8..1, Var2 im Bereich 0..3 und Var3 im Bereich 2..4.

Im Beispiel rufen Sie mit **dreidim_array[-7,0,4]** das Element 15 auf. Wenn Sie z.B. in einem AWL-Programm diesem Element einen Wert zuweisen möchten, geben Sie folgendes ein:

```
LD  Binaerwert

ST  dreidim_array[-7,0,4]
```

Adressen von dreidimensionalen Array-Elementen

Array-Elemente eines dreidimensionalen Arrays werden nacheinander, beginnend mit Element 1 im Speicher der SPS abgelegt. Für das hier beschriebene Beispiel ergibt sich folgende Speicherbelegung:

Array-Elementname	Array-Element	FP-Adresse	IEC-Adresse
dreidim_array[-8,0,2]	Element 1	DT0	%MW5.0
dreidim_array[-8,00,3]	Element 2	DT1	%MW5.1
dreidim_array[-8,00,4]	Element 3	DT2	%MW5.2
dreidim_array[-8,1,2]	Element 4	DT3	%MW5.3
dreidim_array[-8,10,3]	Element 5	DT4	%MW5.4
...	...	...	...
dreidim_array[-8,30,3]	Element 11	DT10	%MW5.10
dreidim_array[-8,30,4]	Element 12	DT11	%MW5.11
dreidim_array[-7,0,2]	Element 13	DT12	%MW5.12
dreidim_array[-7,00,3]	Element 14	DT13	%MW5.13
...	...	...	...
dreidim_array[1,3,2]	Element 118	DT117	%MW5.117
dreidim_array[1,30,3]	Element 119	DT118	%MW5.118
dreidim_array[1,30,4]	Element 120	DT119	%MW5.119

1.4.5 Spezielle Datentypen, nur verfügbar für Umwandlungsfunktionen

◆HINWEIS

- Gültige Datentypen sind: BOOL16, BOOL32, BOOLS, SDT, SDDT, BCD, IPADDR, ETLANADDR
- Diese Datentypen sind nur für Umwandlungsfunktionen zu speziellen Datentypen (s. S. 1207) gültig.
- Diese Datentypen können nicht in der Deklaration des POE-Kopfes ausgewählt werden.

1.4.5.1 BOOL16

Erlaubt sind:

- Arrays mit genau 16 Elementen vom Datentyp BOOL
  Hinweis:
  Diese können in den Bereichen X, Y, R, L, T, C liegen. Bei fehlendem Eintrag im Adressfeld der globalen Variablenliste oder für lokale Variablen wird sie der Compiler automatisch im R Bereich platzieren.

- Alle SDTs mit genau 16 Mitgliedern vom Datentyp BOOL
  Hinweis:
  Diese werden vom Compiler automatisch im R Bereich platziert.

1.4.5.2 BOOL32

Erlaubt sind:

- Arrays mit genau 32 Elementen vom Datentyp BOOL
  Hinweis:
  Diese können in den Bereichen X, Y, R, L, T, C liegen. Bei fehlendem Eintrag im Adressfeld der globalen Variablenliste oder für lokale Variablen wird sie der Compiler automatisch im R Bereich platzieren.

- Alle SDTs mit genau 32 Mitgliedern vom Datentyp BOOL
  Hinweis:
  Diese werden vom Compiler automatisch im R Bereich platziert.

1.4.5.3 BCD_WORD

Der Datentyp BCD_WORD (binär-codierte Dezimalzahl) tritt nur bei den Konvertierungsfunktionen INT_TO_BCD_WORD (s. S. 241) und UINT_TO_BCD_WORD (s. S. 242) auf. Diese Umwandlungsfunktionen verwenden Variablen vom Datentyp WORD die als BCD-Werte interpretiert werden. Zum Beispiel wird die Dezimalzahl 654 als Hexadezimalzahl 16#0654 interpretiert.

Siehe auch:

1.4.5.4 WORD_BCD

Der Datentyp WORD_BCD (binär-codierte Dezimalzahl) tritt nur bei den Konvertierungsfunktionen WORD_BCD_TO_INT (s. S. 147) und WORD_BCD_TO_UINT (s. S. 158) auf. Diese Umwandlungsfunktionen verwenden Variablen vom Datentyp WORD die als BCD-Werte interpretiert werden. Beispiel wird die Dezimalzahl 654 als Hexadezimalzahl 16#0654 interpretiert.

1.4.5.5 BCD_DWORD

Der Datentyp BCD_DWORD (binär-codierte Dezimalzahl) tritt nur bei den Konvertierungsfunktionen DINT_TO_BCD_DWORD (s. S. 242) und UDINT_TO_BCD_DWORD (s. S. 244) auf. Diese Umwandlungsfunktionen verwenden Variablen vom Datentyp DWORD, die als BCD-Werte interpretiert werden. Zum Beispiel wird die Dezimalzahl 654 als Hexadezimalzahl 16#0654 interpretiert.

1.4.5.6 DWORD_BCD

Der Datentyp DWORD_BCD (binär-codierte Dezimalzahl) tritt nur bei den Konvertierungsfunktionen DWORD_BCD_TO_DINT (s. S. 170) und DWORD_BCD_TO_UDINT (s. S. 183) auf. Diese Umwandlungsfunktionen verwenden Variablen vom Datentyp WORD die als BCD-Werte interpretiert werden. Zum Beispiel wird die Dezimalzahl 654 als Hexadezimalzahl 16#0654 interpretiert.

1.4.5.7 IPADDR

Der Datentyp IPADDR wird nur von den folgenden Funktionen verwendet:

- IPADDR_TO_STRING (s. S. 229)
- IPADDR_TO_STRING_NO_LEADING_ZEROS (s. S. 230)
- STRING_TO_IPADDR (s. S. 244)
- STRING_TO_IPADDR_STEPSAVER (s. S. 245)

Diese Umwandlungsfunktionen interpretieren Variablen vom Datentyp DWORD als Zeichenketten im IPADDR-Format. Dieses Format besteht aus vier Oktalzahlen (mit oder ohne führende Nullen) in gegensätzlicher Anordnung, jeweils durch einen Dezimalpunkt getrennt.

Beispiel:

Wert	Umwandlungsfunktion	Ergebnis
16#01020304	IPADDR_TO_STRING	004.003.002.001
	IPADDR_TO_STRING_NO _LEADING_ZEROS	4.3.2.1

◆HINWEIS

Für eine Umwandlung in gleichgerichteter Anordnung verwenden Sie die Funktionen für den Datentyp "ETLANADDR".

1.4.5.8 ETLANADDR

Der Datentyp ETLANADDR wird nur von den folgenden Funktionen verwendet:

- ETLANADDR_TO_STRING (s. S. 231)
- ETLANADDR_TO_STRING_NO_LEADING_ZEROS (s. S. 232)
- STRING_TO_ETLANADDR (s. S. 246)
- STRING_TO_ETLANADDR_STEPSAVER

Diese Umwandlungsfunktionen interpretieren Variablen vom Datentyp DWORD als Zeichenketten im ETLANADDR-Format. Dieses Format besteht aus vier Oktalzahlen (mit oder ohne führende Nullen) in gleichgerichteter Anordnung, jeweils durch einen Dezimalpunkt getrennt.

Beispiel:

Wert	Umwandlungsfunktion	Ergebnis
16#01020304	ETLANADDR_TO_STRING	001.002.003.004
	ETLANADDR_TO_STRING_ NO_LEADING_ZEROS	1.2.3.4

◆HINWEIS

Für eine Umwandlung in gegensätzlicher Anordnung verwenden Sie die Funktionen für den Datentyp "IPADDR".

1.4.5.9 ANY_IN_UNITS_OF_WORDS

Erlaubt sind:

- Datentypen INT, DINT, WORD, DWORD, REAL, STRING, TIME
- Arrays mit Datentypen außer BOOL
- Alle SDTs die Elemente mit Datentypen außer BOOL enthalten

  Hinweis:
  Diese Typen können in folgenden Bereichen liegen: WX, DWX, WY, DWY, WR, DWR, WL, DWL, SV, DSV, EV, DEV, DT, DDT, LD, DLD, FL, DFL. Bei fehlendem Eintrag im Adressfeld der globalen Variablenliste oder für lokale Variablen wird sie der Compiler automatisch entweder in DT, DDT, FL oder DFL platzieren.

- Arrays mit Datentyp BOOL, mit der Einschränkung, dass die Gesamtzahl der Elemente durch 16 teilbar sein muss.

  Hinweis:
  Diese Typen können in den Bereichen X, Y, R, L, T, C liegen. Bei fehlendem Eintrag im Adressfeld der globalen Variablenliste oder für lokale Variablen wird sie der Compiler automatisch entweder in R platzieren.

- Alle SDTs mit einer durch 16 teilbare Anzahl von einfachen BOOL-Variablen bestehen.

  Hinweis:
  Diese werden vom Compiler automatisch im Bereich R platziert.

1.4.5.10 ANY_SIMPLE_NOT_BOOL

Erlaubt sind:

Datentypen INT, DINT, WORD, DWORD, REAL, STRING, TIME (nicht aber BOOL)

Diese Datentypen können in folgenden Bereichen liegen:
WX, DWX, WY, DWY, WR, DWR, WL, DWL, SV, DSV, EV, DEV, DT, DDT, LD, DLD, FL, DFL.
Bei fehlendem Eintrag im Adressfeld der globalen Variablenliste oder für lokale Variablen wird sie der Compiler automatisch entweder in DT, DDT, FL oder DFL platzieren.

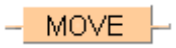
Kapitel 2

Transferbefehle

MOVE

Eingangsvariable an Ausgangsvariable

Erklärung MOVE weist den Wert der Eingangsvariablen unverändert dem Ausgang zu.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von MOVE (s. S. 1194)**

☞

- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Weitere Informationen finden Sie in der Online-Hilfe im Thema: Aktualisierungsprobleme bei Datentyp STRING**

Datentypen

Datentyp	E/A	Funktion
alle Datentypen	Eingang	Quelle
alle Datentypen	Ausgang wie Eingang	destination

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	INT	0	all types allowed
1	VAR	output_value	INT	0	all types allowed

In diesem Beispiel wurde die Eingangsvariable (**Input_value**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Input_value** wird unverändert **output_value** zugewiesen.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value := input_value;
```

Kapitel 3

Arithmetikbefehle

ADD

Addieren

Erklärung Der Inhalt des Akkumulators wird zu dem im Operandenbereich angegebenen Operanden addiert. Das Ergebnis der Addition steht im Akkumulator. In der Online-Hilfe finden Sie ein Programmierbeispiel zu dem Standard Operator ADD.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞

- **Alle Operanden müssen vom gleichen Datentyp sein.**
- **Die Funktion ist bis auf max. 28 Eingangskontakte erweiterbar.**

Datentypen

Datentyp	E/A	Funktion
INT, DINT, REAL	Eingang	Augend
INT, DINT, REAL	Eingang	Summand
INT, DINT, REAL	Ausgang wie Eingang	Summe

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

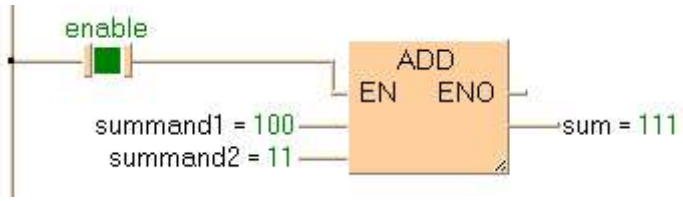
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	enable	BOOL	FALSE
1	VAR	summand_1	INT	0
2	VAR	summand_2	INT	0
3	VAR	sum	INT	0

In diesem Beispiel wurden die Eingangsvariablen (**summand_1, summand_2** und **enable**) deklariert. Stattdessen können Sie Konstanten direkt an die Eingänge der Funktion schreiben (z.B. für Tests).

Rumpf Wenn **enable** gesetzt ist (TRUE), wird **summand_1** zu **summand_2** addiert. Das Ergebnis wird in **sum** geschrieben.

KOP



SUB

Subtrahieren

Erklärung

Der im Operandenbereich angegebene Operand wird vom Akkumulator subtrahiert. Das Ergebnis der Subtraktion steht im Akkumulator.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞

- **Alle Operanden müssen vom gleichen Datentyp sein.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Datentypen

Datentyp	E/A	Funktion
INT, DINT, REAL	Eingang	Minuend
INT, DINT, REAL	Eingang	Subtrahend
INT, DINT, REAL	Ausgang wie Eingang	Ergebnis

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

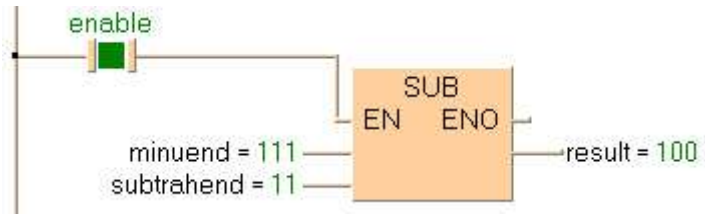
	Klasse	Bezeichner	Typ	Initial
0	VAR	enable	BOOL	FALSE
1	VAR	minuend	INT	0
2	VAR	subtragend	INT	0
3	VAR	result	INT	0

In diesem Beispiel wurden die Eingangsvariablen (**minuend, subtrahend** und **enable**) deklariert. Stattdessen können Sie Konstanten direkt an die Eingänge der Funktion schreiben (z.B. für Tests).

Rumpf

Wenn **enable** gesetzt ist, **subtrahend** (Datentyp INT) wird von **minuend** subtrahiert. Das Ergebnis wird in **result** (Datentyp INT) geschrieben.

KOP



MUL

Multiplizieren

Erklärung MUL multipliziert die Werte der beiden Eingangsvariablen miteinander und schreibt das Ergebnis in die Ausgangsvariable.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞

- **Alle Operanden müssen vom gleichen Datentyp sein.**
- **Die Funktion ist bis auf max. 28 Eingangskontakte erweiterbar.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Datentypen

Datentyp	E/A	Funktion
INT, DINT, REAL	Eingang	Multiplikand
INT, DINT, REAL	Eingang	Multiplikator
INT, DINT, REAL	Ausgang wie Eingang	Ergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

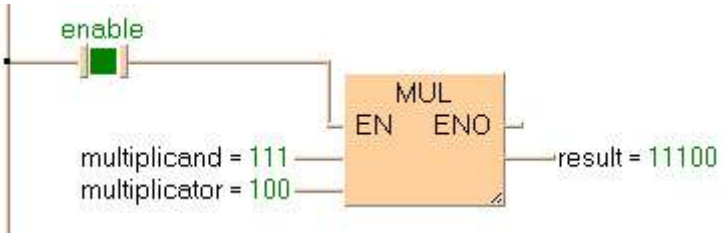
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	enable	BOOL	FALSE
1	VAR	multiplicand	INT	0
2	VAR	multiplicator	INT	0
3	VAR	result	INT	0

In diesem Beispiel wurden die Eingangsvariablen (**multiplicand, multiplicator** und **enable**) deklariert. Stattdessen können Sie Konstanten direkt an die Eingänge der Funktion schreiben (z.B. für Tests).

Rumpf Wenn **enable** gesetzt ist (TRUE), wird der **Multiplikant** mit dem **Multiplikator** multipliziert. Das Ergebnis wird in **result** geschrieben.

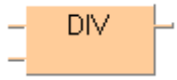
KOP



DIV

Dividieren

Erklärung Der Inhalt des Akkumulators wird durch den im Operandenfeld angegebenen Operanden dividiert. Das Ergebnis der Division steht im Akkumulator.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Datentypen

Datentyp	E/A	Funktion
INT, DINT, REAL	Eingang	Dividend
INT, DINT, REAL	Eingang	Divisor
INT, DINT, REAL	Ausgang wie Eingang	Ergebnis

☞

- **Die Eingangs- und Ausgangsvariablen müssen von einem der oben angegebenen Datentypen sein. Alle Operanden müssen vom gleichen Datentyp sein.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Beispiel In diesem Beispiel wird für alle Programmiersprachen derselbe POE-Kopf verwendet. Ein Beispiel für die Verwendung der Programmiersprache AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

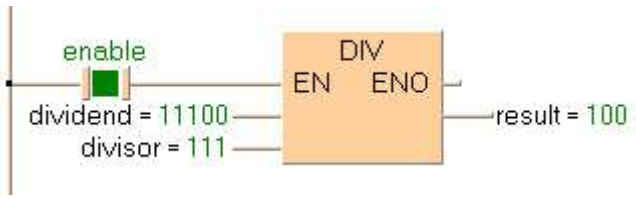
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	enable	BOOL	FALSE
1	VAR	dividend	INT	0
2	VAR	divisor	INT	0
3	VAR	result	INT	0

In diesem Beispiel wurden die Eingangsvariablen (**dividend, divisor** und **enable**) deklariert. Stattdessen können Sie Konstanten direkt an die Eingänge der Funktion schreiben (z.B. für Tests).

Rumpf Wenn **enable** gesetzt ist (TRUE), wird **dividend** durch **divisor** geteilt. Das Ergebnis wird in **result** geschrieben.

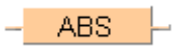
KOP



ABS

Absoluter Wert

Erklärung ABS rechnet den Wert, der im Akkumulator steht, in einen Absolutwert um. Das Ergebnis wird in der Ausgangsvariable gespeichert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von ABS (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
INT, DINT, REAL	Eingänge	Eingangsdatentyp
INT, DINT, REAL	Ausgang wie Eingang	absoluter Wert

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	input_value	INT	-123
1	VAR	absolute_value	INT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Der Eingangswert **Input_value** vom Datentyp INTEGER wird in einen Absolutwert vom Datentyp INTEGER umgewandelt. Der umgewandelte Wert wird in **absolute_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

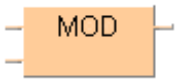
```
absolute_value:=ABS(input_value);
```

MOD

Ganzzahldivision, Rest wird in Ausgangsvariable gespeichert

Erklärung MOD dividiert den Wert der ersten Eingangsvariablen durch den Wert der zweiten. Der Rest der ganzzahligen Division (z.B. 5 : 2 = 2, **Rest = 1**) wird in die Ausgangsvariable geschrieben.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

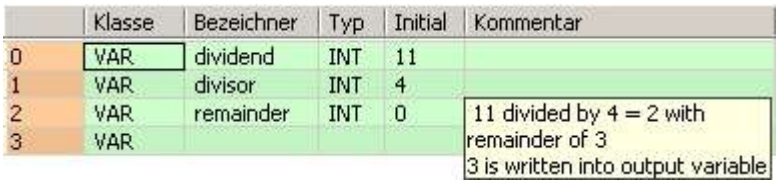
SPS-Typen **Verfügbarkeit von MOD (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
ANY_INT	Eingang	Dividend
	Eingang	Divisor
	Ausgang wie Eingang	Divisionsrest

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

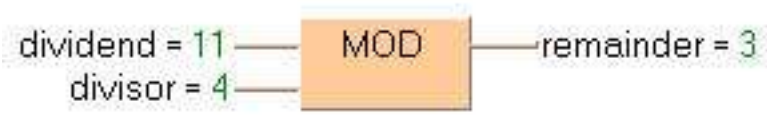
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	dividend	INT	11	
1	VAR	divisor	INT	4	
2	VAR	remainder	INT	0	11 divided by 4 = 2 with remainder of 3 3 is written into output variable
3	VAR				

Rumpf In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden. Dividend wird durch Divisor geteilt. Der ganzzahlige Rest der Division wird in **Divisionsrest** geschrieben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
remainder := dividend MOD divisor;
```

SQRT

Wurzelfunktion

Erklärung SQRT berechnet die Quadratwurzel der Eingangsvariablen vom Datentyp REAL (Wert ≥ 0.0). Das Ergebnis wird in die Ausgangsvariable geschrieben.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞ **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von SQRT (s. S. 1196)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingänge	Eingangswert
REAL	Ausgang wie Eingang	Wurzel des Eingangswerts

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Eingangsvariable nicht vom Datentyp REAL oder nicht ≥ 0,0 ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	REAL	0.0	number >= 0
1	VAR	output_value	REAL	0.0	number >= 0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Für **input_value** wird die Quadratwurzel berechnet und in **output_value** geschrieben.

KOP

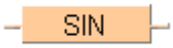
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value := SQRT (input_value);
```

SIN

Sinus im Eingang mit Bogenmaß

Erklärung SIN berechnet den Sinus der Eingangsvariablen und schreibt das Ergebnis in die Ausgangsvariable. Der Winkel muss im Bogenmaß (Wert < 52707176) angegeben werden.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞

- **Die Berechnungsgenauigkeit verringert sich, je größer der in der Eingangsvariablen angegebene Winkel ist. Sie sollten deshalb einen Winkel im Bogenmaß ≥ -2π und ≤ 2π eingeben.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von SIN (s. S. 1196)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingänge	Eingangswert, Winkel im Bogenmaß
REAL	Ausgang wie Eingang	Sinus des Eingangswerts

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Eingangsvariable nicht vom Datentyp REAL oder nicht ≥ 52707176 ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

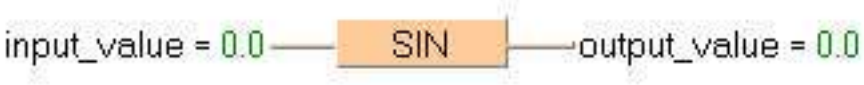
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	REAL	0.0	number >= 0
1	VAR	output_value	REAL	0.0	number >= 0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Für **input_value** wird der Sinuswert berechnet und in **output_value** geschrieben.

KOP



ASIN

Arkussinus

Erklärung ASIN berechnet den Arkussinus der Eingangsvariable und schreibt den Winkel im Bogenmaß in die Ausgangsvariable. Die Funktion liefert einen Wert von - π/2 bis π/2.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞ **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von ASIN (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingänge	Eingangswert zwischen -1 und +1
REAL	Ausgang wie Eingang	Arkussinus des Eingangswerts im Bogenmaß

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Eingangsvariable nicht vom Datentyp REAL oder nicht ≥ -1,0 und ≤ 1,0 ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	REAL	0.0	number between -1 and +1
1	VAR	output_value	REAL	0.0	angle data in radians -Pi/2 to +Pi/2

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Für **input_value** wird der Arkussinuswert berechnet und in **output_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=ASIN(input_value);
```

COS

Cosinus

Erklärung COS berechnet den Cosinus der Eingangsvariablen und schreibt das Ergebnis in die Ausgangsvariable. Der Winkel muss im Bogenmaß (Wert < 52707176) angegeben werden.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞

- **Die Berechnungsgenauigkeit verringert sich, je größer der in der Eingangsvariablen angegebene Winkel ist. Sie sollten deshalb einen Winkel im Bogenmaß ≥ -2π und ≤ 2π eingeben.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von COS (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingänge	Eingangswert, Winkel im Bogenmaß
REAL	Ausgang wie Eingang	Cosinus des Eingangswerts

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Eingangsvariable nicht vom Datentyp REAL oder nicht ≥ 52707176 ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	REAL	0.0	angle data in radians
1	VAR	output_value	REAL	0.0	cosine

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Für **input_value** wird der Cosinuswert berechnet und in **output_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=COS(input_value);
```

ACOS

Arkuscosinus

Erklärung ACOS berechnet den Arkuscosinus der Eingangsvariable und schreibt den Winkel im Bogenmaß in die Ausgangsvariable. Die Funktion liefert einen Wert von 0,0 bis π.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞ **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von ACOS (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingänge	Eingangswert zwischen -1 und +1
REAL	Ausgang wie Eingang	Arkuscosinus des Eingangswerts im Bogenmaß

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Eingangsvariable nicht vom Datentyp REAL oder nicht ≥ -1,0 und ≤ 1,0 ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	REAL	0.0	number between -1 and +1
1	VAR	output_value	REAL	0.0	angle data in radians 0.0 to pi

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Für **input_value** wird der Arkuscosinuswert berechnet und in **output_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=ACOS(input_value);
```

TAN Tangens

Erklärung TAN berechnet den Tangens der Eingangsvariablen und schreibt das Ergebnis in die Ausgangsvariable. Der Winkel muss im Bogenmaß (Wert < 52707176) angegeben werden.

Symbol: 

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞

- **Die Berechnungsgenauigkeit verringert sich, je größer der in der Eingangsvariablen angegebene Winkel ist. Sie sollten deshalb einen Winkel im Bogenmaß ≥ -2π und ≤ 2π eingeben.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von TAN (s. S. 1197)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingänge	Eingangswert im Bogenmaß
REAL	Ausgang wie Eingang	Tangens des Eingangswerts

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Eingangsvariable nicht vom Datentyp REAL oder nicht ≥ 52707176 ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	REAL	0.0	angle data in radians
1	VAR	output_value	REAL	0.0	tangent

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Für **input_value** wird der Tangenswert berechnet und in **output_value** geschrieben.

KOP

ATAN

Arkustangens

Erklärung ATAN berechnet den Arkustangens der Eingangsvariable (Wert ± 52707176) und den Winkel im Bogenmaß in die Ausgangsvariable. Die Funktion liefert einen Wert größer - π/2 und kleiner π/2.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞ **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von TAN (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingänge	Eingangswert zwischen -52707176 und +52707176
REAL	Ausgang wie Eingang	Arkussinus des Eingangswerts im Bogenmaß

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Eingangsvariable nicht vom Datentyp REAL oder nicht ≥ 52707176 ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	REAL	0.0	number between +/-52707176
1	VAR	output_value	REAL	0.0	angle in radians >-Pi/2 and <Pi/2

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Für **input_value** wird der Arkustangenswert berechnet und in **output_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

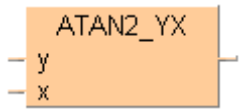
```
output_value:=ATAN(input_value);
```

ATAN2_YX

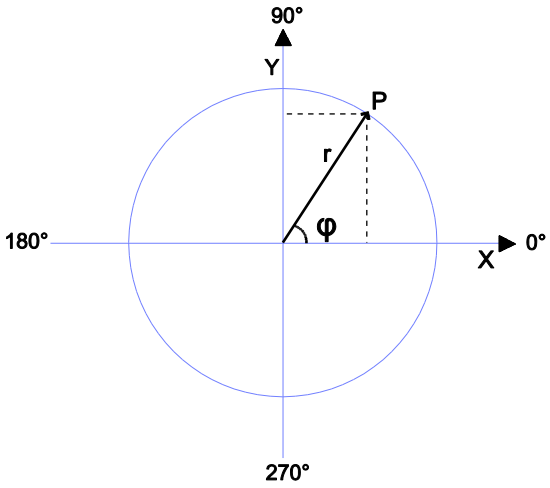
Gibt den Winkel f der kartesischen Koordinaten (x,y) zurück

Erklärung

ATAN2_YX gibt den Winkel φ der kartesischen Koordinaten (x,y) innerhalb des Bereichs -π bis +π zurück.



Jede zweidimensionale Koordinatenposition **P** kann über die kartesischen Koordinaten P(x,y) oder die Polarkoordinaten P(r,φ) (r = Radius, φ = Winkel) festgelegt werden.



Legen Sie ATAN2_YX wie folgt fest:

ATAN2_YX(y,x)	x	y
$\arctan\frac{y}{x}$	x > 0	
$\arctan\frac{y}{x}+\pi$	x < 0	y ≥ 0
$\arctan\frac{y}{x}-\pi$		y < 0
$+\frac{\pi}{2}$	x = 0	y > 0
$-\frac{\pi}{2}$		y < 0
0		y = 0

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von ATAN2_YX (s. S. 1184)

Datentypen

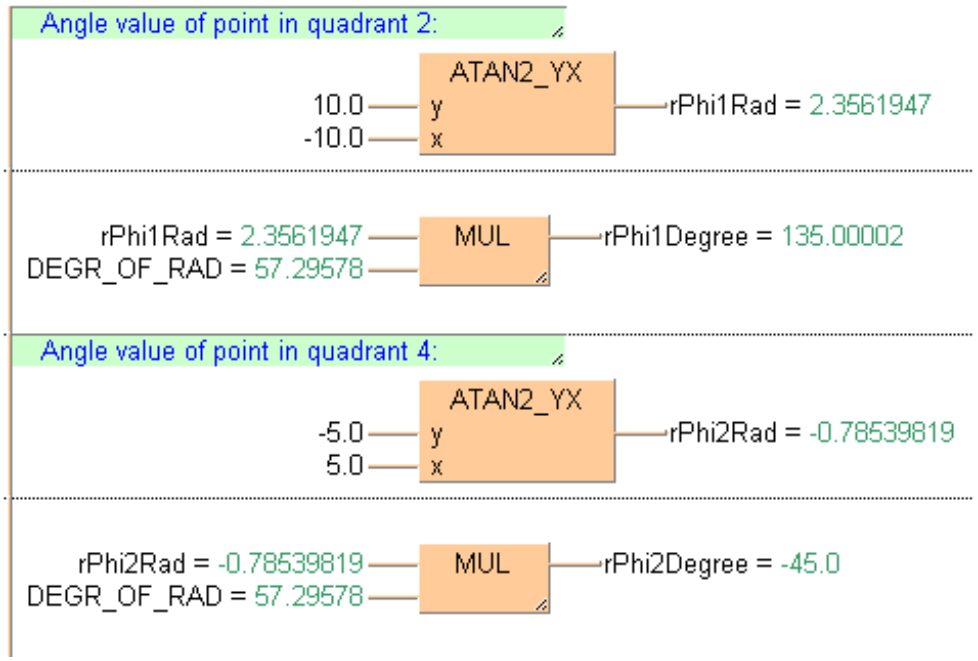
Datentyp	E/A	Funktion
REAL	y	Kartesische Y-Koordinate
REAL	x	Kartesische X-Koordinate

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	rPhi1Rad	REAL	0.0
1	VAR	rPhi2Rad	REAL	0.0
2	VAR	rPhi1Degree	REAL	0.0
3	VAR	rPhi2Degree	REAL	0.0
4	VAR_CONSTANT	DEGR_OF_RAD	REAL	57.295779513082320876798154814105
5	VAR	bCalculatePhi1	BOOL	FALSE

KOP

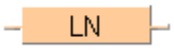


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
(* Angle value of point in quadrant 2 *)
rPhi1Rad:=ATAN2_YX(y := 10.0, x := -10.0); (* Result: 2.3561947 *)
rPhi1Degree := rPhi1Rad * DEGR_OF_RAD;      (* Result: 135.00002 *)

(* Angle value of point in quadrant 4 *)
rPhi2Rad:=ATAN2_YX(y := -5.0, x :=  5.0); (* Result: -0.78539819 *)
rPhi2Degree := rPhi2Rad * DEGR_OF_RAD;      (* Result: -45.0 *)
```

LN Natürlicher Logarithmus

Erklärung LN berechnet den Logarithmus der Eingangsvariablen (Wert > 0,0) zur Basis e (Eulersche Zahl = 2,7182818) und schreibt das Ergebnis in die Ausgangsvariable. Diese Funktion ist die Umkehrung der Funktion EXP (s. S. 78).

Symbol: 

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞ **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von LN (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingänge	Eingangswert
REAL	Ausgang wie Eingang	Natürlicher Logarithmus des Eingangswerts

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Eingangsvariable nicht vom Datentyp REAL und nicht > 0.0 ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	REAL	0.0	number > 0.0
1	VAR	output_value	REAL	0.0	number unequal 0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Für **input_value** wird der Logarithmus zur Basis e berechnet und in **output_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=LN(input_value);
```

LOG

Logarithmus zur Basis 10

Erklärung LOG berechnet den Logarithmus der Eingangsvariablen (Wert > 0,0) zur Basis 10 und schreibt das Ergebnis in die Ausgangsvariable.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞ **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von LOG (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingänge	Eingangswert
REAL	Ausgang wie Eingang	Logarithmus des Eingangswerts

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Eingangsvariable nicht vom Datentyp REAL oder nicht > 0,0 ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	REAL	0.0	number > 0.0
1	VAR	output_value	REAL	0.0	number unequal 0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Für **input_value** wird der Logarithmus zur Basis 10 berechnet und in **output_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=LOG(input_value);
```

EXP

Potenz der Eingangsvariablen zur Basis e

Erklärung EXP berechnet die Potenz der Eingangsvariablen zur Basis e (Eulersche Zahl = 2,7182818) und schreibt das Ergebnis in die Ausgangsvariable. Die Eingangsvariable muss größer -87,33 und kleiner 88,72 sein. Diese Funktion ist die Umkehrung der Funktion LN (s. S. 76).

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞ **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von EXP (s. S. 1186)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingänge	Eingangswert zwischen -87,33 und +88,72
REAL	Ausgang wie Eingang	Potenz der Eingangsvariablen zur Basis e

Fehlermer-ker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Eingangsvariable nicht vom Datentyp REAL oder nicht > -87,33 und < 88,72 ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

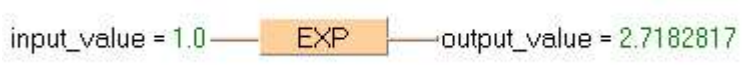
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	REAL	0.0	> -87.33 and < 88.72
1	VAR	output_value	REAL	0.0	number > 0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Für **input_value** wird die Potenz zur Basis e berechnet und in **output_value** geschrieben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=EXP(input_value);
```

EXPT

berechnet die Potenz der zweiten Eingangsvariablen zur Basis der ersten Eingangsvariablen

Erklärung EXPT berechnet die Potenz der zweiten Eingangsvariablen zur Basis der ersten Eingangsvariablen (OUT = $IN1^{IN2}$) und schreibt das Ergebnis in die Ausgangsvariable. Für die Eingangsvariablen sind Werte von -1.70141 x 10 E^{38} bis 1.70141 x 10 E^{38} zulässig.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von EXPT (s. S. 1186)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangswert
REAL	Eingang	Potenz des Eingangswerts
REAL	Ausgang wie Eingang	Ergebnis

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ 1. und 2. Eingangsvariable nicht den Datentyp REAL haben
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ die Ausgangsvariable Null ist
R9009	%MX0.900.9	kurzzeitig	▪ Ergebnis zum Überlauf der Ausgangsvariablen führt

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value_1	REAL	0.0	number from -1.70141x10^38 to
1	VAR	input_value_2	REAL	0.0	number from -1.70141x10^38 to
2	VAR	output_value	REAL	0.0	number from -1.70141x10^38 to 1.70141x10^38
3	VAR				

In diesem Beispiel wurden die Eingangsvariablen (**input_value_1** und **input_value_2**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Für **Input_value_2** wird die Potenz zur Basis **input_value_1** berechnet. Das Ergebnis wird in **output_value** geschrieben.

KOP

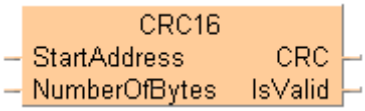
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=input_value_1**input_value_2;
```

CRC16 zyklische Blockprüfung

Erklärung

Diese Funktion berechnet für alle SPS-Typen die zyklische Blockprüfung CRC16 (**C**yclic **R**edundancy **C**heck) über eine Anzahl von Bytes (8 Bit), die mit dem Parameter **NumberOfBytes** und der Anfangsadresse **StartAddress** spezifiziert werden kann.



Je nach SPS-Typ wird eine der beiden folgenden Implementierungen der Funktion verwendet:

- Auf SPSen, die den Befehl F70_BCC (s. S. 405) mit dem Parameter s1=10 für CRC16-Berechnung unterstützen (FP-e, FP-Sigma, FP2, FP2SH, FP10SH) wird direkt der Befehl F70_BCC verwendet.
- Auf allen anderen SPSen (FP0, FP3, FP5, FP10) wird ein spezielles Unterprogramm aufgerufen, in dem die CRC16-Berechnung explizit ausprogrammiert wurde. Hier gelten folgende Beschränkungen:

Die ersten acht Ausführungszyklen bauen eine interne Tabelle auf. Während dieser Zeit wird keine Prüfsumme berechnet, der Ausgang **IsValid** bleibt FALSE. Erst danach wird die Prüfsumme korrekt berechnet und der Ausgang **IsValid** wird auf TRUE gesetzt

StartAddress erfordert eine Adresse in den Bereichen DT oder FL.

☞ **Die Anzahl der Schritte kann auf 200 ansteigen, wenn CRC16 als Unterprogramm verwendet wird.**

Generell muss der Zeitaufwand für den Aufbau der internen Tabelle und anschließend für die Berechnung der Prüfsumme insbesondere für größere Datenmengen sehr bewusst eingeplant werden.

SPS-Typen

Verfügbarkeit von CRC16 (s. S. 1184)

Datentypen

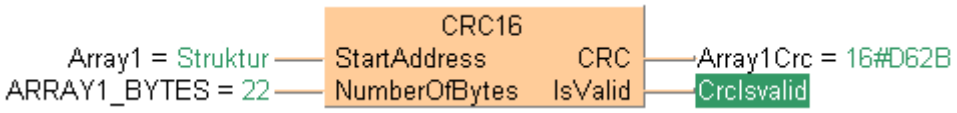
Eingangsvariablen (VAR_INPUT):		
Variable	**Datentyp**	**Funktion**
StartAddress	ANY	Beliebige Variable für die Anfangsadresse, ab der die Prüfsumme berechnet wird. Bei SPSen die den Befehl F70_BCC (s. S. 405) mit möglicher CRC16-Berechnung nicht unterstützen (FP0, FP3, FP5, FP10) muss sie im DT- oder FL-Bereich liegen.
NumberOfBytes	INT	Anzahl der Bytes (8 Bit) über die die Prüfsumme beginnend von AdrStart berechnet wird.
Ausgangsvariablen (VAR_OUTPUT):		
CRC	ANY16	Die berechnete Prüfsumme, die jedoch nur dann gültig ist, wenn der Merker **IsValid** auf TRUE gesetzt ist.
IsValid	BOOL	Merker, der angibt, ob die berechnete Prüfsumme gültig ist. Bei SPSen, die den Befehl F70_BCC (s. S. 405), mit möglicher CRC-Berechnung nicht unterstützen (FP0, FP3, FP5, FP10), ist die CRC-Berechnung nicht gültig: – während der ersten acht Zyklen, in welchen eine interne Tabelle aufgebaut wird – wenn der Adressbereich der Variablen StartAddress nicht im DT- oder FL-Bereich liegt. Für SPSen, die den Befehl F70_BCC mit CRC-Berechnung unterstützen, ist die berechnete Prüfsumme immer gültig.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Array1	ARRAY [0..10] OF INT	[0,1,2,3,4,5,6,7,8,9,10]
1	VAR	ARRAY1_BYTES	INT	22
2	VAR	Array1Crc	WORD	0
3	VAR	CrcIsvalid	BOOL	FALSE

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

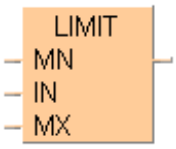
```
CRC16(StartAddress := Array1,
      NumberOfBytes := ARRAY1_BYTES,
      CRC => Array1Crc,
      IsValid => CrcIsvalid);
```

LIMIT

Begrenzer

Erklärung In LIMIT bildet die 1. Eingangsvariable den unteren und die 3. Eingangsvariable den oberen Grenzwert. Wenn die 2. Eingangsvariable innerhalb dieser Grenze liegt, wird die 2. Eingangsvariable an die Ausgangsvariable übertragen. Wenn die 2. Eingangsvariable darüber liegt, wird der obere, falls sie darunter liegt, der untere Grenzwert übertragen.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von LIMIT (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
Alle Datentypen	Eingang 1	Obere Grenze
Alle Datentypen	Eingang 2	Wert, der mit oberem und unterem Grenzwert verglichen wird
Alle Datentypen	Eingang 3	Untere Grenze
Alle Datentypen	Ausgang wie Eingang	Ergebnis ist die zweite Eingangsvariable, falls sie zwischen dem oberen und dem unteren Grenzwert liegt, sonst der obere oder untere Grenzwert

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

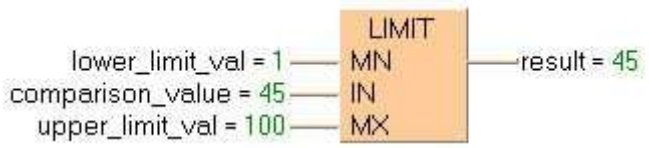
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	lower_limit_var	INT	0	all types allowed
1	VAR	comparision_value	INT	0	all types allowed
2	VAR	upper_limit_val	INT	0	all types allowed
3	VAR	result	INT	0	all types allowed

In diesem Beispiel wurden die Eingangsvariablen (**lower_limit_val, comparison_value** und **upper_val**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Lower_limit_val** und **upper_limit_val** bilden den Bereich, in dem sich der Vergleichswert **comparison_value** befinden muss, wenn er in **result** übertragen werden soll. Liegt der Vergleichswert **comparison_value** über dem oberen Grenzwert **upper_limit_val**, wird der Wert aus **upper_limit_val** in **result** übertragen. Liegt der Vergleichswert unter dem unteren Grenzwert **lower_limit_val**, wird der Wert aus **lower_limit_val** in **result** übertragen.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
result:=LIMIT(MN:=lower_limit_val, IN:=comparison_value,
MX:=upper_limit_val);
```

Kapitel 4

Bitweise Boolsche Befehle

AND

UND-Verknüpfung

Erklärung Der Inhalt des Akkumulators wird mit dem Ergebnis des in der Klammer stehenden Ausdrucks UND-verknüpft. Das Ergebnis der Operation steht im Akkumulator.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von AND (s. S. 1207)**

☞

- **Alle Operanden müssen vom gleichen Datentyp sein.**
- **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**

Datentypen

Datentyp	E/A	Funktion
BOOL, WORD, DWORD	Eingang	Element 1 der logischen AND-Operation
BOOL, WORD, DWORD	Eingang	Element relativ zu Eingang 1
BOOL, WORD, DWORD	Ausgang wie Eingang	Ergebnis

Wahrheitstabelle:

	Eingang 1	Eingang 2	Ausgang
Signal	0	0	0
	0	1	0
	1	0	0
	1	1	1

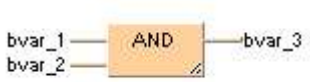
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bvar_1	BOOL	FALSE	Input 1
1	VAR	bvar_2	BOOL	FALSE	Input 2
2	VAR	bvar_3	BOOL	FALSE	Output

Rumpf **bvar_1** wird mit AND logisch mit **bvar_2** verknüpft. Das Ergebnis wird in die Ausgangsvariable **bvar_3** geschrieben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bvar_3 := bvar_1 &bvar_2;
```

OR

ODER-Verknüpfung

Erklärung

Der Inhalt des Akkumulators wird mit dem Ergebnis des in der Klammer stehenden Ausdrucks ODER-verknüpft. Das Ergebnis der Operation steht im Akkumulator.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von OR (s. S. 1207)

- **Alle Operanden müssen vom gleichen Datentyp sein.**
- **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**

Datentypen

Datentyp	E/A	Funktion
BOOL, WORD, DWORD	Eingang	Element 1 der logischen OR-Operation
BOOL, WORD, DWORD	Eingang	Element relativ zu Eingang 1
BOOL, WORD, DWORD	Ausgang wie Eingang	Ergebnis

Wahrheitstabelle:

	Eingang 1	Eingang 2	Ausgang
Signal	0	0	0
	1	0	1
	0	1	1
	1	1	1

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

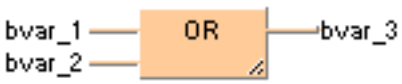
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bvar_1	BOOL	FALSE	Input 1
1	VAR	bvar_2	BOOL	FALSE	Input 2
2	VAR	bvar_3	BOOL	FALSE	Output

Rumpf

bvar_1 und **bvar_2** werden mit einem logischen OR verknüpft. Das Ergebnis wird in **bvar_3** geschrieben. In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bvar_3 := var_1 OR bvar_2;
```

XOR

Exklusiv-ODER

Erklärung Der Inhalt des Akkumulators wird mit dem Ergebnis des in der Klammer stehenden Ausdrucks XOR-verknüpft. Das Ergebnis der Operation steht im Akkumulator.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von XOR (s. S. 1207)**

- **Alle Operanden müssen vom gleichen Datentyp sein.**
- **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**

Datentypen

Datentyp	E/A	Funktion
ANY_BIT	Eingang	Element 1 der logischen XOR-Operation
	Eingang	Element relativ zu Eingang 1
	Ausgang wie Eingang	Ergebnis

Wahrheitstabelle:

Signal	Eingang 1	Eingang 2	Ausgang
	0	0	0
	1	0	1
	0	1	1
	1	1	0

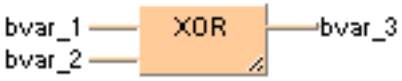
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bvar_1	BOOL	FALSE	Input 1
1	VAR	bvar_2	BOOL	FALSE	Input 2
2	VAR	bvar_3	BOOL	FALSE	Output

Rumpf Die booleschen Variablen **bvar_1** und **bvar_2** werden mit einem Exklusiv_ODER verknüpft und das Ergebnis in **bvar_3** geschrieben.

KOP



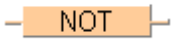
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
var_3 := var_1 XOR var_2;
```

NOT

Negation

Erklärung NOT führt eine Bit-Invertierung der Eingangsvariablen durch. Das Ergebnis wird in die Ausgangsvariable geschrieben.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von NOT (s. S. 1194)**

☞ **Alle Operanden müssen vom gleichen Datentyp sein.**

Datentypen

Datentyp	E/A	Funktion
BOOL, WORD, DWORD	Eingänge	Eingang der NOT-Operation
BOOL, WORD, DWORD	Ausgang wie Eingang	Ergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

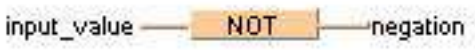
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	WORD	0	type: BOOL, WORD or DWORD
1	VAR	negation	WORD	0	type: BOOL, WORD or DWORD

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Die Bits von **input_value** werden invertiert (0 wird 1 und umgekehrt). Das invertierte Ergebnis wird in **negation** geschrieben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
negation := NOT(input_value);
```

Kapitel 5

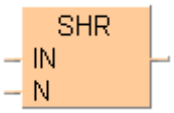
Bit-Schiebefunktionen

SHR

N Bits nach rechts schieben, links mit Nullen füllen

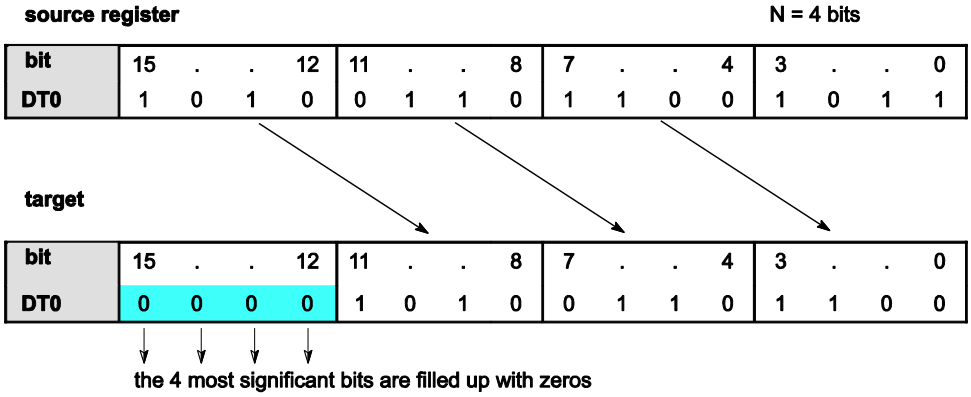
Erklärung SHR schiebt einen Bit-Wert um eine definierte Anzahl Stellen (N) nach rechts und füllt die freiwerdenden Stellen mit Nullen auf.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Bit-Verschiebung nach rechts, links werden Nullen aufgefüllt.



SPS-Typen **Verfügbarkeit von SHR (s. S. 1196)**

Datentypen

Datentyp	E/A	Funktion
BOOL, WORD, DWORD	Eingang	Eingangswert
BOOL, WORD, DWORD	Eingang	Anzahl der Bits, bei der der Eingangswert nach rechts verschoben wird
BOOL, WORD, DWORD	Ausgang wie Eingang	Ergebnis

☞

- **Wenn die 2. Eingangsvariable N (die Anzahl von Bits, um die geschoben wird) vom Typ DWORD ist, dann werden nur die niedrigeren 16-Bit berücksichtigt.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

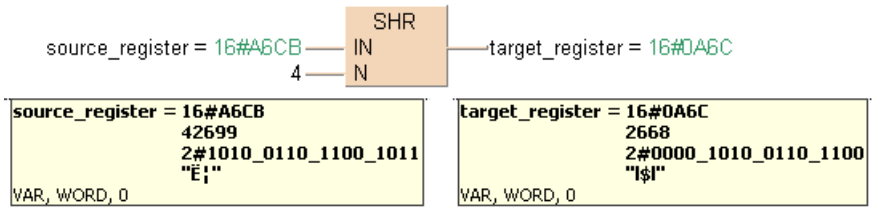
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	source_register	WORD	0
1	VAR	target_register	WORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Die letzten N Bits (hier 4) von **source_register** werden nach rechts verschoben. Die links frei werdenden Stellen werden mit Nullen aufgefüllt. Das Ergebnis wird in **target_register** geschrieben.

KOP

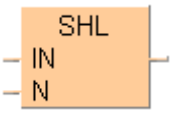


SHL

N Bits nach links schieben, rechts mit Nullen füllen

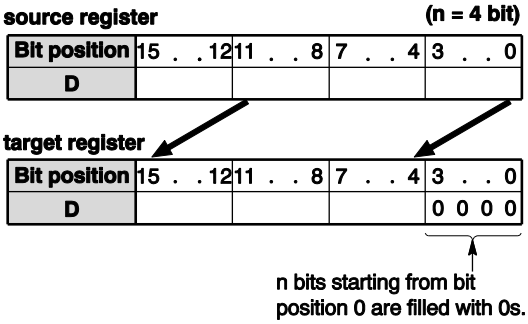
Erklärung SHL schiebt einen Bit-Wert um eine definierte Anzahl Stellen (N) nach links und füllt die freiwerdenden Stellen mit Nullen auf.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Bit-Verschiebung nach links, rechts werden Nullen aufgefüllt.



SPS-Typen **Verfügbarkeit von SHL (s. S. 1196)**

Datentypen

Datentyp	E/A	Funktion
BOOL, WORD, DWORD	Eingang	Eingangswert
BOOL, WORD, DWORD	Eingang	Anzahl der Bits, bei der der Eingangswert nach links verschoben wird
BOOL, WORD, DWORD	Ausgang wie Eingang	Ergebnis

☞

- **Wenn die 2. Eingangsvariable N (die Anzahl von Bits, um die geschoben wird) vom Typ DWORD ist, dann werden nur die niedrigeren 16-Bit berücksichtigt.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

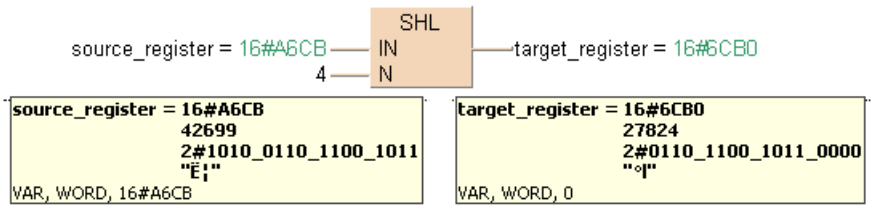
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	source_register	WORD	0
1	VAR	target_register	WORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Die ersten N Bits (hier 4) von **source_register** werden nach links verschoben. Die rechts frei werdenden Stellen werden mit Nullen aufgefüllt. Das Ergebnis wird in **target_register** geschrieben.

KOP

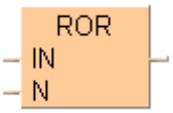


ROR

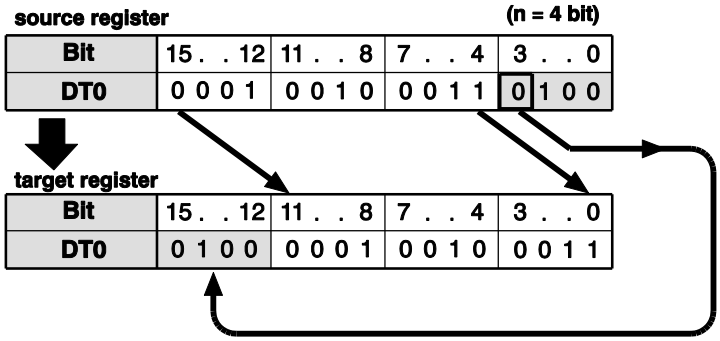
rotiert eine definierte Anzahl (n) Bits nach rechts.

Erklärung ROR rotiert eine definierte Anzahl (n) Bits nach rechts.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.



SPS-Typen **Verfügbarkeit von ROR (s. S. 1196)**

Datentypen

Datentyp	E/A	Funktion
BOOL, WORD, DWORD	Eingang	Eingangswert
BOOL, WORD, DWORD	Eingang	Anzahl der Bits, bei der der Eingangswert nach rechts rotiert wird
BOOL, WORD, DWORD	Ausgang wie Eingang	Ergebnis

☞ **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

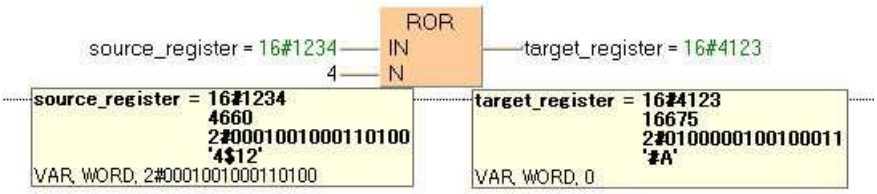
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	source_register	WORD	0
1	VAR	target_register	WORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Die ersten N Bits (hier 4) von **source_register** werden nach rechts rotiert. Das Ergebnis wird in **target_register** geschrieben.

KOP

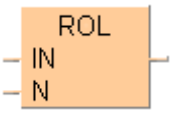


ROL

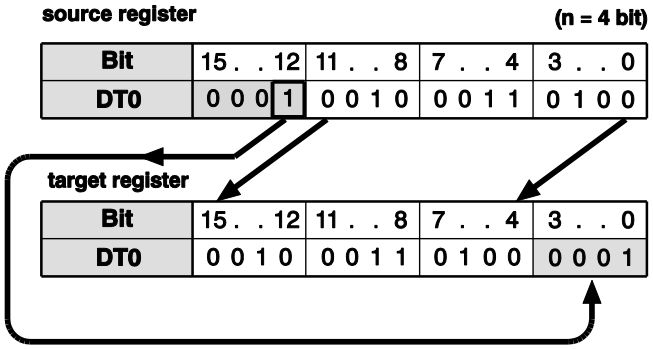
n Bits nach links rotieren

Erklärung ROL rotiert eine definierte Anzahl (n) Bits nach links.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.



SPS-Typen **Verfügbarkeit von ROL (s. S. 1196)**

Datentypen

Datentyp	E/A	Funktion
BOOL, WORD, DWORD	Eingang	Eingangswert
BOOL, WORD, DWORD	Eingang	Anzahl der Bits, bei der der Eingangswert nach links rotiert wird
BOOL, WORD, DWORD	Ausgang wie Eingang	Ergebnis

☞ **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

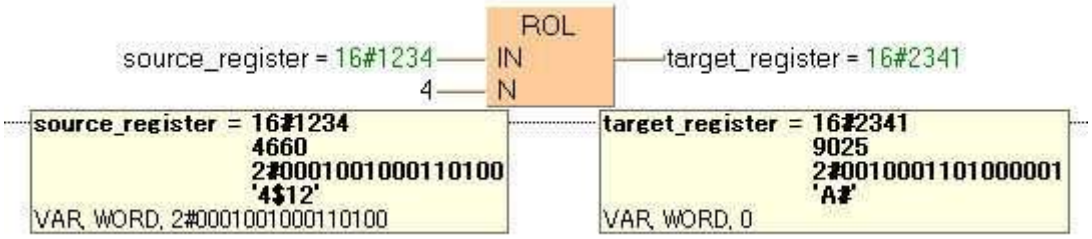
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	source_register	WORD	0
1	VAR	target_register	WORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Die letzten N Bits (hier 4) von **source_register** werden nach links rotiert. Das Ergebnis wird in **target_register** geschrieben.

KOP


ROL
source_register = 16#1234
IN
target_register = 16#2341
4
N
source_register = 16#1234
4660
2#0001001000110100
'4$12'
VAR, WORD, 2#0001001000110100
target_register = 16#2341
9025
2#0010001101000001
'A#'
VAR, WORD, 0

Kapitel 6

Vergleichsfunktionen

GT

Größer als

Erklärung Der Inhalt des Akkumulators wird mit dem im Operandenbereich angegebenen Operanden auf Gleichheit geprüft. Wenn der Akkumulator größer als der Operand ist, wird als Ergebnis im Akkumulator "TRUE" abgelegt, ansonsten "FALSE".

Symbol: 

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von GT (s. S. 1207)**

- **Die Eingänge können jeden Datentyp aufweisen; alle Eingangsvariablen müssen jedoch vom selben Typ sein. Der Ausgang muss vom Typ BOOL sein.**
- **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Datentypen

Datentyp	E/A	Funktion
alle Datentypen	Eingang	Wert für Vergleich
alle Datentypen	Eingang	Referenzwert
BOOL	Ausgabe	Ergebnis, TRUE wenn Wert für Vergleich größer dem Referenzwert

Variablen, die miteinander verglichen werden, müssen denselben Datentyp aufweisen.

Bei mehreren Eingängen wird der erste mit dem zweiten verglichen, der zweite mit dem dritten etc. Ist der erste Wert größer dem zweiten Wert UND der zweite Wert größer dem dritten Wert, wird TRUE als Ergebnis geschrieben, andernfalls FALSE.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

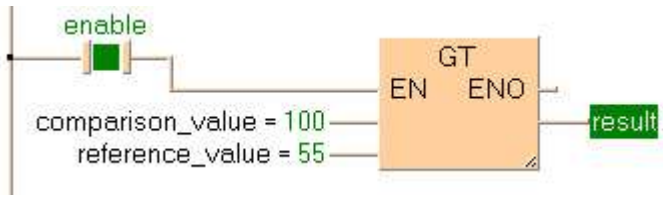
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	enable	BOOL	FALSE
1	VAR	comparison_value	INT	0
2	VAR	reference_value	INT	0
3	VAR	result	BOOL	FALSE

In diesem Beispiel wurden die Eingangsvariablen (**comparison_value, reference_value** und **enable**) deklariert. Stattdessen können Sie Konstanten direkt an die Eingänge der Funktion schreiben (z.B. für Tests).

Rumpf Wenn **enable** gesetzt ist (TRUE), wird **comparison_value** mit der Variablen **reference_value** verglichen. Wenn **comparison_value** größer als **reference_value** ist, wird in **result** der Wert TRUE, sonst FALSE geschrieben.

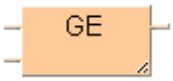
KOP


enable
GT
EN
ENO
comparison_value = 100
reference_value = 55
result

GE

Größer gleich (Greater or Equal)

Erklärung Der Inhalt des Akkumulators wird mit dem im Operandenbereich angegebenen Operanden auf Gleichheit geprüft. Wenn der Akkumulator größer oder gleich dem Operand ist, wird als Ergebnis im Akkumulator "TRUE" abgelegt, ansonsten "FALSE".



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von GE (s. S. 1207)**

- **Die Eingänge können jeden Datentyp aufweisen; alle Eingangsvariablen müssen jedoch vom selben Typ sein. Der Ausgang muss vom Typ BOOL sein.**
- **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Datentypen

Datentyp	E/A	Funktion
alle Datentypen	Eingang	Wert für Vergleich
alle Datentypen	Eingang	Referenzwert
BOOL	Ausgabe	Ergebnis, TRUE wenn Wert für Vergleich größer oder gleich dem Referenzwert

Variablen, die miteinander verglichen werden, müssen denselben Datentyp aufweisen.

Bei mehreren Eingängen wird der erste mit dem zweiten verglichen, der zweite mit dem dritten etc. Ist der erste Wert größer oder gleich dem zweiten Wert UND der zweite Wert größer oder gleich dem dritten Wert, wird TRUE als Ergebnis geschrieben, andernfalls FALSE.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

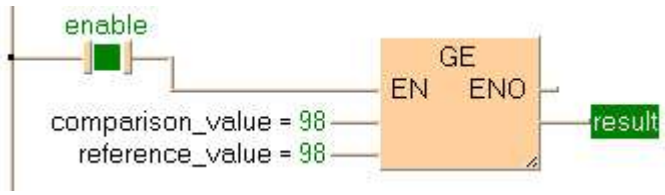
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	enable	BOOL	FALSE
1	VAR	comparison_value	INT	0
2	VAR	reference_value	INT	0
3	VAR	result	BOOL	FALSE

In diesem Beispiel wurden die Eingangsvariablen (**comparison_value, reference_value** und **enable**) deklariert. Stattdessen können Sie Konstanten direkt an die Eingänge der Funktion schreiben (z.B. für Tests).

Rumpf Wenn **enable** gesetzt ist (TRUE), wird **comparison_value** mit der Variablen **reference_value** verglichen. Wenn **comparison_value** größer oder gleich dem Referenzwert **reference_value** ist, wird in **result** der Wert TRUE, sonst FALSE geschrieben.

KOP


enable
GE
EN
ENO
comparison_value = 98
reference_value = 98
result

EQ

Gleich (Equal)

Erklärung Der Inhalt des Akkumulators wird mit dem im Operandenbereich angegebenen Operanden auf Gleichheit geprüft. Wenn der Akkumulator und der Operand gleich sind, wird als Ergebnis im Akkumulator "TRUE" abgelegt ansonsten "FALSE".

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von EQ (s. S. 1207)**

☞

- **Die Eingänge können jeden Datentyp aufweisen; alle Eingangsvariablen müssen jedoch vom selben Typ sein. Der Ausgang muss vom Typ BOOL sein.**
- **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Datentypen

Datentyp	E/A	Funktion
alle Datentypen	Eingang	Wert für Vergleich
alle Datentypen	Eingang	Referenzwert
BOOL	Ausgabe	Ergebnis, TRUE wenn Wert für Vergleich gleich dem Referenzwert

Variablen, die miteinander verglichen werden, müssen denselben Datentyp aufweisen.

Bei mehreren Eingängen wird der erste mit dem zweiten verglichen, der zweite mit dem dritten etc. Ist der erste Wert gleich dem zweiten Wert UND der zweite Wert gleich dem dritten Wert, wird TRUE als Ergebnis geschrieben, andernfalls FALSE.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

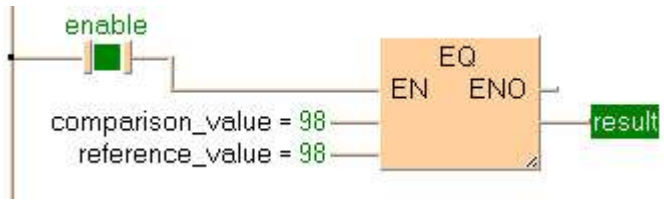
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	enable	BOOL	FALSE
1	VAR	comparison_value	INT	0
2	VAR	reference_value	INT	0
3	VAR	result	BOOL	FALSE

In diesem Beispiel wurden die Eingangsvariablen (**comparison_value, reference_value** und **enable**) deklariert. Stattdessen können Sie Konstanten direkt an die Eingänge der Funktion schreiben (z.B. für Tests).

Rumpf Wenn **enable** gesetzt ist (TRUE), wird **comparison_value** mit der Variablen **reference_value** verglichen. Wenn die Werte der beiden Variablen gleich sind, wird in **result** der Wert TRUE, sonst FALSE geschrieben.

KOP


enable
EQ
EN
ENO
comparison_value = 98
reference_value = 98
result

LE

Kleiner gleich (Less or Equal)

Erklärung Der Inhalt des Akkumulators wird mit dem im Operandenbereich angegebenen Operanden auf Gleichheit geprüft. Wenn der Akkumulator kleiner oder gleich dem Operand ist, wird als Ergebnis im Akkumulator "TRUE" abgelegt, ansonsten "FALSE".

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von LE (s. S. 1207)**

☞

- **Die Eingänge können jeden Datentyp aufweisen; alle Eingangsvariablen müssen jedoch vom selben Typ sein. Der Ausgang muss vom Typ BOOL sein.**
- **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Datentypen

Datentyp	E/A	Funktion
alle Datentypen	Eingang	Wert für Vergleich
alle Datentypen	Eingang	Referenzwert
BOOL	Ausgabe	Ergebnis, TRUE wenn Wert für Vergleich kleiner oder gleich dem Referenzwert

Variablen, die miteinander verglichen werden, müssen denselben Datentyp aufweisen.

Bei mehreren Eingängen wird der erste mit dem zweiten verglichen, der zweite mit dem dritten etc. Ist der erste Wert kleiner oder gleich dem zweiten Wert UND der zweite Wert kleiner oder gleich dem dritten Wert, wird TRUE als Ergebnis geschrieben, andernfalls FALSE.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

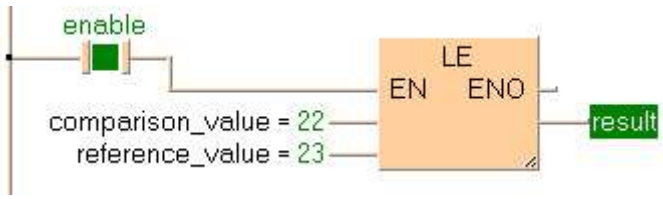
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	enable	BOOL	FALSE
1	VAR	comparison_value	INT	0
2	VAR	reference_value	INT	0
3	VAR	result	BOOL	FALSE

In diesem Beispiel wurden die Eingangsvariablen (**comparison_value, reference_value** und **enable**) deklariert. Stattdessen können Sie Konstanten direkt an die Eingänge der Funktion schreiben (z.B. für Tests).

Rumpf Wenn **enable** gesetzt ist (TRUE), wird **comparison_value** mit der Variablen **reference_value** verglichen. Wenn **comparison_value** kleiner oder gleich dem Referenzwert **reference_value** ist, wird in **result** der Wert TRUE, sonst FALSE geschrieben.

KOP


enable
LE
EN
ENO
comparison_value = 22
reference_value = 23
result

LT

Kleiner als (Lower Than)

Erklärung Der Inhalt des Akkumulators wird mit dem im Operandenbereich angegebenen Operanden auf Gleichheit geprüft. Wenn der Akkumulator kleiner als der Operand ist, wird als Ergebnis im Akkumulator "TRUE" abgelegt, ansonsten "FALSE".

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von LT (s. S. 1207)**

☞

- **Die Eingänge können jeden Datentyp aufweisen; alle Eingangsvariablen müssen jedoch vom selben Typ sein. Der Ausgang muss vom Typ BOOL sein.**
- **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Datentypen

Datentyp	E/A	Funktion
alle Datentypen	Eingang	Wert für Vergleich
alle Datentypen	Eingang	Referenzwert
BOOL	Ausgabe	Ergebnis, TRUE wenn Wert für Vergleich kleiner dem Referenzwert

Variablen, die miteinander verglichen werden, müssen denselben Datentyp aufweisen.

Bei mehreren Eingängen wird der erste mit dem zweiten verglichen, der zweite mit dem dritten etc. Ist der erste Wert kleiner dem zweiten Wert UND der zweite Wert kleiner dem dritten Wert, wird TRUE als Ergebnis geschrieben, andernfalls FALSE.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

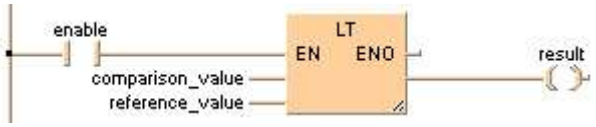
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	enable	BOOL	FALSE
1	VAR	comparison_value	INT	0
2	VAR	reference_value	INT	0
3	VAR	result	BOOL	FALSE

In diesem Beispiel wurden die Eingangsvariablen (**comparison_value, reference_value** und **enable**) deklariert. Stattdessen können Sie Konstanten direkt an die Eingänge der Funktion schreiben (z.B. für Tests).

Rumpf Wenn **enable** gesetzt ist (TRUE), wird **comparison_value** mit der Variablen **reference_value** verglichen. Wenn **comparison_value** kleiner oder gleich dem Referenzwert **reference_value** ist, wird in **result** der Wert TRUE, sonst FALSE geschrieben.

KOP



NE

Ungleich (Not Equal)

Erklärung

Der Inhalt des Akkumulators wird mit dem im Operandenbereich angegebenen Operanden auf Gleichheit geprüft. Wenn beide Werte ungleich sind, wird als Ergebnis im Akkumulator "TRUE" abgelegt, andernfalls "FALSE".

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von NE (s. S. 1207)

- **Die Eingänge können jeden Datentyp aufweisen; alle Eingangsvariablen müssen jedoch vom selben Typ sein. Der Ausgang muss vom Typ BOOL sein.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Datentypen

Datentyp	E/A	Funktion
alle Datentypen	Eingang	Wert für Vergleich
alle Datentypen	Eingang	Referenzwert
BOOL	Ausgabe	Ergebnis, TRUE wenn Wert für Vergleich größer oder kleiner dem Referenzwert

Variablen, die miteinander verglichen werden, müssen denselben Datentyp aufweisen.

Bei mehreren Eingängen wird der erste mit dem zweiten verglichen, der zweite mit dem dritten etc. Ist der erste Wert ungleich dem zweiten Wert UND der zweite Wert ungleich dem dritten Wert, wird TRUE als Ergebnis geschrieben, andernfalls FALSE.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

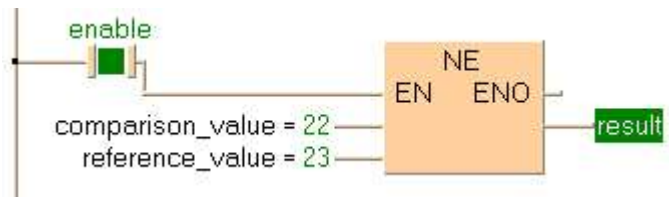
	Klasse	Bezeichner	Typ	Initial
0	VAR	enable	BOOL	FALSE
1	VAR	comparison_value	INT	0
2	VAR	reference_value	INT	0
3	VAR	result	BOOL	FALSE

In diesem Beispiel wurden die Eingangsvariablen (**comparison_value, reference_value** und **enable**) deklariert. Stattdessen können Sie Konstanten direkt an die Eingänge der Funktion schreiben (z.B. für Tests).

Rumpf

Wenn **enable** gesetzt ist (TRUE), wird **comparison_value** mit der Variablen **reference_value** verglichen. Wenn die beiden Werte ungleich sind, wird in **result** der Wert TRUE, sonst FALSE geschrieben.

KOP



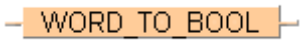
Kapitel 7

Umwandlungsfunktionen

WORD_TO_BOOL

WORD in BOOL

Erklärung WORD_TO_BOOL wandelt einen Wert vom Datentyp WORD in einen Wert vom Datentyp BOOL um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_TO_BOOL (s. S. 1199)**

☞ **Falls der Wert von WORD_value 0 (16#0000) ist, ist das Umwandlungsergebnis = 0 (FALSE), in jedem anderen Fall = 1 (TRUE).**

Datentypen

Datentyp	E/A	Funktion
WORD	Eingang	Eingangsdatentyp
BOOL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

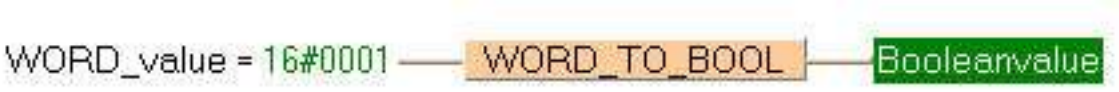
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Boolean_value	BOOL	FALSE
1	VAR	WORD_value	WORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **WORD_value** vom Datentyp WORD (16 bit) wird in einen booleschen Wert (1 Bit) umgewandelt. Das Ergebnis wird in **Boolean_value** geschrieben.

KOP



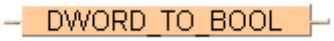
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Boolean_value:=WORD_TO_BOOL(WORD_value);
```

DWORD_TO_BOOL

DOUBLE WORD in BOOL

Erklärung DWORD_TO_BOOL wandelt einen Wert vom Datentyp DOUBLE WORD in einen Wert vom Datentyp BOOL um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_TO_BOOL (s. S. 1185)**

☞ **Falls die Variable DWORD_value den Wert 0 (16#00000000) hat, ist das Ergebnis der Umwandlung = FALSE, in jedem anderen Fall TRUE.**

Datentypen

Datentyp	E/A	Funktion
DWORD	Eingang	Eingangsdatentyp
BOOL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DWORD_value	DWORD	0
1	VAR	Boolean_value	BOOL	FALSE

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **DWORD_value** vom Datentyp DOUBLE WORD wird in einen booleschen Wert (1 Bit) umgewandelt. Der umgewandelte Wert wird in **Boolean_value** geschrieben.

KOP

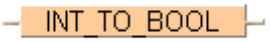
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Boolean_value:=DWORD_TO_BOOL(DWORD_value);
```

INT_TO_BOOL

INTEGER in BOOL

Erklärung INT_TO_BOOL wandelt einen Wert vom Datentyp INTEGER in einen Wert vom Datentyp BOOL um.

Symbol: 

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INT_TO_BOOL (s. S. 1193)**

Datentypen

Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
BOOL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Boolean_value	BOOL	FALSE
1	VAR	INT_value	INT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **INT_value** (16 Bit) vom Datentyp INTEGER wird in einen boolschen Wert umgewandelt. Das Ergebnis wird in **Boolean_value** geschrieben.

KOP

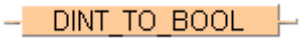
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Boolean_value:=INT_TO_BOOL(INT_value);
```

DINT_TO_BOOL

DOUBLE INTEGER in BOOL

Erklärung DINT_TO_BOOL wandelt einen Wert vom Datentyp DOUBLE INTEGER in einen Wert vom Datentyp BOOL um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DINT_TO_BOOL (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
BOOL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DINT_value	DINT	0
1	VAR	Boolean_value	BOOL	FALSE

In diesem Beispiel wurde die Eingangsvariable (**DINT_value**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **DINT_value** vom Datentyp DOUBLE INTEGER wird in einen Wert vom Datentyp BOOL umgewandelt. Das Ergebnis wird in **Boolean_value** geschrieben.

KOP

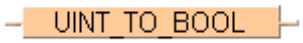
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Boolean_value:=DINT_TO_BOOL(DINT_value);
```

UINT_TO_BOOL

vorzeichenloser INTEGER in BOOL

Erklärung UINT_TO_BOOL wandelt einen Wert vom Datentyp Unsigned INTEGER in einen Wert vom Datentyp BOOL um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UINT_TO_BOOL (s. S. 1198)**

☞ **Bei Eingangswert 0 (16#0000) ist das Umwandlungsergebnis 0 (FALSE), in jedem anderen Fall 1 (TRUE).**

Datentypen

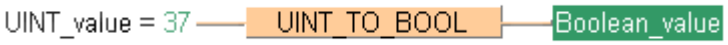
Datentyp	E/A	Funktion
UINT	Eingang	Eingangsdatentyp
BOOL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	37
1	VAR	Boolean_value	BOOL	FALSE

KOP



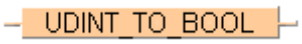
ST

```
Boolean_value:= UINT_TO_BOOL(UINT_value);
```

UDINT_TO_BOOL

vorzeichenloser DOUBLE INTEGER in BOOL

Erklärung UDINT_TO_BOOL wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen Wert vom Datentyp BOOL um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_BOOL (s. S. 1198)**

☞ **Bei Eingangswert 0 (16#0000) ist das Umwandlungsergebnis 0 (FALSE), in jedem anderen Fall 1 (TRUE).**

Datentypen

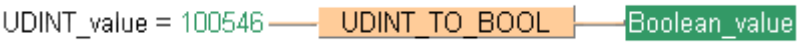
Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
BOOL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	100546
1	VAR	Boolean_value	BOOL	FALSE

KOP



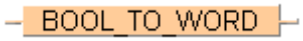
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Boolean_value := UDINT_TO_BOOL (UDINT_value);
```

BOOL_TO_WORD

BOOL in WORD

Erklärung BOOL_TO_WORD wandelt einen Wert vom Datentyp BOOL in einen Wert vom Datentyp WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von BOOL_TO_WORD (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
BOOL	Eingang	Eingangsdatentyp
WORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

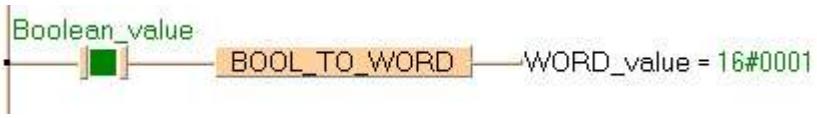
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Boolean_value	BOOL	FALSE
1	VAR	WORD_value	WORD	0

In diesem Beispiel wurde die Eingangsvariable (**Boolean_value**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Boolean_value** vom Datentyp BOOL wird in einen Wert vom Datentyp WORD umgewandelt. Der umgewandelte Wert wird in **WORD_value** geschrieben.

KOP



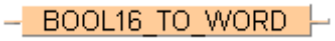
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF Boolean_value THEN
    WORD_value:=BOOL_TO_WORD(Boolean_value);
END_IF;
```

BOOL16_TO_WORD

BOOL16 in WORD

Erklärung Diese Funktion kopiert Daten eines Arrays mit 16 Elementen vom Datentyp BOOL am Eingang (BOOL16 (s. S. 54)) auf eine Variable vom Typ WORD am Ausgang.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von BOOL16_TO_WORD (s. S. 1184)**

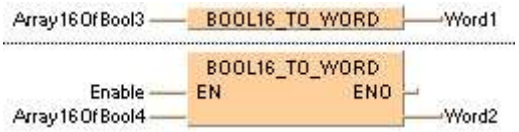
Datentypen

Datentyp	Kommentar
ARRAY of BOOL	ARRAY mit 16 Elementen
WORD	Ausgangsvariable

POE-Kopf:

	Klasse	Bezeichner	Typ	Initial
0	VAR	Array16OfBool1	ARRAY [0..15] OF BOOL	[16(FALSE)]
1	VAR	Array16OfBool2	ARRAY [0..15] OF BOOL	[16(FALSE)]
2	VAR	Enable	BOOL	FALSE
3	VAR	Word_1	WORD	0
4	VAR	Word_2	WORD	0

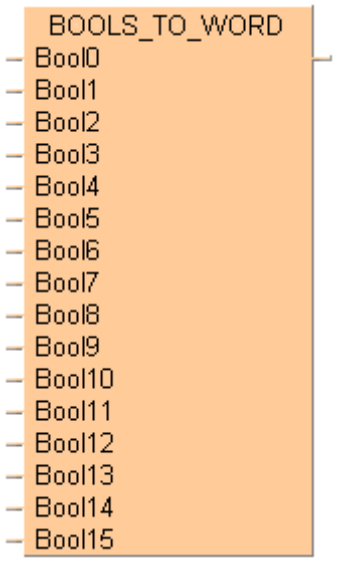
Beispiel:



BOOLS_TO_WORD

16 Variablen vom Typ BOOL in WORD

Erklärung Diese Funktion wandelt 16 Werte vom Datentyp BOOL bitweise in einen Wert vom Datentyp WORD um.



Die Eingänge Bool0 bis Bool15 müssen im KOP und im FBD nicht verbunden werden bzw. im ST-Editor in der formalen Parameterliste nicht explizit verwendet werden. Diese nicht verbundenen bzw. benutzten Eingänge werden als FALSE angenommen. Für diese Eingänge sowie für Eingänge, die mit den Konstanten TRUE oder FALSE belegt sind, wird kein Programmcode generiert. Nur für die mit Variablen belegten Eingänge wird auch Programmcode generiert.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von BOOLS_TO_WORD (s. S. 1184)**

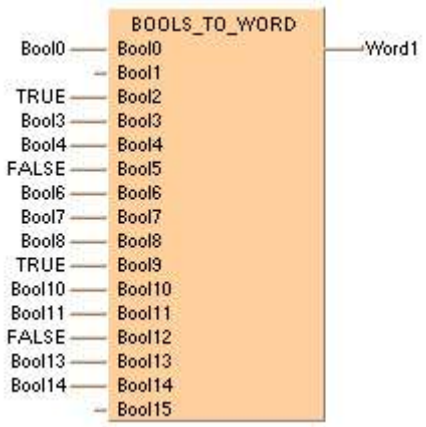
Datentypen

Variable	Datentyp	Funktion
BOOL0 ... BOOL15	BOOL	16 Eingangsvariablen vom Datentyp BOOL
	WORD	Ausgangsvariable

POE-Kopf:

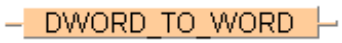
	Klasse	Bezeichner	Typ	Initial
0	VAR	Word0	WORD	0
1	VAR	Bool0	BOOL	FALSE
2	VAR	Bool1	BOOL	FALSE
3	VAR	Bool2	BOOL	FALSE
4	VAR	Bool3	BOOL	FALSE
5	VAR	Bool4	BOOL	FALSE
6	VAR	Bool5	BOOL	FALSE
7	VAR	Bool6	BOOL	FALSE
8	VAR	Bool7	BOOL	FALSE
9	VAR	Bool8	BOOL	FALSE
10	VAR	Bool9	BOOL	FALSE
11	VAR	Bool10	BOOL	FALSE
12	VAR	Bool11	BOOL	FALSE
13	VAR	Bool12	BOOL	FALSE
14	VAR	Bool13	BOOL	FALSE
15	VAR	Bool14	BOOL	FALSE
16	VAR	Bool15	BOOL	FALSE

Beispiel:



DWORD_TO_WORD DOUBLE WORD in WORD

Erklärung DWORD_TO_WORD wandelt einen Wert vom Datentyp DOUBLE WORD in einen Wert vom Datentyp WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_TO_WORD (s. S. 1185)**

☞ **Die ersten 16 Bit der Eingangsvariablen werden der Ausgangsvariablen zugewiesen.**

Datentypen

Datentyp	E/A	Funktion
DWORD	Eingang	Eingangsdatentyp
WORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DWORD_value	DWORD	0
1	VAR	WORD_value	WORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Wenn Freigabe gesetzt ist (TRUE), wird **DWORD_value** vom Datentyp DOUBLE WORD (32 Bit) in einen Wert vom Datentyp WORD (16 Bit) umgewandelt. Der umgewandelte Wert wird in **WORD_value** geschrieben.

KOP

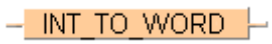
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
WORD_value:=DWORD_TO_WORD(DWORD_value);
```

INT_TO_WORD

INTEGER in WORD

Erklärung INT_TO_WORD wandelt einen Wert vom Datentyp INTEGER in einen Wert vom Datentyp WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INT_TO_WORD (s. S. 1193)**

☞ **Die Bitkombination der Eingangsvariablen wird der Ausgangsvariablen zugewiesen.**

Datentypen

Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
WORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	WORD_value	WORD	0
1	VAR	INT_value	INT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **INT_value** vom Datentyp INTEGER wird in einen Wert vom Datentyp WORD umgewandelt. Das Ergebnis wird in **WORD_value** geschrieben.

KOP

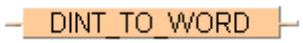
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
WORD_value:=INT_TO_WORD(INT_value);
```

DINT_TO_WORD

DOUBLE INTEGER in WORD

Erklärung DINT_TO_WORD wandelt einen Wert vom Datentyp DOUBLE INTEGER in einen Wert vom Datentyp WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DINT_TO_WORD (s. S. 1185)**

☞ **Die ersten 16 Bit der Eingangsvariablen werden der Ausgangsvariablen zugewiesen.**

Datentypen

Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
WORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DINT_value	DINT	0
1	VAR	WORD_value	WORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **DINT_value** vom Datentyp DOUBLE WORD (32 Bit) wird in einen Wert vom Datentyp WORD (16 Bit) umgewandelt. Der umgewandelte Wert wird in **WORD_value** geschrieben.

KOP

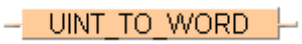
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
WORD_value:=DINT_TO_WORD(DINT_value);
```

UINT_TO_WORD

vorzeichenloser INTEGER in WORD

Erklärung UINT_TO_WORD wandelt einen Wert vom Datentyp Unsigned INTEGER in einen Wert vom Datentyp WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UINT_TO_WORD (s. S. 1198)**

☞ **Die ersten 16 Bit der Eingangsvariable werden der Ausgangsvariable zugewiesen.**

Datentypen

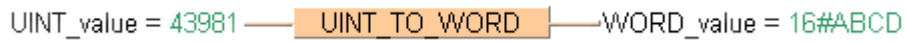
Datentyp	E/A	Funktion
UINT	Eingang	Eingangsdatentyp
WORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	43981
1	VAR	WORD_value	WORD	16#0000

KOP



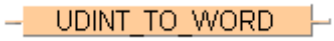
ST

```
WORD_value := UINT_TO_WORD (UINT_value);
```

UDINT_TO_WORD

vorzeichenloser DOUBLE INTEGER in WORD

Erklärung UDINT_TO_WORD wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen Wert vom Datentyp WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_WORD (s. S. 1198)**

☞ **Die ersten 16 Bit der Eingangsvariable werden der Ausgangsvariable zugewiesen.**

Datentypen

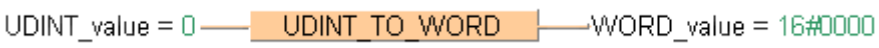
Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
WORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	65535
1	VAR	WORD_value	WORD	0

KOP



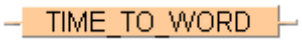
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
WORD_value := UDINT_TO_WORD (UDINT_value);
```

TIME_TO_WORD

TIME in WORD

Erklärung TIME_TO_WORD wandelt einen Wert vom Datentyp TIME in einen Wert vom Datentyp WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TIME_TO_WORD (s. S. 1197)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang	Eingangsdatentyp
WORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

Beispiele:

Eingangsvariable	Ausgangsvariable
T#123.4s	1234
T#1.00s	16#0064

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	TIME_value	TIME	T#0s
1	VAR	WORD_value	WORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **Time_value** vom Datentyp TIME wird in einen Wert vom Datentyp WORD umgewandelt. Das Ergebnis wird in die Ausgangsvariable **WORD_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
WORD_value:=TIME_TO_WORD(time_value);
```

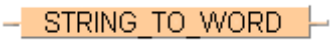
STRING_TO_WORD

STRING (Hexadezimal-Format) in WORD

Erklärung

Diese Funktion wandelt eine Zeichenkette im Hexadezimal-Format in einen Wert vom Datentyp WORD um.

Symbol:

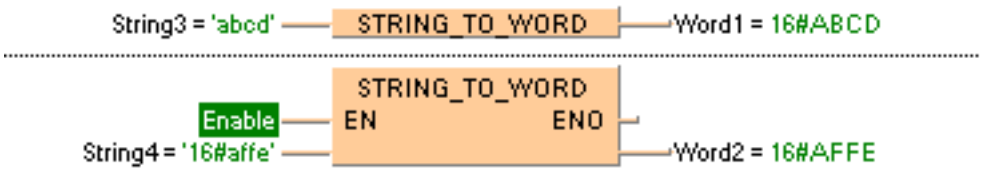


Dabei wird die anliegende Zeichenkette zunächst in einen Wert vom Datentyp STRING[32] gewandelt. Anschließend wird dieser in einem Unterprogramm mit ca. 270 Schritten, welches gemeinsam für die Funktionen STRING_TO_INT, STRING_TO_WORD, STRING_TO_DINT oder STRING_TO_DWORD verwendet wird, in einen Wert vom Datentyp WORD gewandelt.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Siehe auch: STRING_TO_WORD_STEPSAVER

Beispiel:



Zulässiges Format:

'[Leerzeichen][Hexadezimalzahlen][Leerzeichen]' e.g. ' afFE '

Zulässige Zeichen:

Leerzeichen	Alle Zeichen außer "+“ (Plus), "-" (Minus) und allen Hexadezimalzahlen
Hexadezimalzahlen	Hexadezimalzahlen aus den Bereichen "0 - 9“, "A - F“ oder "a - f“.

Die Analyse endet mit der ersten Nicht-Hexadezimalzahl.

SPS-Typen **Verfügbarkeit von STRING_TO_WORD (s. S. 1197)**

Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
WORD	Ausgangsvariable

STRING_TO_WORD _STEPSAVER

STRING (Hexadezimal-Format rechtsbündig) in WORD

Erklärung Diese Funktion wandelt die Zeichenkette, die rechtsbündig im Hexadezimalformat anliegt, mit der maximal möglichen Anzahl von Zeichen in einen Wert vom Datentyp WORD um.

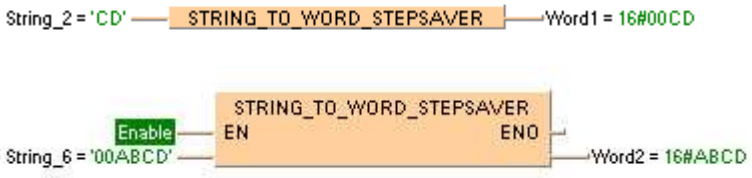
Beispiele:

Eingang	definiert als	liefert
'D'	STRING[1]	16#D
'CD'	STRING[2]	16#CD
'BCD'	STRING[3]	16#BCD
'ABCD'	STRING[4]	16#ABCD
'0ABCD'	STRING[5]	16#ABCD
'00ABCD'	STRING[6]	16#ABCD

Hierbei wird direkt der Befehl F72_A2HEX (s. S. 604) mit nur ca. 7 Schritten verwendet. Die Steuerung liefert insbesondere dann einen Operationsfehler, wenn ein Buchstabe erscheint, der keiner Hexadezimalzahl "0 - F" oder "A-F" entspricht.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiel:



Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
WORD	Ausgangsvariable

Zulässiges Format für STRING[4]:

'Hex1Hex2Hex3Hex4' also z.B. 'AFFE'

Zulässige Zeichen:

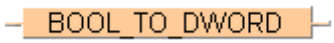
Hex1 bis Hex4	Hexadezimalzahlen aus den Bereichen "0 - 9“ oder "A - F“ (nicht "a - f").

SPS-Typen **Verfügbarkeit von STRING_TO_WORD STEPSAVER (s. S. 1197)**

BOOL_TO_DWORD

BOOL in DOUBLE WORD

Erklärung BOOL_TO_DWORD wandelt einen Wert vom Datentyp BOOL in einen Wert vom Datentyp DOUBLE WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von BOOL_TO_DWORD (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
BOOL	Eingang	Eingangsdatentyp
DWORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

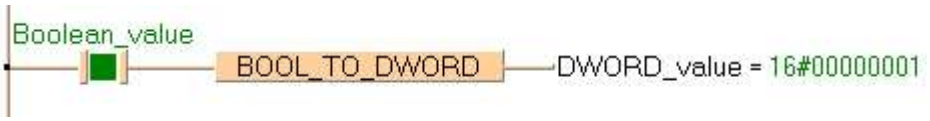
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Boolean_value	BOOL	FALSE
1	VAR	WORD_value	WORD	0

In diesem Beispiel wurde die Eingangsvariable (**Boolean_value**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Boolean_value** vom Datentyp BOOL wird in einen Wert vom Datentyp DOUBLE INTEGER umgewandelt. Der umgewandelte Wert wird in **DWORD_value** geschrieben.

KOP



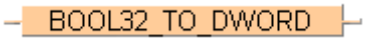
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF Boolean_value THEN
    DWORD_value:=BOOL_TO_DWORD(Boolean_value);
END_IF;
```

BOOL32_TO_DWORD

BOOL32 in DOUBLE WORD

Erklärung Diese Funktion **kopiert Daten eines Arrays mit 32 Elementen vom Datentyp BOOL am Eingang (BOOL32 (s. S. 55)) auf eine Variable vom** Typ WORD am Ausgang.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von BOOL32_TO_DWORD (s. S. 1184)**

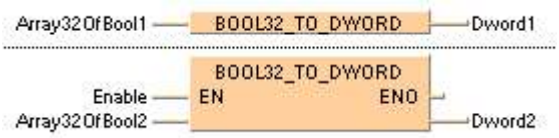
Datentypen

Datentyp	Kommentar
ARRAY of BOOL	ARRAY mit 32 Elementen
DWORD	Ausgangsvariable

POE-Kopf:

	Klasse	Bezeichner	Typ	Initial
0	VAR	Enable	BOOL	FALSE
1	VAR	Array32OfBool1	ARRAY [0..31 OF BOOL	[FALSE]
2	VAR	Array32OfBool2	ARRAY [0..31 OF BOOL	
3	VAR	DWord1	DWORD	0
4	VAR	DWord2	DWORD	0

Beispiel:

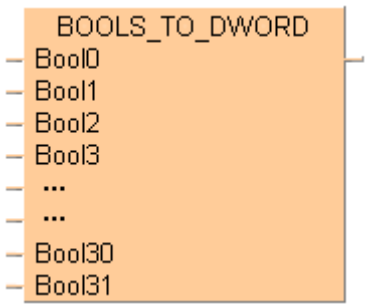


BOOLS_TO_DWORD

32 Variablen vom Typ BOOL in DWORD

Erklärung Diese Funktion wandelt 32 Werte vom Datentyp BOOL bitweise in einen Wert vom Datentyp DWORD um.

Die Eingänge Bool0 bis Bool31 müssen im KOP und im FBD nicht verbunden werden bzw. im ST-Editor in der formalen Parameterliste nicht explizit verwendet werden. Diese nicht verbundenen bzw. benutzten Eingänge werden als FALSE angenommen. Für diese Eingänge sowie für Eingänge, die mit den Konstanten TRUE oder FALSE belegt sind, wird kein Programmcode generiert. Nur für die mit Variablen belegten Eingänge wird auch Programmcode generiert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von BOOL_TO_DWORD (s. S. 1184)**

Datentypen

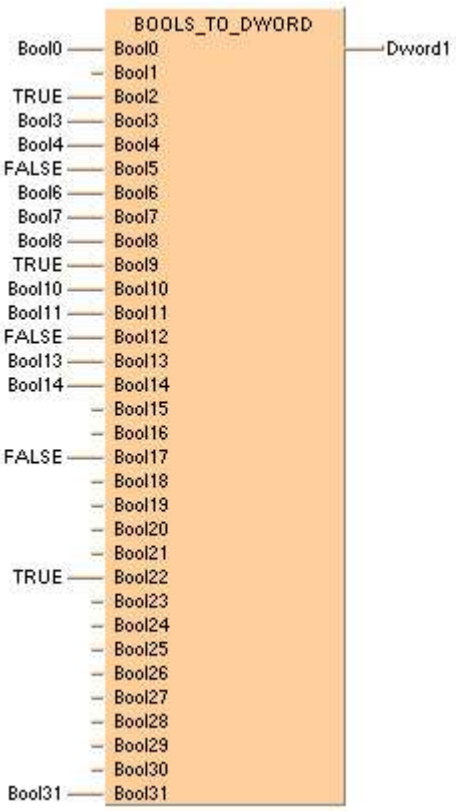
Variable	Datentyp	Funktion
BOOL0 ... BOOL31	BOOL	32 Eingangsvariablen vom Datentyp BOOL
	DWORD	Ausgangsvariable

POE-Kopf:

	Klasse	Bezeichner	Typ	Initial
0	VAR	dWord1	DWORD	0
1	VAR	Bool0	BOOL	FALSE
2	VAR	Bool1	BOOL	FALSE
3	VAR	Bool2	BOOL	FALSE
4	VAR	Bool3	BOOL	FALSE
5	VAR	Bool4	BOOL	FALSE
6	VAR	Bool5	BOOL	FALSE
7	VAR	Bool6	BOOL	FALSE
8	VAR	Bool7	BOOL	FALSE

usw. bis Bool31

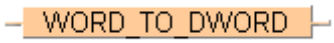
Beispiel:


BOOLS_TO_DWORD
Bool0
Bool1
Bool2
Bool3
Bool4
Bool5
Bool6
Bool7
Bool8
Bool9
Bool10
Bool11
Bool12
Bool13
Bool14
Bool15
Bool16
Bool17
Bool18
Bool19
Bool20
Bool21
Bool22
Bool23
Bool24
Bool25
Bool26
Bool27
Bool28
Bool29
Bool30
Bool31
Dword1
TRUE
TRUE
TRUE
FALSE
FALSE
FALSE

WORD_TO_DWORD

WORD in DOUBLE WORD

Erklärung WORD_TO_DWORD wandelt einen Wert vom Datentyp WORD in einen Wert vom Datentyp DOUBLE WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_TO_DWORD (s. S. 1199)**

Datentypen

Datentyp	E/A	Funktion
WORD	Eingang	Eingangsdatentyp
DWORD	Ausgang	Umwandlungsergebnis

☞ **Die Bitkombination von WORD_value wird DWORD_value zugewiesen.**

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	WORD_value	WORD	0
1	VAR	DWORD_value	DWORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **WORD_value** vom Datentyp WORD wird in einen Wert vom Datentyp DOUBLE WORD umgewandelt. Das Ergebnis wird in **DWORD_value** geschrieben.

KOP

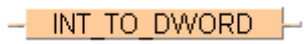
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DWORD_value:=WORD_TO_DWORD(WORD_value);
```

INT_TO_DWORD

INTEGER in DOUBLE WORD

Erklärung INT_TO_DWORD wandelt einen Wert vom Datentyp INTEGER in einen Wert vom Datentyp DOUBLE WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INT_TO_DWORD (s. S. 1193)**

Datentypen

Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
DWORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	INT_value	INT	0
1	VAR	DWORD_value	DWORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **INT_value** vom Datentyp INTEGER wird in einen Wert vom Datentyp DOUBLE WORD (32 Bit) umgewandelt. Das Ergebnis wird in **DWORD_value** geschrieben.

KOP

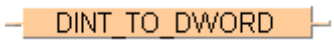
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DWORD_value:=INT_TO_DWORD(INT_value);
```

DINT_TO_DWORD

DOUBLE INTEGER in DOUBLE WORD

Erklärung DINT_TO_DWORD wandelt einen Wert vom Datentyp DOUBLE INTEGER in einen Wert vom Datentyp DWORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DINT_TO_DWORD (s. S. 1185)**

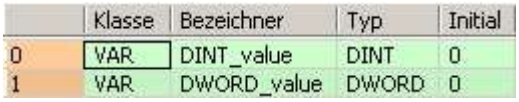
☞ **Die Bitkombination der Eingangsvariablen wird der Ausgangsvariablen zugewiesen.**

Datentypen

Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
DWORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



	Klasse	Bezeichner	Typ	Initial
0	VAR	DINT_value	DINT	0
1	VAR	DWORD_value	DWORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **DINT_value** vom Datentyp DOUBLE INTEGER wird in einen Wert vom Datentyp DOUBLE WORD umgewandelt. Der umgewandelte Wert wird in **DWORD_value** geschrieben.

KOP

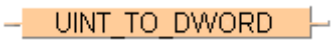
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DWORD_value:=DINT_TO_DWORD(DINT_value);
```

UINT_TO_DWORD

vorzeichenloser INTEGER in DOUBLE WORD

Erklärung UINT_TO_DWORD wandelt einen Wert vom Datentyp Unsigned INTEGER in einen Wert vom Datentyp DOUBLE WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UINT_TO_DWORD (s. S. 1198)**

Datentypen

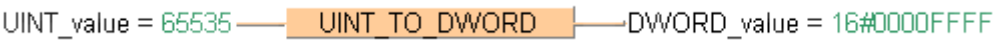
Datentyp	E/A	Funktion
UINT	Eingang	Eingangsdatentyp
DWORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	65535
1	VAR	DWORD_value	DWORD	16#00000000

KOP



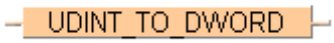
ST

```
DWORD_value := UINT_TO_DWORD(UINT_value);
```

UDINT_TO_DWORD

vorzeichenloser DOUBLE INTEGER in DOUBLE WORD

Erklärung UDINT_TO_DWORD wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen Wert vom Datentyp DOUBLE WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_DWORD (s. S. 1198)**

Datentypen

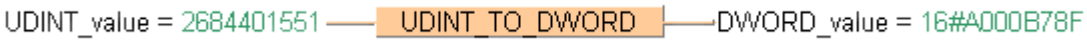
Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
DWORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	16#190854
1	VAR	DWORD_value	DWORD	16#00000000

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

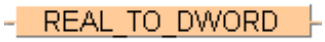
```
DWORD_value := UDINT_TO_DWORD(UDINT_value);
```

REAL_TO_DWORD

REAL in DOUBLE WORD

Erklärung

REAL_TO_DWORD verschiebt die Bitstring-Informationen einer REAL-Variable in eine DWORD-Variable. Mit DWORD_OVERLAPPING_DUT kann die gleiche Funktionalität erzielt werden.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von REAL_TO_DWORD (s. S. 1195)

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
DWORD	Ausgang	Umwandlungsergebnis

Beispiel

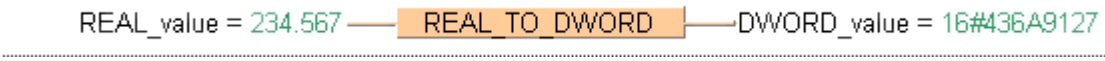
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	REAL_value	REAL	234.567
1	VAR	DWORD_value	DWORD	16#00000000

KOP



ST

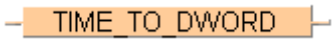
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DWORD_value := REAL_TO_DWORD(REAL_value);
```

TIME_TO_DWORD

TIME in DOUBLE WORD

Erklärung TIME_TO_DWORD wandelt einen Wert vom Datentyp TIME in einen Wert vom Datentyp DWORD um. Die Zeit 10ms entspricht dem Wert 1, z.B. wird der Eingangswert T#1s in den Wert 100 (16#64) gewandelt.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TIME_TO_DWORD (s. S. 1197)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang	Eingangsdatentyp
DWORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	TIME_value	TIME	T#0ms
1	VAR	DWORD_value	DWORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **Time_value** vom Datentyp TIME wird in einen Wert vom Datentyp DWORD umgewandelt und in die Ausgangsvariable **DWORD_value** geschrieben.

KOP

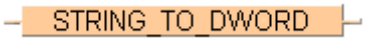
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DWORD_value:=TIME_TO_DWORD(time_value);
```

STRING_TO_DWORD

STRING (Hexadezimal-Format) in DOUBLE WORD

Erklärung Diese Funktion wandelt eine Zeichenkette im Hexadezimal-Format in einen Wert vom Datentyp DWORD um.

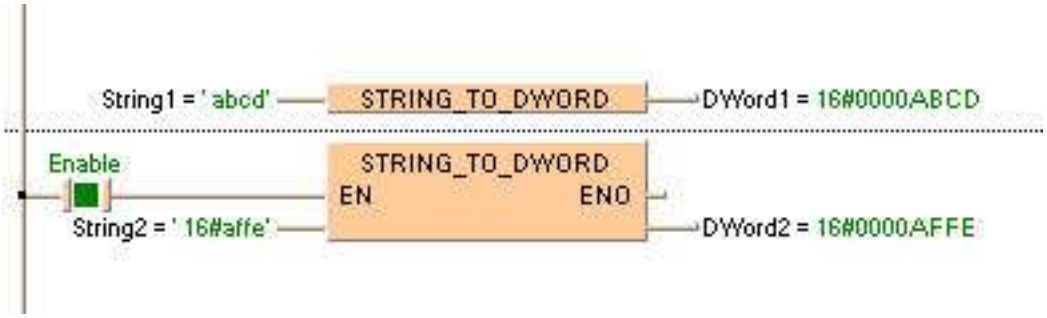
Symbol: 

Dabei wird die anliegende Zeichenkette zunächst in einen Wert vom Datentyp STRING[32] gewandelt. Anschließend wird dieser in einem Unterprogramm mit ca. 270 Schritten, welches gemeinsam für die Funktionen STRING_TO_INT, STRING_TO_WORD, STRING_TO_DINT oder STRING_TO_DWORD verwendet wird, in einen Wert vom Datentyp DWORD gewandelt.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Siehe auch: STRING_TO_DWORD_STEPSAVER

Beispiel:



Zulässiges Format:

'[Leerzeichen][Hexadezimalzahlen][Leerzeichen]' e.g. ' afFE '

Zulässige Zeichen:

Leerzeichen	Leerzeichen " "
Vorzeichen	Pluszeichen "+" und Minuszeichen "-"
Hexadezimalzahlen	Hexadezimalzahlen aus dem Bereichen von "0 - 9", "A -F" oder "a - f".

Die Analyse endet mit der ersten Nicht-Dezimalzahl.

SPS-Typen **Verfügbarkeit von STRING_TO_DWORD (s. S. 1197)**

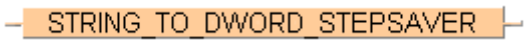
Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
DWORD	Ausgangsvariable

STRING_TO_DWORD _STEPSAVER

STRING (Hexadezimal-Format rechtsbündig) in DOUBLE WORD

Erklärung Diese Funktion wandelt die maximal mögliche Zeichenkettenlänge (STRING[8]), wie sie im POE-Kopf festgelegt wurde, in einen Wert vom Datentyp DWORD um.



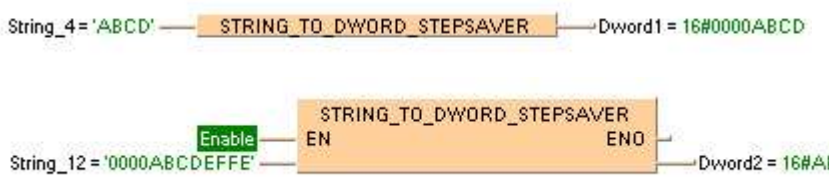
Beispiele:

Eingang	definiert als	liefert
'FE'	STRING[2]	16#FE
'EFFE'	STRING[4]	16#EFFE
'CDEFFE'	STRING[6]	16#CDEFFE
'ABCDEFFE'	STRING[8]	16#ABCDEFFE
'00ABCDEFFE'	STRING[10]	16#ABCDEFFE

Hierbei wird direkt der Befehl F72_A2HEX (s. S. 604) mit nur ca. 7 Schritten verwendet. Die Steuerung liefert insbesondere dann einen Operationsfehler, wenn ein Buchstabe erscheint, der keiner Hexadezimalzahl "0 - F" oder "A-F" entspricht.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiel:



Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
DWORD	Ausgangsvariable

Zulässiges Format für STRING[8]:

'Hex1Hex2Hex3Hex4Hex5Hex6Hex7Hex8' also z.B. '001AAFFE'

Zulässige Zeichen:

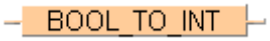
Hex1 bis Hex8	Hexadezimalzahlen aus den Bereichen "0 - 9“ oder "A - F“ (nicht "a - f").

SPS-Typen **Verfügbarkeit von STRING_TO_DWORD STEPSAVER (s. S. 1197)**

BOOL_TO_INT

BOOL in INTEGER

Erklärung BOOL_TO_INT wandelt einen Wert vom Datentyp BOOL in einen Wert vom Datentyp INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von BOOL_TO_INT (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
BOOL	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

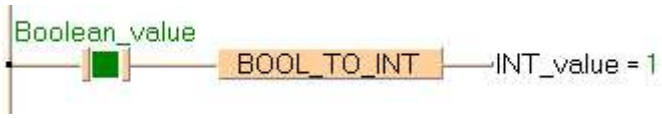
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Boolean_value	BOOL	FALSE
1	VAR	INT_value	INT	0

In diesem Beispiel wurde die Eingangsvariable (**Boolean_value**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Boolean_value** vom Datentyp BOOL wird in einen Wert vom Datentyp INTEGER umgewandelt. Der umgewandelte Wert wird in **INT_value** geschrieben.

KOP



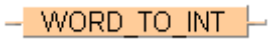
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF Boolean_value THEN
    INT_value:=BOOL_TO_INT(Boolean_value);
END_IF;
```

WORD_TO_INT

WORD in INTEGER

Erklärung WORD_TO_INT wandelt einen Wert vom Typ WORD in einen Wert vom Typ INT um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_TO_INT (s. S. 1199)**

☞ **Die Bitkombination von WORD_value wird INT_value zugewiesen.**

Datentypen

Datentyp	E/A	Funktion
WORD	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	WORD_value	WORD	0
1	VAR	INT_value	INT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **WORD_value** vom Datentyp WORD wird in einen Wert vom Datentyp INTEGER umgewandelt. Das Ergebnis wird in **INT_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
INT_value:=WORD_TO_INT(WORD_value);
```

BCD_TO_INT

BCD in INTEGER

Erklärung WORD_BCD_TO_INT wandelt einen BCD-Wert von WORD in Binärwerte vom Datentyp INTEGER um.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_BCD_TO_INT (s. S. 1199)**

Datentypen

Datentyp	E/A	Funktion
WORD_BCD	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	BCD_value_16bit	WORD	0
1	VAR	INT_value	INT	1

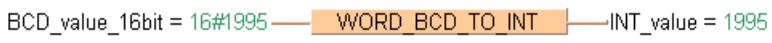
In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

BCD-Konstanten können wie folgt in Control FPWIN Pro dargestellt werden:

2#0001100110010101 oder
16#1995

Rumpf **BCD_value_16bit** vom Datentyp WORD wird in einen INTEGER-Wert umgewandelt. Der umgewandelte Wert wird in die Ausgangsvariable **INT_value** geschrieben.

KOP



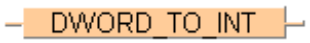
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
INT_value:=WORD_BCD_TO_INT(BCD_value_16bit);
```

DWORD_TO_INT

DOUBLE WORD in INTEGER

Erklärung DWORD_TO_INT wandelt einen Wert vom Datentyp DOUBLE WORD in einen Wert vom Datentyp INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_TO_INT (s. S. 1185)**

☞ **Die ersten 16 Bit der Eingangsvariablen werden der Ausgangsvariablen zugewiesen.**

Datentypen

Datentyp	E/A	Funktion
DWORD	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DWORD_value	DWORD	0
1	VAR	INT_value	INT	0

In diesem Beispiel wurde die Eingangsvariable (**DWORD_value**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **DWORD_value** vom Datentyp DOUBLE WORD (32 Bit) wird in einen Wert vom Datentyp INTEGER (16 Bit) umgewandelt. Der umgewandelte Wert wird in **INT_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
INT_value:=DWORD_TO_INT(DWORD_value);
```

DINT_TO_INT

DOUBLE INTEGER in INTEGER

Erklärung DINT_TO_INT wandelt einen Wert vom Datentyp DOUBLE INTEGER in einen Wert vom Datentyp INTEGER um.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DINT_TO_INT (s. S. 1185)**

☞ **Der Wert der Eingangsvariablen sollte zwischen -32768 und 32767 liegen, um durch die Wandlung nicht verfälscht zu werden.**

Datentypen

Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DINT_value	DINT	0
1	VAR	INT_value	INT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **DINT_value** vom Datentyp DOUBLE INTEGER (32 Bit) wird in einen Wert vom Datentyp INTEGER (16 Bit) umgewandelt. Der umgewandelte Wert wird in **INT_value** geschrieben.

KOP

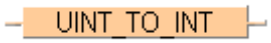
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
INT_value:=DINT_TO_INT(DINT_value);
```

UINT_TO_INT

vorzeichenloser INTEGER in INTEGER

Erklärung UINT_TO_INT wandelt einen Wert vom Datentyp Unsigned INTEGER in einen Wert vom Datentyp INT um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UINT_TO_INT (s. S. 1198)**

Datentypen

Datentyp	E/A	Funktion
UINT	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	16383
1	VAR	INT_value	INT	0

KOP

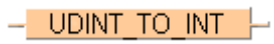
ST

```
INT_value := UINT_TO_INT(UINT_value);
```

UDINT_TO_INT

vorzeichenloser DOUBLE INTEGER in INTEGER

Erklärung UDINT_TO_INT wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen Wert vom Datentyp INT um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_INT (s. S. 1198)**

Datentypen

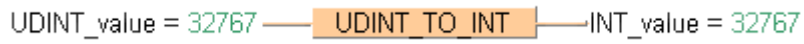
Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	32767
1	VAR	INT_value	INT	0

KOP



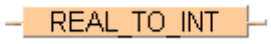
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
INT_value := UDINT_TO_INT (UDINT_value);
```

REAL_TO_INT

REAL in INTEGER

Erklärung REAL_TO_INT wandelt einen Wert vom Datentyp REAL in einen Wert vom Datentyp INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von REAL_TO_INT (s. S. 1196)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DATE_value1	DATE	D#2010-07-11
1	VAR	DATE_value2	DATE	D#2010-07-03
2	VAR	TIME_result	TIME	T#0s

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **REAL_value** vom Datentyp REAL wird in einen Wert vom Datentyp INTEGER umgewandelt. Der umgewandelte Wert wird in **INT_value** gespeichert.

KOP

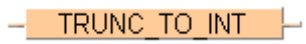
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
INT_value := REAL_TO_INT(REAL_value);
```

TRUNC_TO_INT

TRUNC in INTEGER

Erklärung TRUNC_TO_INT schneidet die Nachkommastellen einer Zahl vom Datentyp REAL ab und liefert eine Ausgangsvariable vom Datentyp INTEGER.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TRUNC_TO_INT (s. S. 1198)**

☞

- **Durch das Abschneiden der Nachkommastellen wird eine positive Zahl in Richtung 0 verkleinert und eine negative Zahl in Richtung 0 vergrößert.**
- **Die ersten 16 Bit der Eingangsvariablen werden der Ausgangsvariablen zugewiesen.**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Eingangsvariable nicht den Datentyp REAL hat
R9008	%MX0.900.8	kurzzeitig	▪ Ausgangsvariable größer als ein 16-Bit INTEGER ist
R9009	%MX0.900.9	kurzzeitig	▪ die Ausgangsvariable Null ist

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	REAL_value	REAL	0.0	number betw. -32768.99 ... +32767
1	VAR	INT_value	INT	0	number betw. -32768 ... +32768

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Von **REAL_value** werden die Nachkommastellen abgeschnitten. Das Ergebnis wird als 16-Bit INTEGER in **INT_value** geschrieben.

KOP

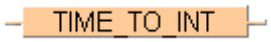
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
INT_value:=TRUNC_TO_INT(REAL_value);
```

TIME_TO_INT

TIME in INTEGER

Erklärung TIME_TO_INT wandelt einen Wert vom Datentyp TIME in einen Wert vom Datentyp INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TIME_TO_INT (s. S. 1197)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	TIME_value	TIME	T#0s
1	VAR	INT_value	INT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **Time_value** vom Datentyp TIME wird in einen Wert vom Datentyp INTEGER umgewandelt. Das Ergebnis wird in die Ausgangsvariable **INT_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
INT_value:=TIME_TO_INT(time_value);
```

STRING_TO_INT

STRING (Dezimalformat) in INTEGER

Erklärung

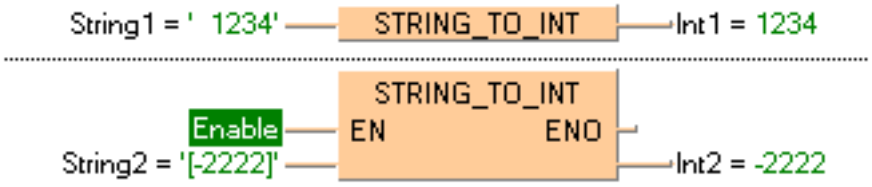
Diese Funktion wandelt eine Zeichenkette im Dezimalformat (STRING) in einen Wert vom Datentyp INT um.

Symbol: 

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Dabei wird die anliegende Zeichenkette zunächst in einen Wert vom Datentyp STRING[32] gewandelt. Anschließend wird dieser in einem Unterprogramm mit ca. 270 Schritten, welches gemeinsam für die Funktionen STRING_TO_INT, STRING_TO_WORD, STRING_TO_DINT oder STRING_TO_DWORD verwendet wird, in einen Wert vom Datentyp INT gewandelt.

Beispiel:



Zulässiges Format:

´[Leerzeichen][Vorzeichen][Dezimalzahlen][Leerzeichen]´ z.B. ´ 123456 ´

Zulässige Zeichen:

Leerzeichen	Alle Zeichen außer "+“ (Plus), "-" (Minus) und allen Dezimalzahlen
Vorzeichen	"+“ (Plus), "-" (Minus)
Dezimalzahlen	Dezimalzahlen von "0 - 9“

Die Analyse endet mit der ersten Nicht-Dezimalzahl.

SPS-Typen **Verfügbarkeit von STRING_TO_INT (s. S. 1197)**

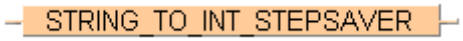
Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
INT	Ausgangsvariable

STRING_TO_INT_ STEPSAVER

STRING (Dezimal-Format rechtsbündig) in INTEGER

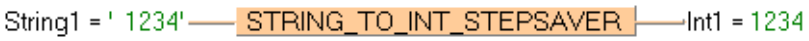
Erklärung Diese Funktion wandelt eine rechtsbündige Dezimalzahl in einer Zeichenkette in einen Wert vom Datentyp INT um.



Hierbei wird direkt der Befehl F76_A2BIN (s. S. 616) mit nur ca. 7 Schritten verwendet. Die Steuerung liefert insbesondere dann einen Operationsfehler, wenn ein Buchstabe erscheint, der keiner Dezimalzahl "0 - 9" und nicht "+" oder "-" oder führenden Leerzeichen entspricht.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiel:



Zulässiges Format:

´[Leerzeichen][Vorzeichen][Dezimalzahlen]´ z.B. ´ 123456´

Zulässige Zeichen:

Leerzeichen	Leerzeichen " "
Vorzeichen	Pluszeichen "+" und Minuszeichen "-"
Dezimalzahl	Dezimalzahlen von "0" - "9"

SPS-Typen **Verfügbarkeit von STRING_TO_INT STEPSAVER (s. S. 1197)**

Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
INT	Ausgangsvariable

BOOL_TO_UINT

BOOL in vorzeichenlosen INTEGER

Erklärung BOOL_TO_UINT wandelt einen Wert vom Datentyp BOOL in einen Wert vom Datentyp Unsigned INTEGER um.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von BOOL_TO_UINT (s. S. 1184)**

Datentypen

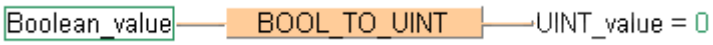
Datentyp	E/A	Funktion
BOOL	Eingang	Eingangsdatentyp
UINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	Boolean_value	BOOL	FALSE

KOP



ST

```
UINT_value := BOOL_TO_UINT(Boolean_value);
```

SDT_TO_INT

Special Double Data Register in INTEGER

Erklärung WORD_TO_UINT wandelt einen Wert vom Datentyp WORD in einen Wert vom Datentyp Unsigned INTEGER um.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_TO_UINT (s. S. 1199)**

Datentypen

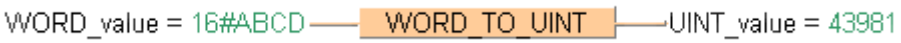
Datentyp	E/A	Funktion
WORD	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	WORD_value	WORD	16#ABCD

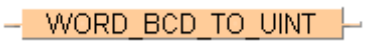
KOP



WORD_BCD_TO_UINT

Binärwert von WORD in vorzeichenlosen INTEGER

Erklärung WORD_BCD_TO_UINT wandelt einen Wert vom Datentyp WORDin einen Wert vom Datentyp Unsigned INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_BCD_TO_UINT (s. S. 1199)**

Datentypen

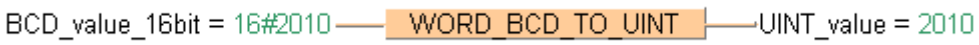
Datentyp	E/A	Funktion
WORD_BCD	Eingang	Eingangsdatentyp
UINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	BCD_value_16bit	WORD	16#2010

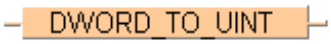
KOP



DWORD_TO_UINT

DOUBLE WORD in vorzeichenlosen INTEGER

Erklärung DWORD_TO_UINT wandelt einen Wert vom Datentyp DWORD in einen Wert vom Datentyp Unsigned INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_TO_UINT (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
DWORD	Eingang	Eingangsdatentyp
UINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	DWORD_value	DWORD	16#00001234

KOP

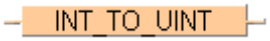
ST

```
UINT_value := DWORD_TO_UINT(DWORD_value);
```

INT_TO_UINT

INTEGER to Unsigned INTEGER

Erklärung INT_TO_UINT wandelt einen Wert vom Datentyp INTEGERin einen Wert vom Datentyp Unsigned INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INT_TO_UINT (s. S. 1193)**

Datentypen

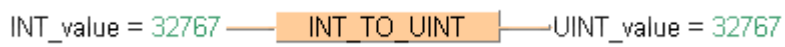
Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
UINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Kla...	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	INT_value	INT	32767

KOP



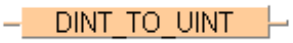
ST

```
UINT_value := INT_TO_UINT(INT_value);
```

DINT_TO_UINT

INTEGER in vorzeichenlosen INTEGER

Erklärung DINT_TO_UINT wandelt einen Wert vom Datentyp DINT in einen Wert vom Datentyp Unsigned INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DINT_TO_UINT (s. S. 1185)**

Datentypen

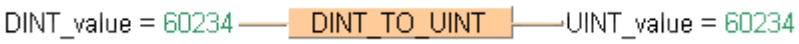
Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
UINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	DINT_value	DINT	60234

KOP



ST

```
UINT_value := DINT_TO_UINT(DINT_value);
```

UDINT_TO_UINT

vorzeichenloser DOUBLE INTEGER in vorzeichenlosen INTEGER

Erklärung UDINT_TO_UINT wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen Wert vom Datentyp Unsigned INTEGER um.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_UINT (s. S. 1198)**

Datentypen

Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
UINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	UDINT_value	UDINT	53123

KOP

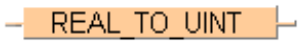
ST

```
UINT_value := UDINT_TO_UINT(UDINT_value);
```

REAL_TO_UINT

REAL in vorzeichenlosen INTEGER

Erklärung REAL_TO_UINT wandelt einen Wert vom Datentyp REAL in einen Wert vom Datentyp Unsigned INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Siehe auch: TRUNC_TO_UINT (s. S. 165)

SPS-Typen **Verfügbarkeit von REAL_TO_UINT (s. S. 1196)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
UINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	REAL_value	REAL	28.5

KOP

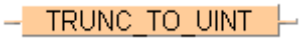
ST

```
UINT_value := REAL_TO_UINT(REAL_value);
```

TRUNC_TO_INT

TRUNC in INTEGER

Erklärung TRUNC_TO_UINT schneidet die Nachkommastellen einer Zahl vom Datentyp REAL ab und liefert eine Ausgangsvariable vom Datentyp Unsigned INTEGER.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TRUNC_TO_UINT (s. S. 1198)**

☞

- **Durch das Abschneiden der Nachkommastellen wird eine positive Zahl in Richtung 0 verkleinert und eine negative Zahl in Richtung 0 vergrößert.**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
INT	Ausgang	Umwandlungsergebnis

Fehlermerker

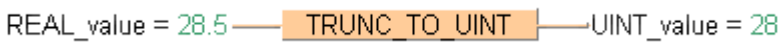
Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	dauerhaft	▪ die Eingangsvariable nicht vom Datentyp REAL ist
R9008	%MX0.900.8	kurzzeitig	▪ die Ausgangsvariable größer als ein 16-Bit-INTEGER ist
R9009	%MX0.900.9	kurzzeitig	▪ die Ausgangsvariable Null ist

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	REAL_value	REAL	28.5

KOP



ST

```
UINT_value := TRUNC_TO_UINT(REAL_value);
```

STRING_TO_UINT

STRING (Dezimalformat) in vorzeichenlosen INTEGER

Erklärung STRING_TO_UINT wandelt eine Zeichenfolge im Dezimalformat in einen Wert vom Datentyp Unsigned INTEGER um.

Siehe auch: STRING_TO_UINT_STEPSAVER (s. S. 167)

Zunächst wird die Zeichenfolge in einen Wert vom Datentyp STRING[32] umgewandelt. Das Ergebnis wird anschließend (von einem Unterprogramm mit ca. 270 Schritten) in einen Wert vom Datentyp UINT umgewandelt. Dieses Unterprogramm wird auch von den Funktionen STRING_TO_INT, STRING_TO_WORD, STRING_TO_UDINT und STRING_TO_DWORD verwendet.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Zulässiges Format:

'[Leerzeichen][Vorzeichen][Dezimalzahl][Leerzeichen]', z.B. ' 123456 '

Zulässige Zeichen:

Leerzeichen	Leerzeichen " "
Vorzeichen	Pluszeichen "+" und Minuszeichen "-"
Dezimalzahlen	Dezimalzahlen "0" - "9"

Die Analyse endet mit der ersten Nicht-Dezimalzahl.

SPS-Typen **Verfügbarkeit von STRING_TO_UINT (s. S. 1197)**

Datentypen

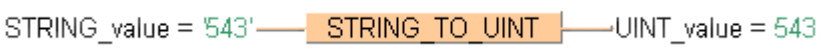
Datentyp	Kommentar
STRING	Eingang
UINT	Ausgang

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	STRING_value	STRING[8]	'543'

KOP



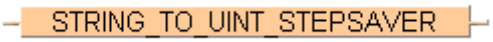
ST

```
UINT_value:=  STRING_TO_UINT(STRING_value);
```

STRING_TO_UINT_STEPSAVER

STRING (Dezimalformat rechtsbündig) in vorzeichenlosen DOUBLE INTEGER

Erklärung STRING_TO_UINT_STEPSAVER wandelt eine rechtsbündige Dezimalzahl in eine Zeichenfolge in einen Wert vom Datentyp Unsigned INTEGER um.



Hierbei wird der Basisbefehl F76_A2BIN (s. S. 616) mit ca. 7 Schritten verwendet. Wenn unzulässige Zeichen verwendet werden (siehe nachfolgende Tabelle "Zulässige Zeichen") wird ein Operationsfehler ausgegeben.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Zulässiges Format:

'[Leerzeichen][Vorzeichen][Dezimalzahl]', z.B. '␣123456'

Zulässige Zeichen:

Leerzeichen	Leerzeichen "␣"
Vorzeichen	Pluszeichen "+" und Minuszeichen "-"
Dezimalzahl	Dezimalzahlen "0" - "9"

SPS-Typen **Verfügbarkeit von STRING_TO_UINT_STEPSAVER (s. S. 1197)**

Datentypen

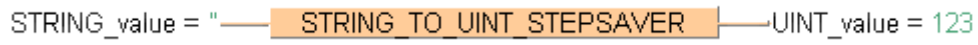
Datentyp	Kommentar
STRING	Eingang
UINT	Ausgang

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	0
1	VAR	STRING_value	STRING[8]	'543'

KOP



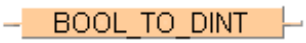
ST

```
UINT_value := STRING_TO_UINT_STEPSAVER(STRING_value);
```

BOOL_TO_DINT

BOOL in DOUBLE INTEGER

Erklärung BOOL_TO_DINT wandelt einen Wert vom Datentyp BOOL in einen Wert vom Datentyp DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von BOOL_TO_DINT (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
BOOL	Eingang	Eingangsdatentyp
DINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

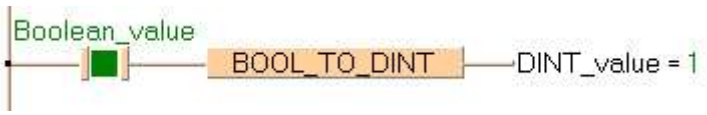
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Boolean_value	BOOL	FALSE
1	VAR	DINT_value	DINT	0

In diesem Beispiel wurde die Eingangsvariable (**Boolean_value**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Boolean_value** vom Datentyp BOOL wird in einen Wert vom Datentyp DOUBLE INTEGER umgewandelt. Der umgewandelte Wert wird in **DINT_value** geschrieben.

KOP



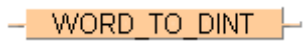
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF Boolean_value THEN
    DINT_value:=BOOL_TO_DINT(Boolean_value);
END_IF;
```

WORD_TO_DINT

WORD in DOUBLE INTEGER

Erklärung WORD_TO_DINT wandelt einen Wert vom Typ WORD in einen Wert vom Typ DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_TO_DINT (s. S. 1199)**

Datentypen

Datentyp	E/A	Funktion
WORD	Eingang	Eingangsdatentyp
DINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DINT_value	DINT	0
1	VAR	WORD_value	WORD	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **WORD_value** vom Datentyp WORD wird in einen Wert vom Datentyp INTEGER umgewandelt. Das Ergebnis wird in **DINT_value** geschrieben.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DINT_value:=WORD_TO_DINT(WORD_value);
```

BCD_TO_DINT

BCD_TO_DINT, BCD in DOUBLE INTEGER

Erklärung DWORD_BCD_TO_DINT konvertiert einen binär codierten Wert vom Datentyp DOUBLE WORD in einen Binärwert vom Datentyp DOUBLE INTEGER, damit ein BCD-Wert im Doppelwortformat verarbeitet werden kann.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_BCD_TO_DINT (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
DWORD_BCD	Eingang	Eingangsdatentyp
DINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

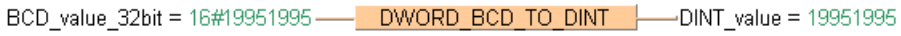
	Klasse	Bezeichner	Typ	Initial
0	VAR	BCD_value_32bit	DWORD	0
1	VAR	DINT_value	DINT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

In Control FPWIN Pro können BCD-Konstanten auch im folgenden Format dargestellt werden:
2#00011001100101010001100110010101 oder
16#19951995

Rumpf **BCD_value_32bit** vom Datentyp DOUBLE WORD wird in einen Wert vom Datentyp DOUBLE INTEGER umgewandelt. Der umgewandelte Wert wird in **DINT_value** geschrieben.

KOP



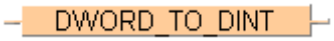
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DINT_value:=DWORD_BCD_TO_DINT(BCD_value_32bit);
```

DWORD_TO_DINT

DOUBLE WORD in DOUBLE INTEGER

Erklärung DWORD_TO_DINT wandelt einen Wert vom Datentyp DOUBLE WORD in einen Wert vom Datentyp DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_TO_DINT (s. S. 1185)**

☞ **Die Bitkombination der Eingangsvariablen wird der Ausgangsvariablen zugewiesen.**

Datentypen

Datentyp	E/A	Funktion
DWORD	Eingang	Eingangsdatentyp
DINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DWORD_value	DWORD	0
1	VAR	DINT_value	DINT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **DWORD_value** vom Datentyp DOUBLE WORD wird in einen Wert vom Datentyp DOUBLE INTEGER umgewandelt. Der umgewandelte Wert wird in **DINT_value** geschrieben.

KOP

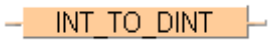
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DINT_value:=DWORD_TO_DINT(DWORD_value);
```

INT_TO_DINT

INTEGER in DOUBLE INTEGER

Erklärung INT_TO_DINT wandelt einen Wert vom Datentyp INTEGER in einen Wert vom Datentyp DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INT_TO_DINT (s. S. 1193)**

Datentypen

Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
DINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	INT_value	INT	0
1	VAR	DINT_value	DINT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **INT_value** vom Datentyp INTEGER wird in einen Wert vom Datentyp DOUBLE INTEGER umgewandelt. Das Ergebnis wird in **DINT_value** geschrieben.

KOP

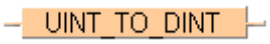
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DINT_value:=INT_TO_DINT(INT_value);
```

UINT_TO_DINT

vorzeichenloser INTEGER in DOUBLE INTEGER

Erklärung UINT_TO_DINT wandelt einen Wert vom Datentyp Unsigned INTEGER in einen Wert vom Datentyp DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UINT_TO_DINT (s. S. 1198)**

Datentypen

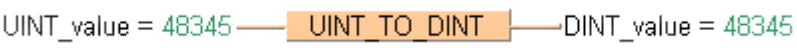
Datentyp	E/A	Funktion
UINT	Eingang	Eingangsdatentyp
DINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	48345
1	VAR	DINT_value	DINT	0

KOP



ST

```
DINT_value := UINT_TO_DINT(UINT_value);
```

UDINT_TO_DINT

vorzeichenloser DOUBLE INTEGER in DOUBLE INTEGER

Erklärung UDINT_TO_DINT wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen Wert vom Datentyp DOUBLE INTEGER um.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_DINT (s. S. 1198)**

Datentypen

Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
DINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	2147483647
1	VAR	DINT_value	DINT	0

KOP

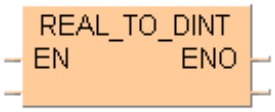
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DINT_value := UDINT_TO_DINT (UDINT_value);
```

REAL_TO_DINT

REAL in DOUBLE INTEGER

Erklärung REAL_TO_DINT wandelt einen Wert vom Datentyp REAL in einen Wert vom Datentyp DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von REAL_TO_DINT (s. S. 1195)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
DINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	REAL_value	REAL	0.0
1	VAR	DINT_value	DINT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **REAL_value** vom Datentyp REAL wird in einen Wert vom Datentyp DOUBLE INTEGER umgewandelt. Der umgewandelte Wert wird in **DINT_value** gespeichert.

KOP

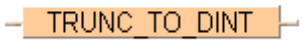
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DINT_value := REAL_TO_DINT(REAL_value);
```

TRUNC_TO_DINT

TRUNC in DOUBLE INTEGER

Erklärung TRUNC_TO_DINT schneidet die Nachkommastellen einer Zahl vom Datentyp REAL ab und liefert eine Ausgangsvariable vom Datentyp DOUBLE INTEGER.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TRUNC_TO_DINT (s. S. 1198)**

☞ **Durch das Abschneiden der Nachkommastellen wird eine positive Zahl in Richtung 0 verkleinert und eine negative Zahl in Richtung 0 vergrößert.**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
DINT	Ausgang	Umwandlungsergebnis

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Eingangsvariable nicht den Datentyp REAL hat
R9008	%MX0.900.8	kurzzeitig	▪ Ausgangsvariable größer als ein 32-Bit DINT ist
R9009	%MX0.900.9	kurzzeitig	▪ die Ausgangsvariable Null ist

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	REAL_value	REAL	0.0	number betw. -2147483.000 ... +2147483.000
1	VAR	DINT_value	DINT	0	number betw. -2147483 ... +2147483

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf Von **REAL_value** werden die Nachkommastellen abgeschnitten. Das Ergebnis wird als 32-Bit DOUBLE INTEGER in **DINT_value** geschrieben.

KOP

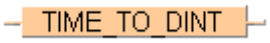
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DINT_value:=TRUNC_TO_DINT(REAL_value);
```

TIME_TO_DINT

TIME in DOUBLE INTEGER

Erklärung TIME_TO_DINT wandelt einen Wert vom Datentyp TIME in einen Wert vom Datentyp DOUBLE INTEGER um. Die Zeit 10 ms entspricht dem Wert 1, z.B. wird der Eingangswert T#1m 0s in den Wert 6000 gewandelt.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TIME_TO_DINT (s. S. 1197)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang	Eingangsdatentyp
DINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Class	Identifier	Type	Initial
0	VAR	time_value	TIME	T100ms
1	VAR	DINT_value	DINT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **Time_value** vom Datentyp TIME wird in einen Wert vom Datentyp DOUBLE INTEGER umgewandelt. Das Ergebnis wird in die Ausgangsvariable **DINT_value** geschrieben.

KOP

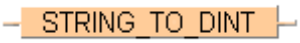
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DINT_value:=TIME_TO_DINT(time_value);
```

STRING_TO_DINT

STRING (Dezimal-Format) in DOUBLE INTEGER

Erklärung Diese Funktion wandelt eine Zeichenkette im Dezimal-Format in einen Wert vom Datentyp DINT um.

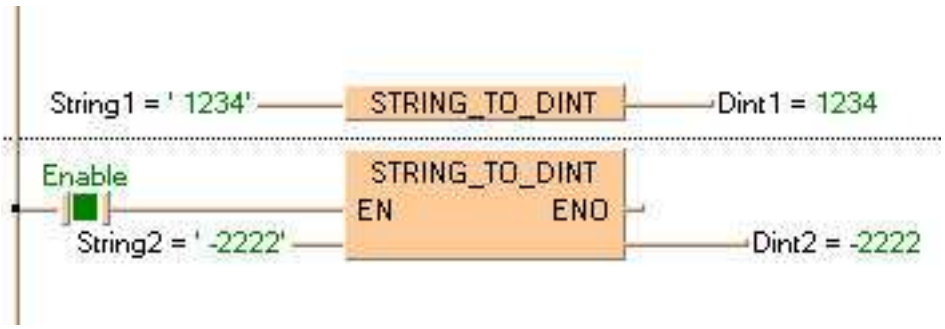
Symbol: 

Dabei wird die anliegende Zeichenkette zunächst in einen Wert vom Datentyp STRING[32] gewandelt. Anschließend wird dieser in einem Unterprogramm mit ca. 270 Schritten, welches gemeinsam für die Funktionen STRING_TO_INT, STRING_TO_WORD, STRING_TO_DINT oder STRING_TO_DWORD verwendet wird, in einen Wert vom Datentyp DINT gewandelt.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Siehe auch: STRING_TO_DINT_STEPSAVER

Beispiel:



Zulässiges Format:

´[Leerzeichen][Vorzeichen][Dezimalzahlen][Leerzeichen]´ z.B. ´ 123456 ´

Zulässige Zeichen:

Leerzeichen	Leerzeichen “ “
Vorzeichen	Pluszeichen “+” und Minuszeichen “-“
Dezimalzahlen	Dezimalzahlen von "0“ - "9“

Die Analyse endet mit der ersten Nicht-Dezimalzahl.

SPS-Typen **Verfügbarkeit von STRING_TO_DINT (s. S. 1197)**

Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
DINT	Ausgangsvariable

STRING_TO_DINT_ STEPSAVER

STRING (Dezimal-Format rechtsbündig) in DOUBLE INTEGER

Erklärung Diese Funktion wandelt eine rechtsbündige Dezimalzahl in einer Zeichenkette in einen Wert vom Datentyp DINT um.

Hierbei wird direkt der Befehl F78_DA2BIN (s. S. 622) mit nur ca. 11 Schritten verwendet. Die Steuerung liefert insbesondere dann einen Operationsfehler, wenn ein Buchstabe erscheint, der keiner Dezimalzahl "0 - 9" und nicht "+" oder "-" oder führenden Leerzeichen entspricht.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiel:

Zulässiges Format:

´[Leerzeichen][Vorzeichen][Dezimalzahlen]´ z.B. ´ 123456´

Zulässige Zeichen:

Leerzeichen	Leerzeichen " "
Vorzeichen	Pluszeichen "+" und Minuszeichen "-"
Dezimalzahlen	Dezimalzahlen von "0" - "9"

SPS-Typen **Verfügbarkeit von STRING_TO_DINT STEPSAVER (s. S. 1197)**

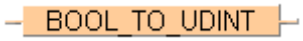
Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
DINT	Ausgangsvariable

BOOL_TO_DINT

BOOL in vorzeichenlosen DOUBLE INTEGER

Erklärung BOOL_TO_UDINT wandelt einen Wert vom Datentyp BOOL in einen Wert vom Datentyp Unsigned DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von BOOL_TO_UDINT (s. S. 1184)**

Datentypen

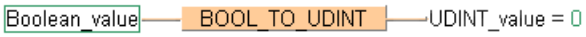
Datentyp	E/A	Funktion
BOOL	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	100546
1	VAR	Boolean_value	BOOL	FALSE

KOP



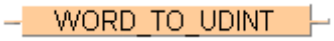
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
UDINT_value := BOOL_TO_UDINT (Boolean_value);
```

WORD_TO_UDINT

WORD in Unsigned DOUBLE INTEGER

Erklärung WORD_TO_UDINT wandelt einen Wert vom Datentyp WORD in einen Wert vom Datentyp Unsigned DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_TO_UDINT (s. S. 1199)**

Datentypen

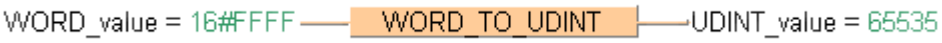
Datentyp	E/A	Funktion
WORD	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	0
1	VAR	WORD_value	WORD	16#FFFF

KOP



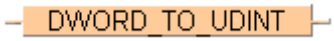
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
UDINT_value :=WORD_TO_UDINT(WORD_value);
```

DWORD_TO_UDINT

WORD in vorzeichenlosen DOUBLE INTEGER

Erklärung DWORD_TO_UDINT wandelt einen Wert vom Datentyp DOUBLE WORD in einen Wert vom Datentyp Unsigned DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_TO_UDINT (s. S. 1185)**

Datentypen

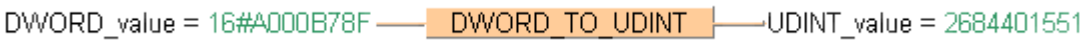
Datentyp	E/A	Funktion
DWORD	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	0
1	VAR	DWORD_value	DWORD	16#A000B78F

KOP



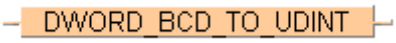
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
UDINT_value := DWORD_TO_UDINT (DWORD_value);
```

DWORD_BCD_TO_UDINT

Binärwert von DOUBLE WORD in vorzeichenlosen DOUBLE INTEGER

Erklärung DWORD_BCD_TO_UDINT wandelt einen Wert vom Datentyp DOUBLE WORD in einen Wert vom Datentyp Unsigned DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_BCD_TO_UDINT (s. S. 1185)**

Datentypen

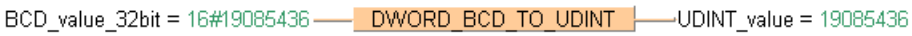
Datentyp	E/A	Funktion
DWORD_BCD	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	419976246
1	VAR	BCD_value_32bit	DWORD	16#19085436

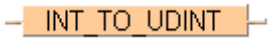
KOP



INT_TO_UDINT

INTEGER in vorzeichenlosen DOUBLE INTEGER

Erklärung INT_TO_UDINT wandelt einen Wert vom Datentyp INTEGER in einen Wert vom Datentyp Unsigned DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INT_TO_UDINT (s. S. 1193)**

Datentypen

Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	0
1	VAR	INT_value	INT	32767

KOP

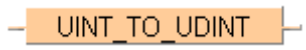
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
UDINT_value := INT_TO_UDINT(INT_value);
```

UINT_TO_UDINT

vorzeichenloser INTEGER in vorzeichenlosen DOUBLE INTEGER

Erklärung UINT_TO_UDINT wandelt einen Wert vom Datentyp Unsigned INTEGER in einen Wert vom Datentyp Unsigned DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UINT_TO_UDINT (s. S. 1198)**

Datentypen

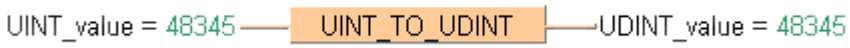
Datentyp	E/A	Funktion
UINT	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	48345
1	VAR	UDINT_value	UDINT	0

KOP



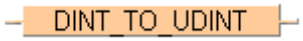
ST

```
UDINT_value := UINT_TO_UDINT (UINT_value);
```

DINT_TO_UDINT

DOUBLE INTEGER in vorzeichenlosen DOUBLE INTEGER

Erklärung DINT_TO_UDINT wandelt einen Wert vom Datentyp DOUBLE INTEGER in einen Wert vom Datentyp Unsigned DOUBLE INTEGER um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DINT_TO_UDINT (s. S. 1185)**

Datentypen

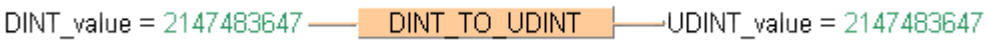
Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	0
1	VAR	DINT_value	DINT	2147483647

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

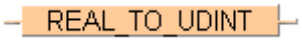
```
UDINT_value := DINT_TO_UDINT (DINT_value);
```

REAL_TO_UDINT

REAL in vorzeichenlosen DOUBLE INTEGER

Erklärung REAL_TO_UDINT wandelt einen Wert vom Datentyp REAL in einen Wert vom Datentyp Unsigned DOUBLE INTEGER um. Das Ergebnis wird für die Konvertierung auf die nächste Ganzzahl abgerundet

Siehe auch: TRUNC_TO_UDINT (s. S. 188)



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von REAL_TO_UDINT (s. S. 1196)**

☞ **Zahlen vom Datentyp REAL sind auf ca. 7 Stellen beschränkt. Für größere Zahlen können die Daten daher nicht gespeichert werden.**

Datentypen

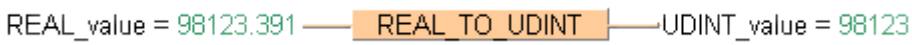
Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	0
1	VAR	REAL_value	REAL	98123.39

KOP



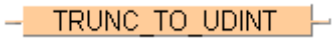
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
UDINT_value := REAL_TO_UDINT (REAL_value);
```

TRUNC_TO_UDINT

Nachkommastellen der REAL-Eingangsvariable abschneiden, in vorzeichenlosen DOUBLE INTEGER umwandeln

Erklärung TRUNC_TO_UDINT schneidet die Nachkommastellen einer Zahl vom Datentyp REAL ab und liefert eine Ausgangsvariable vom Datentyp Unsigned DOUBLE INTEGER.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TRUNC_TO_UDINT (s. S. 1198)**

- Durch das Abschneiden der Nachkommastellen wird eine positive Zahl in Richtung 0 verkleinert und eine negative Zahl in Richtung 0 vergrößert.
- Zahlen vom Datentyp REAL sind auf ca. 7 Stellen beschränkt. Für größere Zahlen können die Daten daher nicht gespeichert werden.

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Fehlermerker

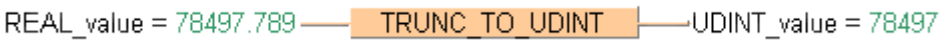
Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	dauerhaft	▪ die Eingangsvariable nicht vom Datentyp REAL ist
R9008	%MX0.900.8	kurzzeitig	▪ die Ausgangsvariable größer als ein 32-Bit-DOUBLE INTEGER ist
R9009	%MX0.900.9	kurzzeitig	▪ die Ausgangsvariable Null ist

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	0
1	VAR	REAL_value	REAL	78497.79

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

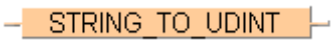
```
UDINT_value := TRUNC_TO_UDINT (REAL_value);
```

STRING_TO_UDINT

STRING (Dezimalformat) in vorzeichenlosen DOUBLE INTEGER

Erklärung

STRING_TO_UDINT wandelt eine Zeichenfolge im Dezimalformat in einen Wert vom Datentyp Unsigned DOUBLE INTEGER um.



Zunächst wird die Zeichenfolge in einen Wert vom Datentyp STRING[32] umgewandelt. Das Ergebnis wird anschließend (von einem Unterprogramm mit ca. 270 Schritten) in einen Wert vom Datentyp UDINT umgewandelt . Dieses Unterprogramm wird auch von den Funktionen STRING_TO_INT, STRING_TO_WORD, STRING_TO_UDINT und STRING_TO_DWORD verwendet.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Zulässiges Format:

'[Leerzeichen][Vorzeichen][Dezimalzahl][Leerzeichen]', z.B. ' 123456 '

Zulässige Zeichen:

Leerzeichen	Leerzeichen " "
Vorzeichen	Pluszeichen "+" und Minuszeichen "-"
Dezimalzahlen	Dezimalzahlen "0" - "9"

Die Analyse endet mit der ersten Nicht-Dezimalzahl.

SPS-Typen **Verfügbarkeit von STRING_TO_UDINT (s. S. 1197)**

Datentypen

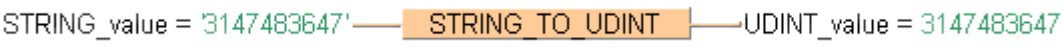
Datentyp	Kommentar
STRING	Eingang
UDINT	Ausgang

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	0
1	VAR	STRING_value	STRING[10]	'3147483647'

KOP



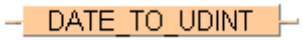
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
UDINT_value := STRING_TO_UDINT (STRING_value);
```

DATE_TO_UDINT

DATE in vorzeichenlosen DOUBLE INTEGER

Erklärung DATE_TO_UDINT konvertiert einen Wert vom Datentyp DATE in einen Wert vom Datentyp Unsigned DOUBLE INTEGER entsprechend der internen Datumsdarstellung, d.h. abgelaufene Sekunden nach 2001-01-01.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DATE_TO_UDINT (s. S. 1185)**

Datentypen

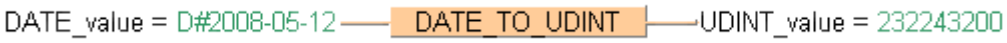
Datentyp	E/A	Funktion
DATE	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	0
1	VAR	DATE_value	DATE	D#2008-05-12

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

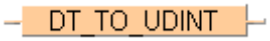
```
UDINT_value := DATE_TO_UDINT (DATE_value);
```

DT_TO_UDINT

DATE_AND_TIME in vorzeichenlosen DOUBLE INTEGER

Erklärung

DT_TO_UDINT konvertiert einen Wert vom Datentyp DATE_AND_TIME in einen Wert vom Datentyp Unsigned DOUBLE INTEGER entsprechend der internen Datumsdarstellung, d.h. abgelaufene Sekunden nach 2001-01-01-00:00:00.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von DT_TO_UDINT (s. S. 1185)

Datentypen

Datentyp	E/A	Funktion
DT	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel

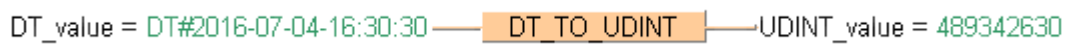
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	0
1	VAR	DT_value	DATE_AND_TIME	DT#2016-07-04-16:30:30

KOP



ST

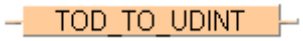
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
UDINT_value := DT_TO_UDINT(DT_value);
```

TOD_TO_UDINT

TIME_OF_DAY in vorzeichenlosen DOUBLE INTEGER

Erklärung TOD_TO_UDINT konvertiert einen Wert vom Datentyp TIME_OF_DAY in einen Wert vom Datentyp Unsigned DOUBLE INTEGER entsprechend der internen Datumsdarstellung, d.h. abgelaufene Sekunden nach 2001-01-01-00:00:00.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TOD_TO_UDINT (s. S. 1198)**

Datentypen

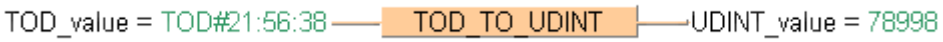
Datentyp	E/A	Funktion
TOD	Eingang	Eingangsdatentyp
UDINT	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	0
1	VAR	TOD_value	TOD	TOD#21:56:38

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
UDINT_value := TOD_TO_UDINT (TOD_value);
```

DWORD_TO_REAL

DOUBLE WORD in REAL

Erklärung DWORD_TO_REAL verschiebt die Bitstring-Informationen einer DWORD-Variable in eine REAL-Variable. Mit DWORD_OVERLAPPING_DUT kann die gleiche Funktionalität erzielt werden.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_TO_REAL (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
DWORD	Eingang	Eingangsdatentyp
REAL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	REAL_value	REAL	0.0
1	VAR	DWORD_value	DWORD	16#4007F80D

KOP

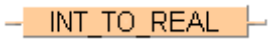
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
REAL_value := DWORD_TO_REAL(DWORD_value);
```

INT_TO_REAL

INTEGER in REAL

Erklärung INT_TO_REAL wandelt einen Wert vom Datentyp INTEGER in einen Wert vom Datentyp REAL um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INT_TO_REAL (s. S. 1193)**

Datentypen

Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
REAL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

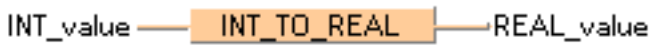
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	INT_value	INT	0
1	VAR	REAL_value	REAL	0.0

In diesem Beispiel wurde die Eingangsvariable (**INT_value**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **INT_value** vom Datentyp INTEGER wird in einen Wert vom Datentyp REAL umgewandelt. Der umgewandelte Wert wird in **REAL_value** gespeichert.

KOP



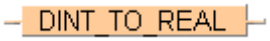
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
REAL_value:=INT_TO_REAL(INT_value);
```

DINT_TO_REAL

DOUBLE INTEGER in REAL

Erklärung DINT_TO_REAL wandelt einen Wert vom Datentyp DOUBLE INTEGER in einen Wert vom Datentyp REAL um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DINT_TO_REAL (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
REAL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	REAL_value	REAL	0.0
1	VAR	DINT_value	DINT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **DINT_value** vom Datentyp DOUBLE INTEGER wird in einen Wert vom Datentyp REAL umgewandelt. Der umgewandelte Wert wird in **REAL_value** gespeichert.

KOP

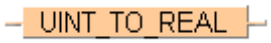
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
REAL_value:=DINT_TO_REAL(DINT_value);
```

UINT_TO_REAL

vorzeichenloser INTEGER in REAL

Erklärung UINT_TO_REAL wandelt einen Wert vom Datentyp Unsigned INTEGER in einen Wert vom Datentyp REAL um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UINT_TO_REAL (s. S. 1198)**

☞ **Zahlen vom Datentyp REAL sind auf ca. 7 Stellen beschränkt. Für größere Zahlen können die Daten daher nicht gespeichert werden.**

Datentypen

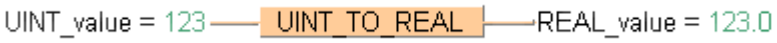
Datentyp	E/A	Funktion
UINT	Eingang	Eingangsdatentyp
REAL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	123
1	VAR	REAL_value	REAL	0.0

KOP



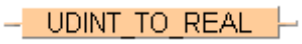
ST

```
REAL_value := UINT_TO_REAL(UINT_value);
```

UDINT_TO_REAL

vorzeichenloser DOUBLE INTEGER in REAL

Erklärung UDINT_TO_REAL wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen Wert vom Datentyp REAL um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_REAL (s. S. 1198)**

☞ **Zahlen vom Datentyp REAL sind auf ca. 7 Stellen beschränkt. Für größere Zahlen können die Daten daher nicht gespeichert werden.**

Datentypen

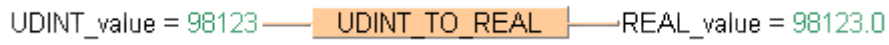
Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
REAL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	98123
1	VAR	REAL_value	REAL	0.0

KOP



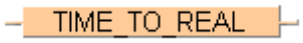
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
REAL_value := UDINT_TO_REAL (UDINT_value);
```

TIME_TO_REAL

TIME in REAL

Erklärung TIME_TO_REAL wandelt einen Wert vom Typ TIME in einen Wert vom Typ REAL um. 10ms des Datentyps TIME entsprechen 1,0 REAL-Einheit, z.B. wenn TIME = 10ms, REAL = 1.0, ; wenn TIME = 1s, REAL = 100.0. Die Resolution ist 10ms.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TIME_TO_REAL (s. S. 1197)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang	Eingangsdatentyp
REAL	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_time	TIME	T#1h1m1s	
1	VAR	result_time	REAL	0.0	result: here 366100.0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

KOP

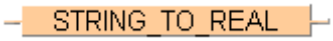
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
result_real :=TIME_TO_REAL (input_time);
```

STRING_TO_REAL

STRING in REAL

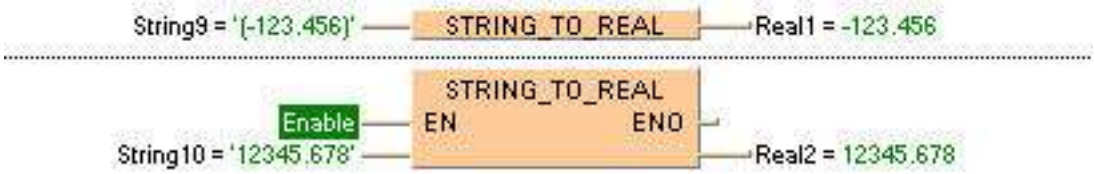
Erklärung Diese Funktion wandelt eine Zeichenkette (STRING) im Fließkommaformat in einen Wert vom Datentyp REAL um.

Symbol: 

Dabei wird die anliegende Zeichenkette zunächst in einen Wert vom Datentyp STRING[32] gewandelt. Anschließend wird dieser dann in einem Unterprogramm mit ca. 290 Schritten in einen Wert vom Datentyp REAL gewandelt.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiel:



Zulässiges Format:

'[Leerzeichen][Vorzeichen][Dezimalzahlen][.][Dezimalzahlen][Leerzeichen]' z.B. ' -123.456 '

Zulässige Zeichen:

Leerzeichen	Alle Zeichen außer "+“ (Plus), "-" (Minus) und allen Dezimalzahlen
Dezimalzahlen	Dezimalzahlen von "0“ - "9“

Die Analyse endet mit der ersten Nicht-Dezimalzahl.

SPS-Typen **Verfügbarkeit von STRING_TO_REAL (s. S. 1197)**

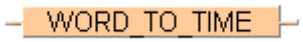
Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
REAL	Ausgangsvariable

WORD_TO_TIME

WORD in TIME

Erklärung WORD_TO_TIME wandelt einen Wert vom Typ WORD in einen Wert vom Datentyp TIME um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_TO_TIME (s. S. 1199)**

Datentypen

Datentyp	E/A	Funktion
WORD	Eingang	Eingangsdatentyp
TIME	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

Beispiele:

Eingangsvariable	Ausgangsvariable
12345	T#123.45s
16#0012	T#180.00ms

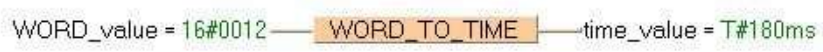
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	WORD_value	WORD	0
1	VAR	time_value	TIME	T#0s

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **WORD_value** vom Datentyp WORD (16 Bit) wird in einen Wert vom Datentyp TIME (16 Bit) umgewandelt. Das Ergebnis wird in die Ausgangsvariable **Time_value** geschrieben.

KOP



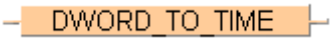
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
time_value:=WORD_TO_TIME(WORD_value);
```

DWORD_TO_TIME

DOUBLE WORD in TIME

Erklärung DWORD_TO_TIME wandelt einen Wert vom Datentyp DWORD in einen Wert vom Datentyp TIME um. Der Wert 1 entspricht einer Zeit von 10ms, z.B. wird der Eingangswert 12345 (16# 3039) in die Zeit T# 2m 3s 450.00ms gewandelt.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_TO_TIME (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
DWORD	Eingang	Eingangsdatentyp
TIME	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DWORD_value	DWORD	0
1	VAR	time_value	TIME	T#0s

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **DWORD_value** vom Datentyp DWORD (32 Bit) wird in einen Wert vom Datentyp TIME (16 Bit) umgewandelt. Das Ergebnis wird in die Ausgangsvariable **Time_value** geschrieben.

KOP

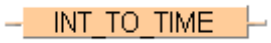
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
time_value:=DWORD_TO_TIME(DWORD_value);
```

INT_TO_TIME

INTEGER in TIME

Erklärung INT_TO_TIME wandelt einen Wert vom Datentyp INT in einen Wert vom Datentyp TIME um. Die Zeitbasis beträgt 10ms; z.B. wenn INT=350, dann ist der Wert vom Typ TIME 3s 500ms.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INT_TO_TIME (s. S. 1193)**

Datentypen

Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
TIME	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	INT_value	INT	0
1	VAR	time_value	TIME	T#0s

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **INT_value** vom Datentyp INTEGER wird in einen Wert vom Datentyp TIME umgewandelt. Das Ergebnis wird in die Ausgangsvariable **Time_value** geschrieben.

KOP

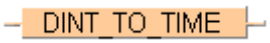
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
time_value:=INT_TO_TIME(INT_value);
```

DINT_TO_TIME

DOUBLE INTEGER in TIME

Erklärung DINT_TO_TIME wandelt einen Wert vom Datentyp DINT in einen Wert vom Datentyp TIME um. Der Wert 1 entspricht einer Zeit von 10ms, z.B. wird der Eingangswert 123 in die Zeit T#1s 230.00ms gewandelt.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DINT_TO_TIME (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
TIME	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	DINT_value	DINT	0	
1	VAR	TIME_value	TIME	T#0s	result: T#1s230.00ms

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **DINT_value** vom Datentyp DOUBLE INTEGER wird in einen Wert vom Datentyp TIME umgewandelt. Das Ergebnis wird in die Ausgangsvariable **Time_value** geschrieben.

KOP

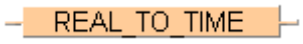
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
time_value :=DINT_TO_TIME (DINT_value);
```

REAL_TO_TIME

REAL in TIME

Erklärung REAL_TO_TIME wandelt einen Wert vom Typ REAL in einen Wert vom Typ TIME um. 10ms des Datentyps TIME entsprechen 1.0 REAL-Einheit, z.B. wenn REAL = 1.0, TIME = 10ms; wenn REAL = 100.0, TIME = 1s. Für die Umwandlung wird der Wert des Datentyps REAL auf eine ganze Zahl auf- bzw. abgerundet.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von REAL_TO_TIME (s. S. 1196)**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
TIME	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe. Da Konstante direkt an die Eingänge der Funktion geschrieben wurden, müssen Sie nur die Ausgangsvariable im POE-Kopf deklarieren.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	result_time	TIME	T#0s

Rumpf Wenn Sie das Fernglassymbol im Online-Modus anklicken, können Sie sofort das Ergebnis T#0.00ms sehen. Da der Wert am REAL-Eingang weniger als 0.5 ist, wird er auf 0.0 abgerundet.

KOP

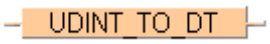
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
result_time := REAL_TO_TIME (0.499);
```

UDINT_TO_DT

vorzeichenloser DOUBLE INTEGER in DATE_AND_TIME

Erklärung UDINT_TO_DT wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen Wert vom Datentyp DATE_AND_TIME um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_DT (s. S. 1198)**

Datentypen

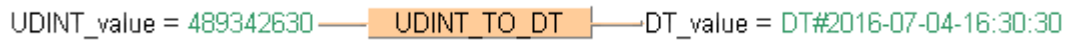
Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
DATE_AND_TIME	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	489342630
1	VAR	DT_value	DATE_AND_TIME	DT#2001-01-01-00:00:00

KOP



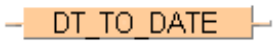
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DT_value :=UDINT_TO_DT(UDINT_value);
```

DT_TO_DATE

DATE_AND_TIME in DATE

Erklärung DT_TO_DATE wandelt einen Wert vom Datentyp DATE_AND_TIME in einen Wert vom Datentyp DATE um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DT_TO_DATE (s. S. 1185)**

Datentypen

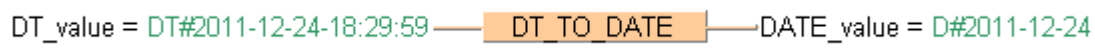
Datentyp	E/A	Funktion
DATE_AND_TIME	Eingang	Datum und Dauer
DATE	Ausgang	Datum

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Kl...	Bezeichner	Typ	Initial
0	VAR	DT_value	DATE_AND_TIME	DT#2011-12-24-18:29:59
1	VAR	DATE_value	DATE	D#2001-01-01

KOP



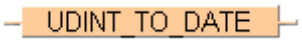
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DATE_value := DT_TO_DATE (DT_value);
```

UDINT_TO_DATE

vorzeichenloser DOUBLE INTEGER in DATE

Erklärung UDINT_TO_DATE wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen Wert vom Datentyp DATE um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_DATE (s. S. 1198)**

Datentypen

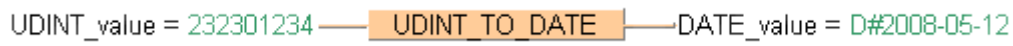
Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
DATE	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	232301234
1	VAR	DATE_value	DATE	D#2001-01-01

KOP



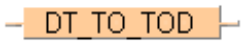
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DATE_value := UDINT_TO_DATE (UDINT_value);
```

DT_TO_TOD

DATE_AND_TIME in TIME_OF_DAY

Erklärung DT_TO_TOD wandelt einen Wert vom Datentyp DATE_AND_TIME in einen Wert vom Datentyp TIME_OF_DAY um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DT_TO_TOD (s. S. 1185)**

Datentypen

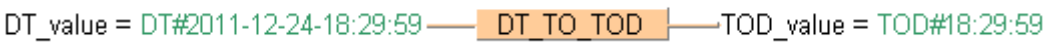
Datentyp	E/A	Funktion
DATE_AND_TIME	Eingang	Eingangsdatentyp
TIME_OF_DAY	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DT_value	DATE_AND_TIME	DT#2011-12-24-18:29:59
1	VAR	TOD_value	TOD	TOD#00:00:00

KOP



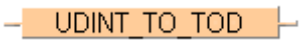
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
TOD_value := DT_TO_TOD (DT_value);
```

UDINT_TO_TOD

vorzeichenloser DOUBLE INTEGER in TIME_OF_DAY

Erklärung UDINT_TO_TOD wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen Wert vom Datentyp TIME_OF_DAY um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_TOD (s. S. 1198)**

Datentypen

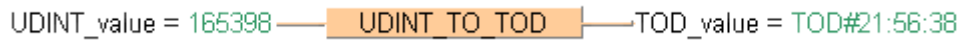
Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
TIME_OF_DAY	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	165398
1	VAR	TOD_value	TOD	TOD#00:00:00

KOP



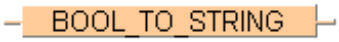
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
TOD_value := UDINT_TO_TOD (UDINT_value);
```

BOOL_TO_STRING

BOOL in STRING

Erklärung Die Funktion BOOL_TO_STRING wandelt einen Wert vom Datentyp BOOL in einen Wert vom Datentyp STRING[2]. Der Ergebnisstring wird als '0' oder '1' dargestellt.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞

- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Weitere Informationen finden Sie in der Online-Hilfe im Thema: Aktualisierungsprobleme bei Datentyp STRING**

SPS-Typen **Verfügbarkeit von BOOL_TO_STRING (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
BOOL	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

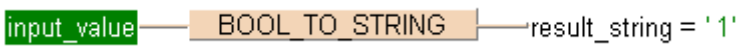
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	BOOL	TRUE	example value
1	VAR	result_string	STRING[2]	''	result: here '1'

Die Eingangsvariable **input_value** vom Datentyp BOOL wird mit dem Wert TRUE initialisiert. Die Ausgangsvariable **result_string** ist vom Datentyp STRING[2]. Sie kann maximal zwei Zeichen speichern. Sie können auch eine Zeichenkette mit mehr als einem Zeichen deklarieren z.B. STRING[5]. Von den reservierten 5 Zeichen würden dann 2 nicht benutzt werden.
Statt der Variablen **input_value**, können Sie im Rumpf die Konstanten TRUE oder FALSE direkt an den Eingang der Funktion schreiben.

Rumpf Der **input_value** vom Datentyp BOOL wird in einen STRING[2] umgewandelt. Der konvertierte Wert wird in **result_string** geschrieben. Mit dem **input_value** = TRUE steht im **result_string** '1'.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF Boolean_value THEN
    output_value:=BOOL_TO_STRING(input_value);
END_IF;
```

Beispiel 2

Wenn Sie als Ergebnis statt '0' oder '1' 'TRUE' oder 'FALSE' wollen, können Sie die Funktion BOOL_TO_STRING nicht verwenden. Dieses Beispiel zeigt, wie man mit einem Eingangswert vom Datentyp BOOL einen STRING der Länge 5 erzeugt, welcher die Zeichen 'TRUE' oder 'FALSE' beinhaltet.

Das Beispiel wird in KOP und in AWL programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POU header

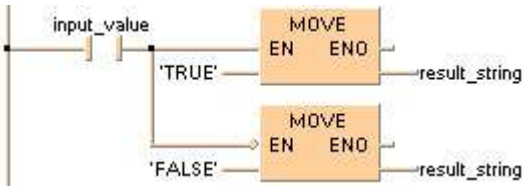
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	BOOL	TRUE	example value
1	VAR	result_string	STRING[5]	''	result: here 'TRUE'

In diesem Beispiel werden eine Eingangsvariable **input_value** vom Datentyp BOOL sowie eine Ausgangsvariable **result_string** vom Datentyp STRING[5] deklariert.

Body

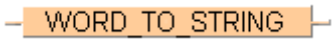
Zur Realisierung der geforderten Aufgabe wird die Standardfunktion E_MOVE verwendet. Sie weist den Wert ihres Eingangs unverändert dem Ausgang zu. Am Eingang wird die STRING-Konstante 'TRUE' bzw. 'FALSE' angelegt. Es findet eine "quasi BOOL to STRING-Konvertierung" statt, da die boole'sche Variable input_value am EN-Freigabe-Eingang über die Ausgabe des STRING entscheidet.

LD



WORD_TO_STRING WORD in STRING

Erklärung Die Funktion WORD_TO_STRING wandelt einen Wert vom Datentyp WORD in einen Wert vom Datentyp STRING. Sie erzeugt eine rechtsbündige Ergebniszeichenkette in Hexadezimaldarstellung. Diese wird mit führenden Nullen aufgefüllt, bis die maximale Anzahl der Zeichen erreicht ist, die für diese Zeichenkette definiert ist.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiele:

Eingang	Ausgang definiert als	liefert
16#ABCD	STRING[1]	'D'
	STRING[2]	'CD'
	STRING[3]	'BCD'
	STRING[4]	'ABCD'
	STRING[5]	'0ABCD'
	STRING[6]	'00ABCD'
	und so weiter ...	

SPS-Typen **Verfügbarkeit von WORD_TO_STRING (s. S. 1199)**

☞
- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Weitere Informationen finden Sie in der Online-Hilfe im Thema: Aktualisierungsprobleme bei Datentyp STRING**

Datentypen

Datentyp	E/A	Funktion
WORD	Eingang	Eingangsdatentyp
STRING	Ausgabe	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	WORD_value	WORD	0	example value
1	VAR	result_string	STRING[6]	''	result: here '00ABCD'

Die Eingangsvariable **input_value** vom Datentyp WORD wird mit dem Wert 16#ABCD initialisiert. Die Ausgangsvariable **result_string** ist vom Datentyp STRING[6]. Sie kann maximal 6 Zeichen speichern. Statt der Variablen **input_value**, können Sie im Rumpf eine Konstante direkt an den Eingang der Funktion schreiben.

Rumpf Der **input_value** vom Datentyp WORD wird in einen STRING[6] umgewandelt. Der konvertierte Wert wird in **result_string** geschrieben. Mit der Variablen **input_value** = 16#ABCD steht im **result_string** '00ABCD'.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
result_string:=WORD_TO_STRING(input_value);
```

Beispiel 2 Dieses Beispiel zeigt, wie man aus einem Eingangswert vom Datentyp WORD einen STRING der Länge 4 erzeugt, in dem der führende Teilstring '16#' abgeschnitten wird.

Das Beispiel wird in KOP und in AWL programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POU header

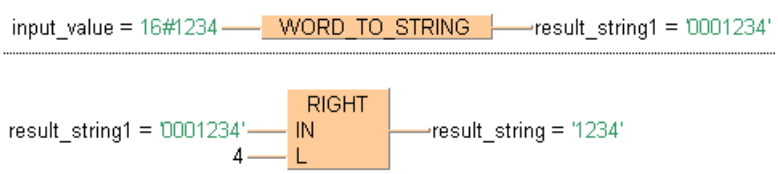
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	WORD	16#1234	example value
1	VAR	result_string1	STRING[7]	''	result: here '0001234'
2	VAR	result_string	STRING[4]	''	result: here '1234'

In diesem Beispiel werden eine Eingangsvariable **input_value** vom Datentyp BOOL sowie eine Ausgangsvariable **result_string** vom Datentyp STRING[4] deklariert.

Body Zur Realisierung der geforderten Aufgabe wird der Funktion WORD_TO_STRING zusätzlich die Standardfunktion RIGHT nachgeschaltet. Sie erzeugt eine rechtsbündige Zeichenkette der Länge **L**.

Im Beispiel liegt am Eingang der RIGHT-Funktion der Ausgangsstring von WORD_TO_STRING an. Am L-Eingang von RIGHT gibt die INT-Konstante 4 die Länge des auszugebenden rechtsbündigen STRING an. Aus der Variablen **input_value** = 16#1234 wird nach der Typwandlung und der RIGHT-Funktion der **result_string** '1234'.

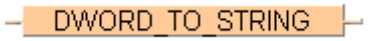
LD



DWORD_TO_STRING

DOUBLE WORD in STRING

Erklärung Die Funktion DWORD_TO_STRING wandelt einen Wert vom Datentyp DWORD in einen Wert vom Datentyp STRING. Sie erzeugt eine rechtsbündige Ergebniszeichenkette in Hexadezimaldarstellung. Diese wird mit führenden Nullen aufgefüllt, bis die maximale Anzahl der Zeichen erreicht ist, die für diese Zeichenkette definiert ist.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiele:

Eingang	Ausgang definiert als	liefert
16#ABCDEFFE	STRING[2]	'FE'
	STRING[4]	'EFFE'
	STRING[6]	'CDEFFE'
	STRING[8]	'ABCDEFFE'
	STRING[10]	'00ABCDEFFE'
	STRING[12]	'0000ABCDEFFE'
	und so weiter ...	

☞

- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Weitere Informationen finden Sie in der Online-Hilfe im Thema: Aktualisierungsprobleme bei Datentyp STRING**

SPS-Typen **Verfügbarkeit von DWORD_TO_STRING (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
DWORD	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

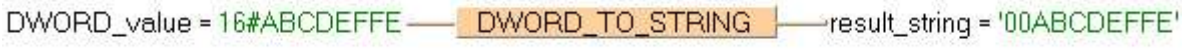
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Funktion verwendet werden.

	Class	Identifier	Type	Initial	Comment
0	VAR	DWORD_value	DWORD	0	example value: 16#ABCDEFFE
1	VAR	result_string	STRING[10]	''	result: '00ABCDEFFE'

Die Eingangsvariable **input_value** vom Datentyp BOOL wird mit folgendem Wert initialisiert: 16#ABCDEFFE. Die Ausgangsvariable **result_string** ist vom Datentyp STRING[10]. Sie kann maximal 10 Zeichen speichern. Statt der Variablen **input_value,** könnnen Sie im Rumpf eine Konstante direkt an den Eingang der Funktion schreiben.

Rumpf Der **input_value** vom Datentyp DWORD wird in einen STRING[10] umgewandelt. Der konvertierte Wert wird in **result_string** geschrieben. Mit der Variablen **input_value** = 16#ABCDEFFE steht im **result_string** '00ABCDEFFE'.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
result_string:=DWORD_TO_STRING(input_value);
```

Beispiel 2 Dieses Beispiel zeigt, wie man aus einem Eingangswert vom Datentyp DWORD einen STRING der Länge 10 erzeugt, in dem der führende Teilstring '16#' durch den String '0x' ersetzt wird.

Das Beispiel wird in KOP und in AWL programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POU header

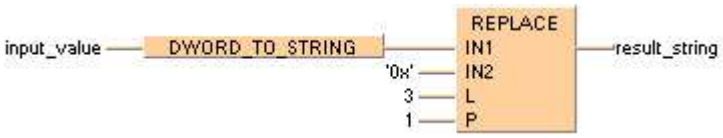
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	DWORD	16#12345678	example value
1	VAR	result_string	STRING[10]	''	result: here '0x12345678'

In diesem Beispiel werden eine Eingangsvariablen **input_value** vom Datentyp DWORD sowie eine Ausgangsvariable **result_string** vom Datentyp STRING[10] deklariert.

Body Zur Realisierung der geforderten Aufgabe wird der Funktion DWORD_TO_STRING zusätzlich die Standardfunktion REPLACE nachgeschaltet. Sie ersetzt einen Abschnitt in einer Zeichenfolge durch einen anderen.

Im Beispiel liegt am Eingang IN1 der REPLACE-Funktion der Ausgangsstring von DWORD_TO_STRING an. Am Eingang IN2 liegt der Ersatz-STRING als STRING-Konstante '0x' an. Am L-Eingang von REPLACE gibt die INT-Konstante 3 die Länge des zu ersetzenden STRING an. Der P-Eingang gibt die Position an, ab der ersetzt werden soll. In diesem Fall ist dies die INT-Zahl 1. Aus der Variablen **input_value** = 16#12345678 wird nach der Typwandlung und der REPLACE-Funktion der **result_string** = '0x12345678'.

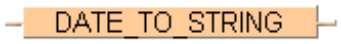
KOP



DATE_TO_STRING DATE in STRING

Erklärung DATE_TO_STRING wandelt einen Wert vom Datentyp DATE in einen Wert vom Datentyp STRING[10] um.

Der Wertebereich für das Eingangsdatum beträgt D#2001-01-01 bis D#2099-12-31.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DATE_TO_STRING (s. S. 1185)**

☞ **In der Ergebniszeichenfolge werden sämtliche Zeichenpositionen gefüllt.**

Datentypen

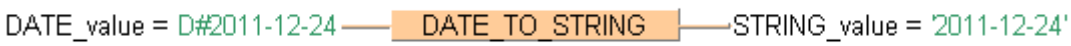
Datentyp	E/A	Funktion
DATE	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis STRING[10]

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DATE_value	DATE	D#2011-12-24
1	VAR	STRING_value	STRING[10]	''

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
STRING_value := DATE_TO_STRING(DATE_value);
```

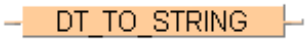
DT_TO_STRING

DATE_AND_TIME in STRING

Erklärung

DT_TO_STRING wandelt einen Wert vom Datentyp DATE_AND_TIME in einen Wert vom Datentyp STRING[19] um.

Der Wertebereich für das Eingangsdatum beträgt DT#2001-01-01-00:00:00 bis DT#2099-12-31-23:59:59.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von DT_TO_STRING (s. S. 1185)

☞ **In der Ergebniszeichenfolge werden sämtliche Zeichenpositionen gefüllt.**

Datentypen

Datentyp	E/A	Funktion
DATE_AND_TIME	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis STRING[19]

Beispiel

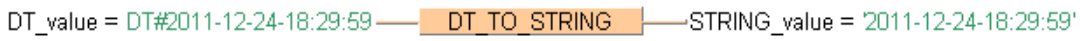
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DT_value	DATE_AND_TIME	DT#2011-12-24-18:29:59
1	VAR	STRING_value	STRING[19]	''

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

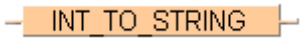
```
STRING_value := DT_TO_STRING(DT_value);
```

INT_TO_STRING

INTEGER in STRING

Erklärung Die Funktion INT_TO_STRING wandelt einen Wert vom Datentyp INT in einen Wert vom Datentyp STRING. Sie erzeugt eine rechtsbündige Ergebniszeichenkette in Hexadezimaldarstellung. Diese wird mit führenden Nullen aufgefüllt, bis die maximale Anzahl der Zeichen erreicht ist, die für diese Zeichenkette definiert ist.

Siehe auch: INT_TO_STRING_LEADING_ZEROS (s. S. 219)



Bedeutung:

Verwendete Funktion	String1 definiert als	Ergebnis
String1:=INT_TO_STRING(-12345)	STRING[1]	'5'
	STRING[2]	'45'
	STRING[3]	'345'
	STRING[4]	'2345'
	STRING[5]	'12345'
	STRING[6]	'-12345'
	STRING[7]	'␣-12345'
	STRING[8]	'␣␣-12345'
	und so weiter ...	

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INT_TO_STRING (s. S. 1193)**

☞
- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Weitere Informationen finden Sie in der Online-Hilfe im Thema: Aktualisierungsprobleme bei Datentyp STRING**

Datentypen

Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	INT_value	INT	-12345	example value
1	VAR	result_string	STRING[8]	''	result: here '-12345'

Die Eingangsvariable **input_value** vom Datentyp INT wird mit dem Wert -12345 initialisiert. Die Ausgangsvariable **result_string** ist vom Datentyp STRING[8]. Sie kann maximal 8 Zeichen speichern. Statt der Variablen **input_value**, können Sie im Rumpf eine Konstante direkt an den Eingang der Funktion schreiben.

Rumpf Der **input_value** vom Datentyp INT wird in einen STRING[8] umgewandelt. Der konvertierte Wert wird in **result_string** geschrieben. Mit der Variablen **input_value** = -12345 steht im **result_string** ' ␣␣-12345'.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
result_string := INT_TO_STRING(input_value);
```

Dieses Beispiel zeigt, wie man aus einem Eingangswert vom Datentyp INT einen STRING der Länge 2 mit rechtsbündiger Darstellung erzeugt.

POU header

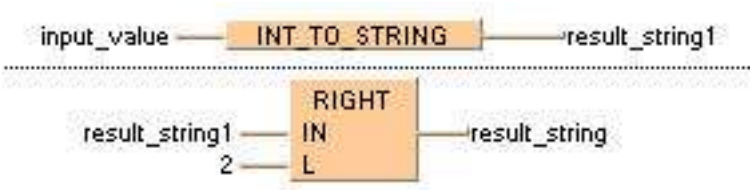
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	INT_value	INT	12	example value
1	VAR	result_string	STRING[2]	''	result: here '12'

In diesem Beispiel werden eine Eingangsvariablen Eingangswert vom Datentyp INT sowie eine Ausgangsvariable Ergebnisstring vom Datentyp STRING[2] deklariert.

Body Zur Realisierung der geforderten Aufgabe wird der Funktion INT_TO_STRING zusätzlich die Standardfunktion RIGHT (s. S. 260) nachgeschaltet. Sie erzeugt eine rechtsbündige Zeichenkette der Länge L.

Im Beispiel konvertiert die INT_TO_STRING-Funktion aus der Variablen input_value = 12 den Zwischenstring ' 12'. Die RIGHT-Funktion erzeugt danach den result_string '12'.

LD



ST

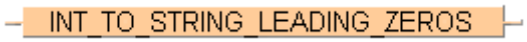
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
result_string:=RIGHT(IN:=INT_TO_STRING(input_value), L:=2);
```

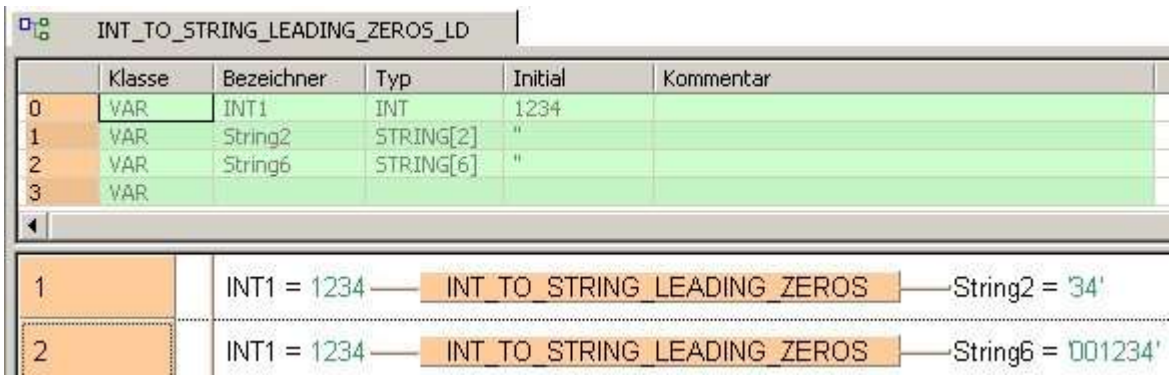
INT_TO_STRING_ LEADING_ZEROS

INTEGER in STRING

Erklärung Die Funktion INT_TO_STRING_LEADING_ZEROS wandelt einen Wert vom Datentyp INT (positiver Wert) in einen Wert vom Datentyp STRING. Sie erzeugt eine rechtsbündige Ergebniszeichenkette in Hexadezimaldarstellung. Diese wird mit führenden Nullen aufgefüllt, bis die maximale Anzahl der Zeichen erreicht ist, die für diese Zeichenkette definiert ist.



Beispiel:



Bedeutung:

Verwendete Funktion	String1 definiert als	Ergebnis
String1:=INT_TO_STRING(25)	STRING[1]	'5'
	STRING[2]	'25'
	STRING[3]	'025'
	STRING[4]	'0025'
	STRING[5]	'00025'
	STRING[6]	'000025'
	STRING[7]	'0000025'
	STRING[8]	'00000025'
	und so weiter ...	

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INT_TO_STRING LEADING ZEROS (s. S. 1193)**

☞
- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Weitere Informationen finden Sie in der Online-Hilfe im Thema: Aktualisierungsprobleme bei Datentyp STRING**

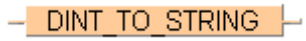
Datentypen

Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis

DINT_TO_STRING

DOUBLE INTEGER in STRING

Erklärung Die Funktion DINT_TO_STRING wandelt einen Wert vom Datentyp DINT in einen Wert vom Datentyp STRING. Sie erzeugt eine rechtsbündige Ergebniszeichenkette in Hexadezimaldarstellung. Diese wird mit führenden Nullen aufgefüllt, bis die maximale Anzahl der Zeichen erreicht ist, die für diese Zeichenkette definiert ist.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Siehe auch: DINT_TO_STRING_LEADING_ZEROS (s. S. 222)

Beispiel:

Verwendete Funktion	String1 definiert als	Ergebnis
String1:=DINT_TO_STRING(-12345678)	STRING[2]	'78'
	STRING[4]	'5678'
	STRING[6]	'345678'
	STRING[8]	'12345678'
	STRING[10]	'␣-12345678'
	STRING[12]	'␣␣␣-12345678'
	und so weiter ...	

SPS-Typen **Verfügbarkeit von DINT_TO_STING (s. S. 1185)**

- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Weitere Informationen finden Sie in der Online-Hilfe im Thema: Aktualisierungsprobleme bei Datentyp STRING**

Datentypen

Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_string	DINT	12345678	example value
1	VAR	result_string	STRING[11]	''	result: here '12345678'

Die Eingangsvariable **input_string** vom Datentyp DINT wird mit dem Wert 12345678 initialisiert. Die Ausgangsvariable **result_string** ist vom Datentyp STRING[11]. Sie kann maximal 11 Zeichen speichern. Statt der Variablen **input_string**, können Sie im Rumpf eine Konstante direkt an den Eingang der Funktion schreiben.

Rumpf Der **input_string** vom Datentyp DINT wird in einen STRING[11] umgewandelt. Der konvertierte Wert wird in **result_string** geschrieben. Mit dem **input_string** = 12345678 steht im **result_string** '␣␣␣12345678'.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
result_string:=DINT_TO_STRING(input_value);
```

Beispiel 2 Dieses Beispiel zeigt, wie man aus einem Eingangswert vom Datentyp DINT einen STRING der Länge 14 erzeugt, welcher eine DINT-Zahlendarstellung mit 1000er-Punkten beinhaltet.

Das Beispiel wird in KOP und in AWL programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POU header

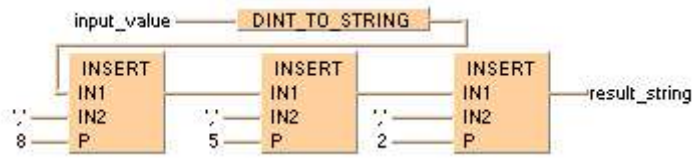
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_string	DINT	1234567890	example value
1	VAR	result_string	STRING[14]	''	result: here '1,234,567,890'

In diesem Beispiel werden eine Eingangsvariable **input_string** vom Datentyp DINT sowie eine Ausgangsvariable **result_string** vom Datentyp STRING[14] deklariert.

Body Zur Realisierung der geforderten Aufgabe werden der Funktion DINT_TO_STRING zusätzlich drei Standardfunktionen INSERT nachgeschaltet. Jede INSERT-Funktion fügt die am Eingang IN2 anliegende Zeichenfolge in die Zeichenfolge am Eingang IN1 ein. Die Stelle, an der die Zeichenfolge eingefügt werden soll, wird durch den INT-Wert am Eingang P festgelegt.

Im Beispiel wird allen drei INSERT-Funktionen am Eingang **IN2** der 1000er-Punkt als STRING-Konstante '.' zugeordnet. Die richtige Position des 1000er Punktes wird an den jeweiligen P-Eingängen als INT-Konstante festgelegt. Aus der Variablen input_string = 1234567890 wird nach der Typwandlung und den drei INSERT-Funktionen der **result_string** '1.234.567.890'.

LD

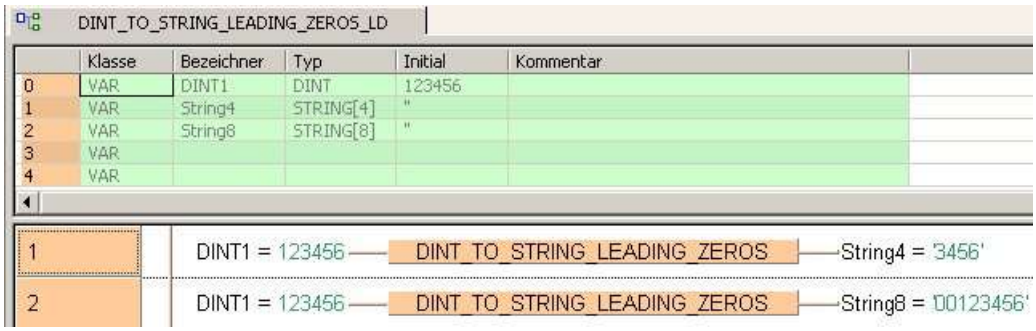


DINT_TO_STRING_ LEADING_ZEROS

DOUBLE INTEGER in STRING

Erklärung Die Funktion DINT_TO_STRING_LEADING_ZEROS wandelt einen Wert vom Datentyp DINT (positiver Wert) in einen Wert vom Datentyp STRING. Sie erzeugt eine rechtsbündige Ergebniszeichenkette in Hexadezimaldarstellung. Diese wird mit führenden Nullen aufgefüllt, bis die maximale Anzahl der Zeichen erreicht ist, die für diese Zeichenkette definiert ist.

Beispiel:



Bedeutung:

Verwendete Funktion	String1 definiert als	Ergebnis
String1:=DINT_TO_STRING(12345678)	STRING[2]	'78'
	STRING[4]	'5678'
	STRING[6]	'345678'
	STRING[8]	'12345678'
	STRING[10]	'0012345678'
	STRING[12]	'000012345678'
	und so weiter ...	

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DINT_TO_STRING_LEADING_ZEROS (s. S. 1185)**

☞

- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Weitere Informationen finden Sie in der Online-Hilfe im Thema: Aktualisierungsprobleme bei Datentyp STRING**

Datentypen

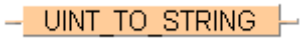
Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis

UINT_TO_STRING

vorzeichenloser INTEGER in STRING

Erklärung UINT_TO_STRING wandelt einen Wert vom Datentyp Unsigned INTEGER in einen Wert vom Datentyp STRING um.

Das Ergebnis wird als rechtsbündige Zeichenfolge in Dezimaldarstellung generiert. Diese Zeichenfolge wird mit führenden Leerzeichen aufgefüllt, bis die vordefinierte maximale Anzahl von Zeichen erreicht ist.



Siehe auch: UINT_TO_STRING_LEADING_ZEROS (s. S. 225)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UINT_TO_STRING (s. S. 1198)**

☞ **Stimmen die jeweiligen Bereiche der Eingangs- und Ausgangswerte nicht überein, wird das Ergebnis nicht angegeben.**

Datentypen

Datentyp	E/A	Funktion
UINT	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	49152
1	VAR	STRING_value	STRING[8]	''

KOP

ST

```
STRING_value := UINT_TO_STRING(UINT_value);
```

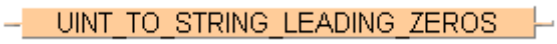
UINT_TO_STRING_LEADING_ZEROS

vorzeichenloser INTEGER in STRING

Erklärung

UINT_TO_STRING_LEADING_ZEROS wandelt einen Wert vom Datentyp Unsigned INTEGER in einen Wert vom Datentyp STRING um.

Das Ergebnis wird als rechtsbündige Zeichenfolge in Dezimaldarstellung generiert. Diese Zeichenfolge wird mit führenden Leerzeichen aufgefüllt, bis die vordefinierte maximale Anzahl von Zeichen erreicht ist.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von UINT_TO_STRING_LEADING_ZEROS (s. S. 1198)

☞ **Stimmen die jeweiligen Bereiche der Eingangs- und Ausgangswerte nicht überein, wird das Ergebnis nicht angegeben.**

Datentypen

Datentyp	E/A	Funktion
UINT	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis

Beispiel

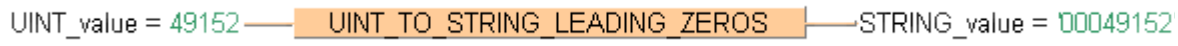
In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	49152
1	VAR	STRING_value	STRING[8]	''

KOP



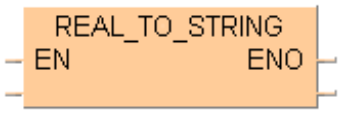
ST

```
STRING_value:=UINT_TO_STRING_LEADING_ZEROS(UINT_value);
```

REAL_TO_STRING

REAL in STRING

Erklärung Die Funktion REAL_TO_STRING wandelt einen Wert vom Datentyp REAL in einen Wert vom Datentyp STRING[15] mit je 7 Vor- und 7 Nachkommastellen. Der Ergebnisstring wird im Bereich von '-999999.0000000' bis '9999999.0000000' dargestellt. Im positiven Bereich wird das Pluszeichen weggelassen. Führende Nullen werden mit Leerzeichen aufgefüllt (z.B. wird aus -12,0 der STRING '␣␣-12,0').



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von REAL_TO_STRING (s. S. 1196)**

☞
- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Weitere Informationen finden Sie in der Online-Hilfe im Thema: Aktualisierungsprobleme bei Datentyp STRING**

Datentypen

Datentyp	E/A	Funktion
REAL	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_string	REAL	-123.4560166	example value
1	VAR	result_string	STRING[15]	''	result: here '-123.4560166'

Die Eingangsvariable **input_string** vom Datentyp REAL wird mit dem Wert -123,4560166 initialisiert. Die Ausgangsvariable **result_string** ist vom Datentyp STRING[15]. Sie kann maximal 15 Zeichen speichern. Statt der Variablen **input_string**, können Sie im Rumpf eine Konstante direkt an den Eingang der Funktion schreiben.

Rumpf Der **input_string** vom Datentyp REAL wird in einen STRING[15] umgewandelt. Der konvertierte Wert wird in **result_string** geschrieben. Mit dem **input_string** = 123,4560166 steht im **result_string** ' -123,4560165'.

KOP

Beispiel 2 Dieses Beispiel zeigt, wie man aus einem Eingangswert vom Datentyp REAL einen STRING der Länge 7 mit 4 Vor- und 2 Nachkommastellen erzeugt.
Das Beispiel wird in KOP und in AWL programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POU header

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_string	REAL	-123.4560166	example value
1	VAR	result_string	STRING[7]	''	result: here '-123.45'

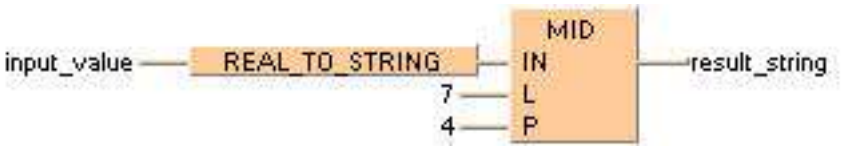
In diesem Beispiel werden eine Eingangsvariable **input_string** vom Datentyp REAL sowie eine Ausgangsvariable **result_string** vom Datentyp STRING[7] deklariert.

Body

Zur Realisierung der geforderten Aufgabe wird der Funktion REAL_TO_STRING zusätzlich die Standardfunktion MID nachgeschaltet. Sie sorgt für einen Mittelabschnitt der Zeichenfolge ab Position P (INT-Wert) mit L (INT-Wert) Zeichen.

Im Beispiel wird am L-Eingang von MID die INT-Konstante 7 angelegt, welche die Länge des Ergebnisstrings festlegt. Die INT-Konstante 4 am Eingang P bestimmt die Position, ab der der Mittelabschnitt beginnt. Aus der Variablen **input_string** = -123.4560166 wird nach der Typwandlung der STRING ' -123.4560166'. Die MID-Funktion schneidet den STRING an der Position 4 ab und gibt den **result_string** '-123.45' aus.

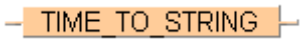
LD



TIME_TO_STRING

TIME into STRING

Erklärung Die Funktion TIME_TO_STRING wandelt einen Wert vom Datentyp TIME in einen Wert vom Datentyp STRING[20]. Entsprechend der Darstellung gemäß IEC-1131 wird der Ergebnisstring mit kurzem Zeitpräfix und ohne Unterstriche dargestellt. Mögliche Werte für den Ergebnisstring liegen im Bereich von 'T#000d00h00m00s000ms' bis 'T#248d13h13m56s470ms'.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞ **Bei Kleinsteuerungen wie FP-E, FP0, FP1 oder FP-M müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**

SPS-Typen **Verfügbarkeit von TIME_TO_STING (s. S. 1197)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

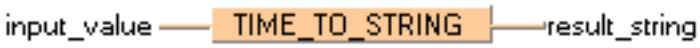
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	TIME	T#1h30m45s	example value
1	VAR	result_string	STRING[20]	''	result: here 'T#000d01h30m45s000ms'

Die Eingangsvariable **input_value** vom Datentyp TIME wird mit dem Wert T#1h30m45s initialisiert. Die Ausgangsvariable **result_string** ist vom Datentyp STRING[20]. Sie kann maximal 20 Zeichen speichern. Statt der Variablen **input_value**, können Sie im Rumpf eine Konstante direkt an den Eingang der Funktion schreiben.

Rumpf Der **input_value** vom Datentyp TIME wird in einen STRING[20] umgewandelt. Der konvertierte Wert wird in **result_string** geschrieben. Mit der Variablen **input_value** = T#1h30m45s steht im **result_string** 'T#000d01h30m45s000ms'.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
result_string:=TIME_TO_STRING(input_value);
```

Beispiel 2

Dieses Beispiel zeigt, wie man aus einem Eingangswert vom Datentyp TIME einen Zeit-STRING der Länge 9 im Format 'xxhxxmxxs' erzeugt (es sollen nur Stunden, Minuten und Sekunden ausgegeben werden).

Das Beispiel wird in KOP und in AWL programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POU header

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_value	TIME	T#1h30m45s	example value
1	VAR	result_string	STRING[9]	''	result: here '01h30m45s'

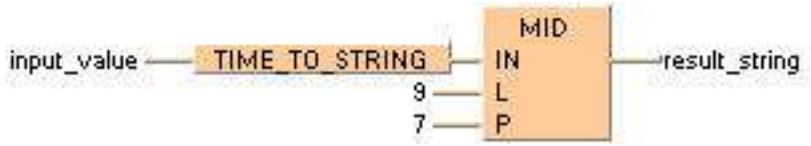
In diesem Beispiel werden eine Eingangsvariable **input_value** vom Datentyp TIME sowie eine Ausgangsvariable **result_string** vom Datentyp STRING[9] deklariert.

Body

Zur Realisierung der geforderten Aufgabe wird der Funktion TIME_TO_STRING zusätzlich die Standardfunktion MID nachgeschaltet. Sie sorgt für einen Mittelabschnitt der Zeichenfolge ab Position P (INT-Wert) mit L (INT-Wert) Zeichen.

Im Beispiel wird am L-Eingang von MID die INT-Konstante 9 angelegt, welche die Länge des Ergebnisstrings festlegt. Die INT-Konstante 7 am Eingang **P** bestimmt die Position, ab der der Mittelabschnitt beginnt. Aus der Variablen **input_value** = T#1h30m45s wird nach der Typwandlung der STRING 'T#000d01h30m45s000ms'. Die MID-Funktion schneidet den STRING an der Position 7 ab und gibt den **result_string** '01h30m45s' aus.

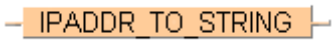
LD



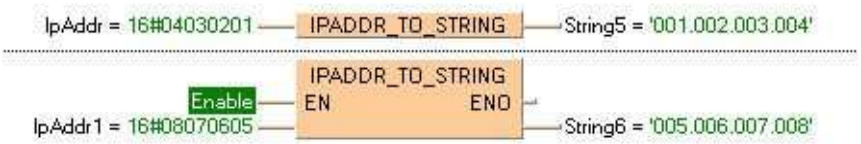
IPADDR_TO_STRING

IP-Adresse in STRING

Erklärung Diese Funktion wandelt eine binäre IP-Adresse vom Datentyp DWORD in eine Zeichenkette im IP-Adressformat um.



Beispiel:



Zulässiges Format:

'Oktett1.Oktettt2.Oktettt3.Oktett4' also z.B.: '192.168.206.004'

Zulässige Zeichen:

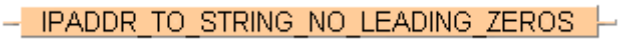
Oktetts 1-4	Dezimalzahlen von "0“ - "9“, maximal 3 Stellen, ohne führende Nullen im Bereich von 0-255

Die Wandlung erfolgt so, dass das höchste Byte der ET-LAN-Adresse das vierte Oktett darstellt und das niedrigste Byte der IP-Adresse das erste Oktett darstellt. Das Format der IP-Adresse entspricht dem Standardformat wie es z.B. von "Standard Socket Application Interfaces" verwendet wird.

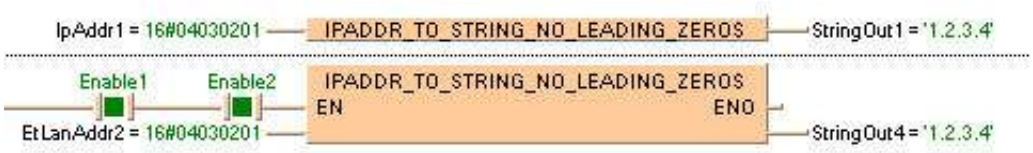
IPADDR_TO_STRING_ NO_LEADING_ZEROS

IP-Adresse in STRING

Erklärung Diese Funktion wandelt eine binäre IP-Adresse vom Datentyp DWORD in eine Zeichenkette im IP-Adressformat um.



Beispiel:



Zulässiges Format:

'Oktett1.Oktettt2.Oktettt3.Oktett4' also z.B.: '192.168.206.4'

Zulässige Zeichen:

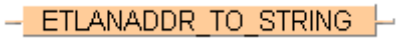
Oktetts 1-4	Dezimalzahlen von "0“ - "9“, maximal 3 Stellen, ohne führende Nullen im Bereich von 0-255

Die Wandlung erfolgt so, dass das höchste Byte der ET-LAN-Adresse das vierte Oktett darstellt und das niedrigste Byte der IP-Adresse das erste Oktett darstellt. Das Format der IP-Adresse entspricht dem Standardformat wie es z.B. von "Standard Socket Application Interfaces" verwendet wird.

ETLANADDR_TO_STRING

ETLAN Addresse in STRING

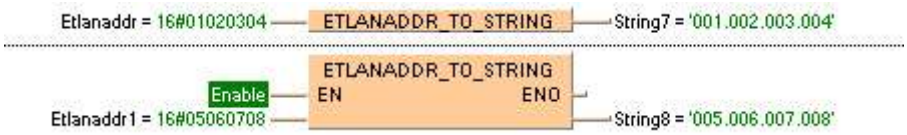
Erklärung Diese Funktion wandelt eine binäre ETLAN-Adresse vom Datentyp DWORD in eine Zeichenkette im ETLAN-Adressformat um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Siehe auch: ETLANADDR_TO_STRING_NO_LEADING_ZEROS (s. S. 232)

Beispiel:



Zulässiges Format:

'Oktett1.Oktettt2.Oktettt3.Oktett4' also z.B.: '192.168.206.004'

Zulässige Zeichen:

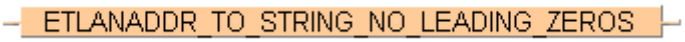
Oktetts 1-4	Dezimalzahlen von "0“-"9“, maximal 3 Stellen, mit führenden Nullen im Bereich von 0-255

Die Wandlung erfolgt so, dass das höchste Byte der ET-LAN-Adresse das erste Oktett darstellt und das niedrigste Byte der IP-Adresse das vierte Oktett darstellt. Dieses Format der ET-LAN-Adresse wird z.B. von den ET-LAN-Steckmodulen der FP-Serien verwendet.

ETLANADDR_TO_STRING _NO_LEADING_ZEROS

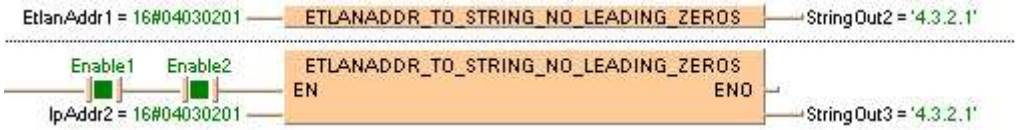
ETLAN Addresse in STRING

Erklärung Diese Funktion wandelt eine binäre ETLAN-Adresse vom Datentyp DWORD in eine Zeichenkette im ETLAN-Adressformat um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiel:



Zulässiges Format:

'Oktett1.Oktettt2.Oktettt3.Oktett4' also z.B.: '192.168.206.4'

Zulässige Zeichen:

Oktetts 1-4	Dezimalzahlen von "0“ - "9“, maximal 3 Stellen, ohne führende Nullen im Bereich von 0-255

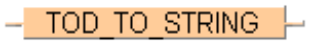
Die Wandlung erfolgt so, dass das höchste Byte der ET-LAN-Adresse das erste Oktett darstellt und das niedrigste Byte der IP-Adresse das vierte Oktett darstellt. Dieses Format der ET-LAN-Adresse wird z.B. von den ET-LAN-Steckmodulen der FP-Serien verwendet.

TOD_TO_STRING

TIME_OF_DAY in STRING

Erklärung TOD_TO_STRING wandelt einen Wert vom Datentyp TIME_OF_DAY in einen Wert vom Datentyp STRING[8] um.

Der Wertebereich für die Eingangsuhrzeit beträgt TOD#00:00:00 bis TOD#23:59:59.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von TOD_TO_STRING (s. S. 1198)**

☞ **In der Ergebniszeichenfolge werden sämtliche Zeichenpositionen gefüllt.**

Datentypen

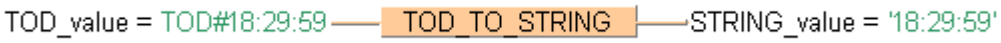
Datentyp	E/A	Funktion
TIME_OF_DAY	Eingang	Eingangsdatentyp
STRING	Ausgang	Umwandlungsergebnis STRING[8]

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	TOD_value	TIME_OF_DAY	TOD#18:29:59
1	VAR	STRING_value	STRING[19]	''

KOP



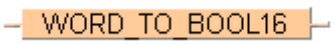
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
STRING_value := TOD_TO_STRING(TOD_value);
```

WORD_TO_BOOL16

WORD in BOOL16

Erklärung Diese Funktion kopiert Daten vom Typ WORD am Eingang auf ein Array mit 16 Elementen vom Datentyp BOOL am Ausgang.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_TO_BOOL16 (s. S. 1199)**

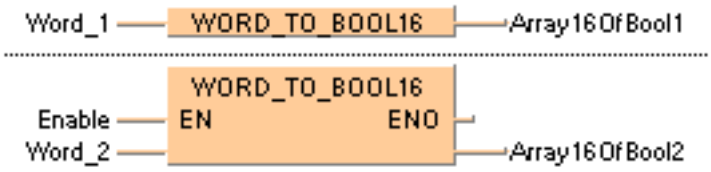
Datentypen

Datentyp	Kommentar
WORD	Eingangsvariable
ARRAY of BOOL	ARRAY mit 16 Elementen

POE-Kopf:

	Klasse	Bezeichner	Typ	Initial
0	VAR	Array16OfBool1	ARRAY [0..15] OF BOOL	[16(FALSE)]
1	VAR	Array16OfBool2	ARRAY [0..15] OF BOOL	[16(FALSE)]
2	VAR	Enable	BOOL	FALSE
3	VAR	Word_1	WORD	0
4	VAR	Word_2	WORD	0

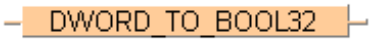
Beispiel:



DWORD_TO_BOOL32

DOUBLE WORD in BOOL32

Erklärung Diese Funktion kopiert Daten vom Typ DWORD am Eingang auf ein Array mit 32 Elementen vom Datentyp BOOL am Ausgang.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DWORD_TO_BOOL32 (s. S. 1185)**

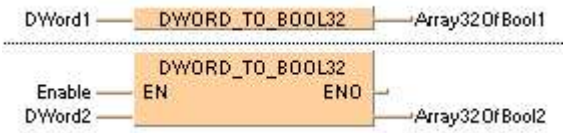
Datentypen

Datentyp	Kommentar
DWORD	Eingangsvariable
ARRAY of BOOL	ARRAY mit 32 Elementen

POE-Kopf:

	Klasse	Bezeichner	Typ	Initial
0	VAR	Enable	BOOL	FALSE
1	VAR	Array32OfBool1	ARRAY [0..31 OF BOOL	[FALSE]
2	VAR	Array32OfBool2	ARRAY [0..31 OF BOOL	
3	VAR	DWord1	DWORD	0
4	VAR	DWord2	DWORD	0

Beispiel mit und ohne EN/ENO:

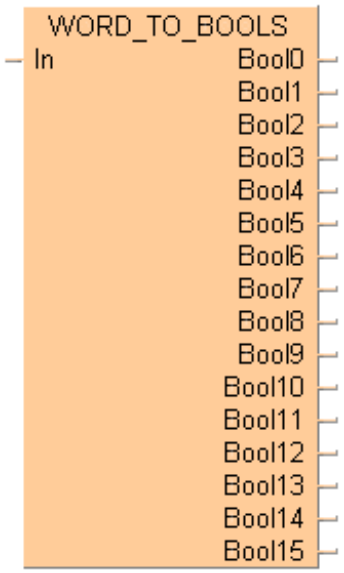


WORD_TO_BOOLS

WORD in 16 Variablen vom Typ BOOL

Erklärung Diese Funktion wandelt einen Wert vom Datentyp WORD bitweise in 16 Werte vom Datentyp BOOL um.

Symbol:



Die Ausgänge Bool0 bis Bool15 müssen im KOP und im FBD nicht verbunden werden bzw. im ST-Editor in der formalen Parameterliste nicht explizit verwendet werden. Nur für die wirklich verbundenen bzw. benutzten Ausgänge wird auch Programmcode generiert.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von WORD_TO_BOOLS (s. S. 1199)**

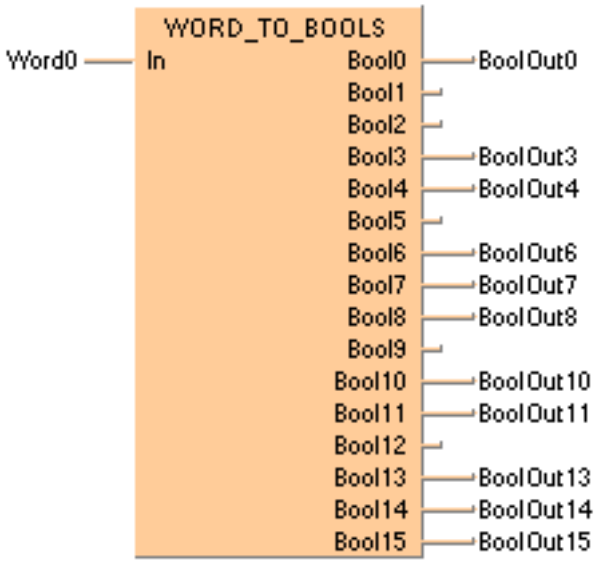
Datentypen

Variable	Datentyp	Funktion
In	WORD	Eingangsvariable
BOOL0 ... BOOL15	BOOL	16 Ausgangsvariablen vom Datentyp BOOL

POE-Kopf

	Klasse	Bezeichner	Typ	Initial
0	VAR	Word0	WORD	0
1	VAR	Bool0	BOOL	FALSE
2	VAR	Bool1	BOOL	FALSE
3	VAR	Bool2	BOOL	FALSE
4	VAR	Bool3	BOOL	FALSE
5	VAR	Bool4	BOOL	FALSE
6	VAR	Bool5	BOOL	FALSE
7	VAR	Bool6	BOOL	FALSE
8	VAR	Bool7	BOOL	FALSE
9	VAR	Bool8	BOOL	FALSE
10	VAR	Bool9	BOOL	FALSE
11	VAR	Bool10	BOOL	FALSE
12	VAR	Bool11	BOOL	FALSE
13	VAR	Bool12	BOOL	FALSE
14	VAR	Bool13	BOOL	FALSE
15	VAR	Bool14	BOOL	FALSE
16	VAR	Bool15	BOOL	FALSE

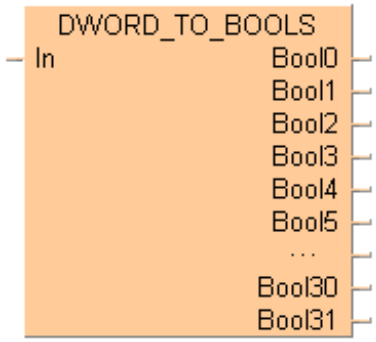
Rumpf


WORD_TO_BOOLS
Word0
In
Bool0
Bool1
Bool2
Bool3
Bool4
Bool5
Bool6
Bool7
Bool8
Bool9
Bool10
Bool11
Bool12
Bool13
Bool14
Bool15
BoolOut0
BoolOut3
BoolOut4
BoolOut6
BoolOut7
BoolOut8
BoolOut10
BoolOut11
BoolOut13
BoolOut14
BoolOut15

DWORD_TO_BOOLS

DOUBLE WORD in 32 Variablen vom Typ BOOL

Erklärung Diese Funktion wandelt einen Wert vom Datentyp DWORD bitweise in 32 Werte vom Datentyp BOOL um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Die Ausgänge Bool0 bis Bool31 müssen im KOP und im FBD nicht verbunden werden bzw. im ST-Editor in der formalen Parameterliste nicht explizit verwendet werden. Nur für die wirklich verbundenen bzw. benutzten Ausgänge wird auch Programmcode generiert.

SPS-Typen **Verfügbarkeit von DWORD_TO_BOOLS (s. S. 1185)**

Datentypen

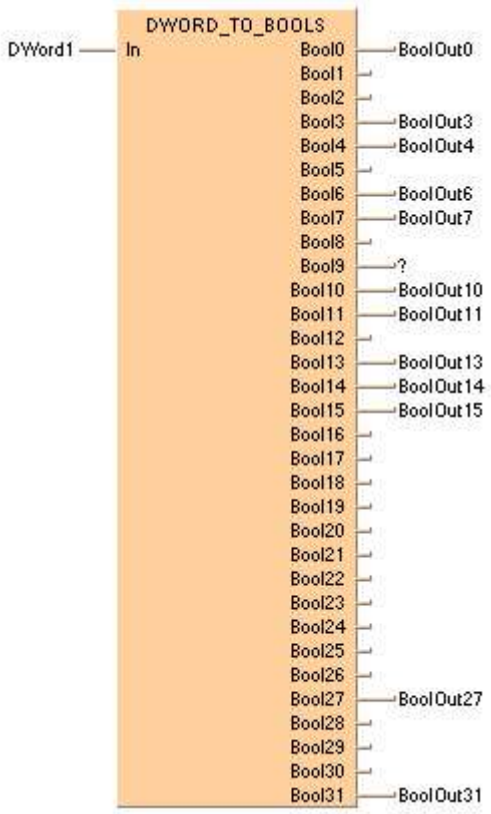
Variable	Datentyp	Funktion
In	DWORD	Eingangsvariable
BOOL0 ... BOOL31	BOOL	32 Ausgangsvariablen vom Datentyp BOOL

POE-Kopf:

	Klasse	Bezeichner	Typ	Initial
0	VAR	dWord1	DWORD	0
1	VAR	Bool0	BOOL	FALSE
2	VAR	Bool1	BOOL	FALSE
3	VAR	Bool2	BOOL	FALSE
4	VAR	Bool3	BOOL	FALSE
5	VAR	Bool4	BOOL	FALSE
6	VAR	Bool5	BOOL	FALSE
7	VAR	Bool6	BOOL	FALSE
8	VAR	Bool7	BOOL	FALSE

usw. bis Bool31

Rumpf:

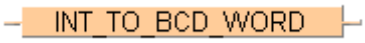

DWORD_TO_BOOLS
DWord1
In
Bool0
BoolOut0
Bool1
Bool2
Bool3
BoolOut3
Bool4
BoolOut4
Bool5
Bool6
BoolOut6
Bool7
BoolOut7
Bool8
Bool9
?
Bool10
BoolOut10
Bool11
BoolOut11
Bool12
Bool13
BoolOut13
Bool14
BoolOut14
Bool15
BoolOut15
Bool16
Bool17
Bool18
Bool19
Bool20
Bool21
Bool22
Bool23
Bool24
Bool25
Bool26
Bool27
BoolOut27
Bool28
Bool29
Bool30
Bool31
BoolOut31

INT_TO_BCD

INTEGER in BCD

Erklärung

INT_TO_BCD_WORD konvertiert einen Binärwert vom Datentyp INTEGER in einen BCD-Wert (binär codierte Dezimalzahl) vom Datentyp WORD, damit BCD-Werte im Wortformat ausgegeben werden können.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von INT_TO_BCD_WORD (s. S. 1193)

☞ **Die Ausgangsvariable ist vom Datentyp WORD und beinhaltet 16 Bit. Folglich ist der Wert für die Eingangsvariable auf 4 Stellen beschränkt und muss zwischen 0 und 9999 liegen.**

Datentypen

Datentyp	E/A	Funktion
INT	Eingang	Eingangsdatentyp
BCD_WORD	Ausgang	Umwandlungsergebnis

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

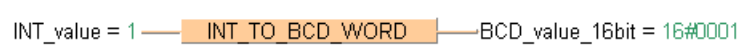
	Klasse	Bezeichner	Typ	Initial
0	VAR	BCD_value_16bit	WORD	0
1	VAR	INT_value	INT	1

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf

INT_value vom Datentyp INTEGER wird in einen BCD-Wert vom Datentyp WORD umgewandelt. Der umgewandelte Wert wird in **BCD_value_16bit** geschrieben.

KOP



ST

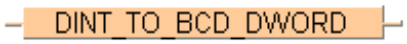
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
BCD_value_16bit:=INT_TO_BCD_WORD(INT_value);
```

DINT_TO_BCD

DOUBLE INTEGER in BCD

Erklärung DINT_TO_BCD_DWORD wandelt einen Wert vom Datentyp DOUBLE INTEGER in einen BCD-Wert vom Datentyp DOUBLE WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DINT_TO_BCD_DWORD (s. S. 1185)**

☞ **Der Wert der Eingangsvariable muss zwischen 0 und 999.999.999 liegen.**

Datentypen

Datentyp	E/A	Funktion
DINT	Eingang	Eingangsdatentyp
BCD_DWORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

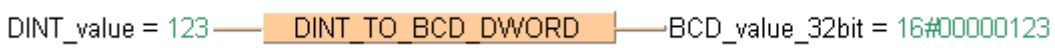
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	BCD_value_32bit	DWORD	0
1	VAR	DINT_value	DINT	0

In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf **DINT_value** vom Datentyp DOUBLE INTEGER wird in einen BCD-Wert vom Datentyp DOUBLE WORD umgewandelt. Der umgewandelte Wert wird in **BCD_value_32bit** geschrieben.

KOP



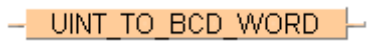
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
BCD_value_32bit:=DINT_TO_BCD_DWORD(DINT_value);
```

UINT_TO_BCD_WORD

vorzeichenloser INTEGER in BCD-Wert vom Datentyp WORD

Erklärung UINT_TO_BCD_WORD wandelt einen Wert vom Datentyp Unsigned INTEGER in einen BCD-Wert vom Datentyp WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UINT_TO_BCD_WORD (s. S. 1198)**

Datentypen

Datentyp	E/A	Funktion
UINT	Eingang	Eingangsdatentyp
BCD_WORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UINT_value	UINT	1270
1	VAR	BCD_value_16bit	WORD	16#0000

KOP

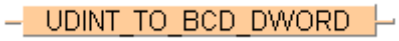
ST

```
BCD_value_16bit:=UINT_TO_BCD_WORD(UINT_value);
```

UDINT_TO_BCD_DWORD

vorzeichenloser DOUBLE INTEGER in BCD DOUBLE WORD

Erklärung UDINT_TO_BCD_DWORD wandelt einen Wert vom Datentyp Unsigned DOUBLE INTEGER in einen BCD-Wert vom Datentyp DOUBLE WORD um.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von UDINT_TO_BCD_DWORD (s. S. 1198)**

Datentypen

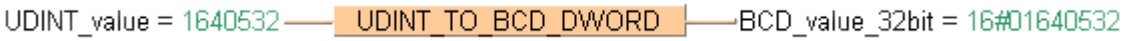
Datentyp	E/A	Funktion
UDINT	Eingang	Eingangsdatentyp
BCD_DWORD	Ausgang	Umwandlungsergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	UDINT_value	UDINT	16#190854
1	VAR	BCD_value_32bit	DWORD	

KOP



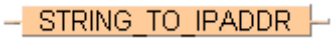
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
BCD_value_32bit := UDINT_TO_BCD_DWORD(UDINT_value);
```

STRING_TO_IPADDR

STRING in IP-Adresse

Erklärung Diese Funktion wandelt eine Zeichenkette im IP-Adressformat in einen Wert vom Datentyp DWORD um.

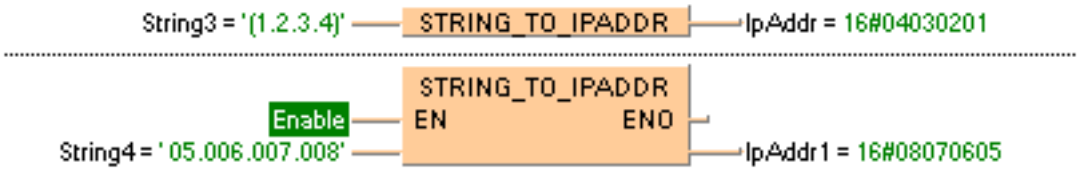
Symbol: 

Dabei wird die anliegende Zeichenkette zunächst in einen Wert vom Datentyp STRING[32] gewandelt. Anschließend wird dieser in einem Unterprogramm mit ca. 330 Schritten, welches gemeinsam für die Funktionen STRING_TO_IPADDR und STRING_TO_ETLANADDR verwendet wird, in einen Wert vom Datentyp DWORD gewandelt.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Siehe auch: STRING_TO_IPADDR_STEPSAVER (s. S. 245)

Beispiel:



Zulässiges Format:

´[Leerzeichen]Oktett1.Oktett2.Oktett3.Oktett4[Leerzeichen]´ also z.B: ´ [192.168.206.4] ´

Zulässige Zeichen:

Leerzeichen	Alle Zeichen außer Dezimalzahlen
Oktetts 1-4	Dezimalzahlen von "0“ - "9“, maximal 3 Stellen, mit oder ohne führende Nullen im Bereich von 0-255

☞

- **Die Analyse endet mit der ersten Nicht-Dezimalzahl nach dem 4.Oktett oder bei einem Formatfehler.**
- **Wenn das Format fehlerhaft ist, wird das Ergebnis 0.**
- **Die Wandlung erfolgt so, dass das erste Oktett das niedrigste Byte der IP-Adresse darstellt und das vierte Oktett das höchste Byte der IP-Adresse darstellt. Dies entspricht dem Standardformat wie es z.B. von "Standard Socket Application Interfaces" verwendet wird.**

SPS-Typen **Verfügbarkeit von STRING_TO_IPADDR (s. S. 1197)**

Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
DWORD	Ausgangsvariable

STRING_TO_IPADDR _STEPSAVER

STRING (IP-Adressen-Format 00a.0bb.0cc.ddd) in DWORD

Erklärung Diese Funktion wandelt eine Zeichenkette im IP-Adressenformat in einen Wert vom Datentyp DWORD um.

Symbol: 

In ca. 50 Schritten des generierten Inline-Codes für diese Funktion wird der Befehl F76_A2BIN (s. S. 616) verwendet. Dieser Befehl erwartet, dass jedes Oktet aus 3 Zeichen mit führenden Leerzeichen besteht. Ansonsten liefert die Steuerung einen Operationsfehler.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiel:

Zulässiges Format:

´Oktett1.Oktett2.Oktett3.Oktett4[Leerzeichen]´ also z.B: ´ [192.168.206.4] ´

Zulässige Zeichen:

Oktetts 1-4	Dezimalzahlen von "0“ - "9“, maximal 3 Stellen, mit oder ohne führende Nullen im Bereich von 0-255

☞

- **Wenn das Format fehlerhaft ist, wird das Ergebnis 0.**
- **Die Wandlung erfolgt so, dass das erste Oktett das niedrigste Byte der IP-Adresse darstellt und das vierte Oktett das höchste Byte der IP-Adresse darstellt. Dies entspricht dem Standardformat wie es z.B. von "Standard Socket Application Interfaces" verwendet wird.**

SPS-Typen **Verfügbarkeit von STRING_TO_IPADDR STEPSAVER (s. S. 1197)**

Datentypen

Datentyp	Kommentar
STRING	Eingangsvariable
DWORD	Ausgangsvariable

STRING_TO_ETLAN ADDR

STRING in ETLAN-Adresse

Erklärung

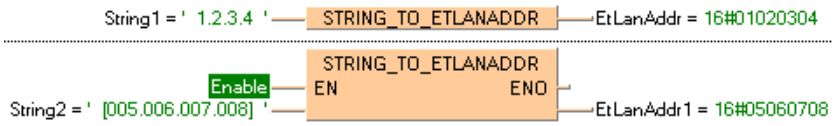
Diese Funktion wandelt eine Zeichenkette im IP-Adressformat in einen Wert vom Datentyp DWORD um.

Dabei wird die anliegende Zeichenkette zunächst in einen Wert vom Datentyp STRING[32] gewandelt. Anschließend wird dieser in einem Unterprogramm mit ca. 330 Schritten, welches gemeinsam für die Funktionen STRING_TO_IPADDR und STRING_TO_ETLANADDR verwendet wird, in einen Wert vom Datentyp DWORD gewandelt.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Siehe auch: STRING_TO_ETLANADDR_STEPSAVER

Beispiel:



Zulässiges Format:

´[Leerzeichen]Oktett1.Oktett2.Oktett3.Oktett4[Leerzeichen]´ also z.B: ´ [192.168.206.4] ´

Zulässige Zeichen:

Leerzeichen	Alle Zeichen außer Dezimalzahlen
Oktetts 1-4	Dezimalzahlen von "0“ - "9“, maximal 3 Stellen, mit oder ohne führende Nullen im Bereich von 0-255

☞

- **Die Analyse endet mit der ersten Nicht-Dezimalzahl nach dem 4.Oktett oder bei einem Formatfehler.**
- **Wenn das Format fehlerhaft ist, wird das Ergebnis 0.**
- **Die Wandlung erfolgt so, dass das höchste Byte der ET-LAN-Adresse das erste Oktett darstellt und das niedrigste Byte der IP-Adresse das vierte Oktett darstellt. Dieses Format der ET-LAN-Adresse wird z.B. von den ET-LAN-Steckmodulen der FP-Serien verwendet.**

STRING_TO_ETLAN ADDR_STEPSAVER

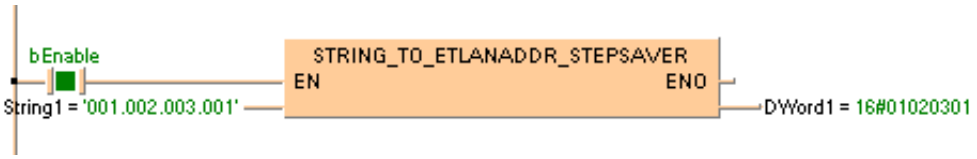
STRING (IP-Adressen-Format 00a.0bb.0cc.ddd) in ET-LAN-Adresse

Erklärung Diese Funktion wandelt eine Zeichenkette im IP-Adressformat in einen Wert vom Datentyp DWORD um.

In ca. 50 Schritten des generierten Inline-Codes für diese Funktion wird der Befehl F76_A2BIN (s. S. 616) verwendet. Dieser Befehl erwartet, dass jedes Oktet aus 3 Zeichen mit führenden Leerzeichen besteht. Ansonsten liefert die Steuerung einen Operationsfehler.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiel:



Zulässiges Format:

´Oktett1.Oktett2.Oktett3.Oktett4[Leerzeichen]´ also z.B: ´ [192.168.206.4] ´

Zulässige Zeichen:

Oktetts 1-4	Dezimalzahlen von "0“ - "9“, maximal 3 Stellen, mit oder ohne führende Nullen im Bereich von 0-255

☞

- **Wenn das Format fehlerhaft ist, wird das Ergebnis 0.**
- **Die Wandlung erfolgt so, dass das höchste Byte der ET-LAN-Adresse das erste Oktett darstellt und das niedrigste Byte der IP-Adresse das vierte Oktett darstellt. Dieses Format der ET-LAN-Adresse wird z.B. von den ET-LAN-Steckmodulen der FP-Serien verwendet.**

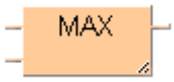
Kapitel 8

Auswahlfunktionen

MAX

Maximumfunktion

Erklärung MAX ermittelt die Eingangsvariable mit dem höchsten Wert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von MAX (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
Alle außer STRING	Eingang 1	Wert 1
Alle außer STRING	Eingang 2	Wert 2
Alle außer STRING	Ausgang wie Eingang	Ergebnis ist der größere Eingangsvariablenwert

☞ **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

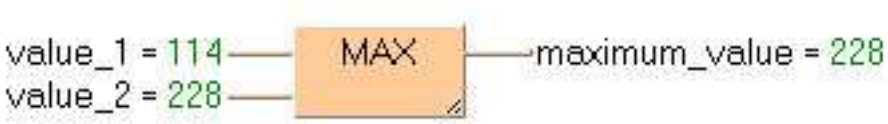
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	value_1	INT	0	all types allowed
1	VAR	value_2	INT	0	all types allowed
2	VAR	maximum_value	INT	0	all types allowed

In diesem Beispiel wurden die Eingangsvariablen (**value_1** und **value_2**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Value_1** und **value_2** werden miteinander verglichen. Der höchste Wert von allen Eingangsvariablen wird in **maximum_value** geschrieben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
maximum_value:=MAX(value_1, value_2);
```

MIN

Minimumfunktion

Erklärung MIN ermittelt die Eingangsvariable mit dem kleinsten Wert.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von MIN (s. S. 1194)**

☞ **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**

Datentypen

Datentyp	E/A	Funktion
Alle außer STRING	Eingang 1	Wert 1
Alle außer STRING	Eingang 2	Wert 2
Alle außer STRING	Ausgang wie Eingang	Ergebnis ist der kleinere Eingangsvariablenwert

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

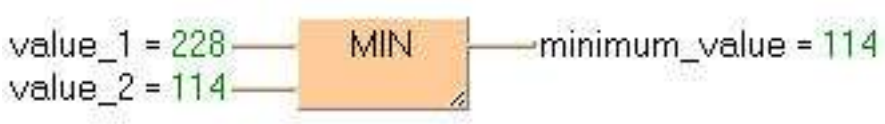
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	value_1	INT	0	all types allowed
1	VAR	value_2	INT	0	all types allowed
2	VAR	minimum_value	INT	0	all types allowed

In diesem Beispiel wurden die Eingangsvariablen (**value_1** und **value_2**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Value_1** und **value_2** werden miteinander verglichen. Der niedrigste Wert von allen Eingangsvariablen wird in **minimum_value** geschrieben.

KOP



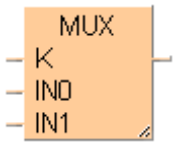
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
minimum_value:=MIN(value_1, value_2);
```

MUX

Multiplexer

Erklärung Die Multiplexer-Funktion wählt eine Eingangsvariable aus und schreibt deren Wert in die Ausgangsvariable. Mit der 1. Eingangsvariablen legen Sie fest, welche Eingangsvariable (IN1 oder IN2 oder...) in die Ausgangsvariable geschrieben werden soll. Die Funktion MUX können Sie für eine beliebige Anzahl an Eingängen konfigurieren.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von MUX (s. S. 1194)**

☞

- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Der Unterschied zwischen den Funktionen MUX und SEL (s. S. 253) besteht darin, dass Sie bei MUX mit einem Integer-Wert aus mehreren Kanälen, bei SEL mit einem booleschen Wert nur aus zwei Kanälen auswählen können.**
- **Die Anzahl der Eingangskontakte liegt zwischen 2 und 28.**

Datentypen

Datentyp	E/A	Funktion
INT	Eingang 1	Wählt beschreibbaren Kanal für zweiten oder dritten Eingangswert
Alle Datentypen	Eingang 2	Wert 1
Alle Datentypen	Eingang 3	Wert 2
Alle Datentypen	Ausgang wie 2. und 3. Eingang	Ergebnis

Die 2. und 3. Eingangsvariablen müssen vom gleichen Typ sein.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

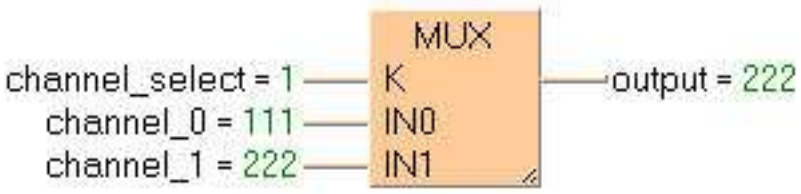
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	channel_select	INT	0	value '0' to 'n'
1	VAR	channel_0	INT	0	all types allowed
2	VAR	channel_1	INT	0	all types allowed
3	VAR	output	INT	0	all types allowed

In diesem Beispiel wurden die Eingangsvariablen (**channel_select, channel_0** und **channel_1**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf In Kanalauswahl **channel_select** steht der Integer-Wert (0, 1...n), mit dem **channel_0** oder **channel_1** ausgewählt wird. Das Ergebnis wird in **output** geschrieben.

KOP



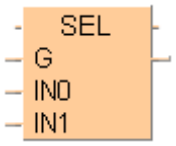
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output :=MUX ( K := channel_select , IN0 := channel_0 ,
        IN1 := channel_1 );
```

SEL

Binäre Auswahl

Erklärung Mit der ersten Eingangsvariablen (Datentyp BOOL) von SEL legen Sie fest, welche Eingangsvariable in die Ausgangsvariable geschrieben werden soll. Wenn der boolesche Wert = 0 ist (FALSE), wird die Eingangsvariable **IN0** in die Ausgangsvariable geschrieben, sonst **IN1**.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von SEL (s. S. 1196)**

☞
- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Weitere Informationen finden Sie in der Online-Hilfe im Thema: Aktualisierungsprobleme bei Datentyp STRING**

Datentypen

Datentyp	E/A	Funktion
BOOL	G	Wählt zwischen **IN0** oder **IN1** aus
Alle Datentypen	IN0	der Wert von **IN0** wird in die Ausgangsvariable geschrieben wenn **G** = FALSE
Alle Datentypen	IN1	der Wert von **IN1** wird in die Ausgangsvariable geschrieben wenn **G** = TRUE
Alle Datentypen	Ausgang	Ergebnis **IN0** oder **IN1**

☞ **Der Unterschied zwischen den Funktionen SEL und MUX (s. S. 251) besteht darin, das bei SEL ein boolescher Wert und bei MUX eine ganze Zahl (INT) zur Kanalauswahl dient. Mit MUX können Sie daher zwischen mehr als zwei Kanälen wählen.**

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

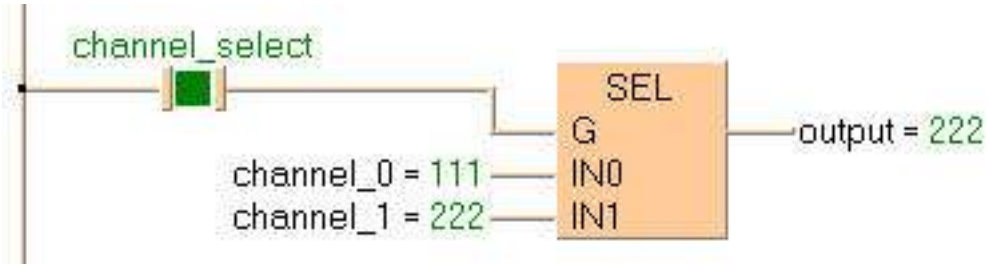
POE Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	channel_select	BOOL	FALSE
1	VAR	channel_0	INT	0
2	VAR	channel_1	INT	0
3	VAR	output	INT	0

In diesem Beispiel wurden die Eingangsvariablen (**channel_select**, **channel_0** und **channel_1**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn **channel_select** den Wert 0 besitzt, wird **channel_0** in die Ausgangsvariable geschrieben, andernfalls **channel_1**.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output := SEL(G := channel_select, IN0 := channel_0, IN1 := channel_1);
```

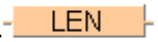
Kapitel 9

Zeichenketten-Funktionen

LEN

Länge einer Zeichenkette

Erklärung LEN berechnet die Länge der als Eingangsvariable angegebenen Zeichenkette und schreibt die Anzahl Zeichen in die Ausgangsvariable.

Symbol: 

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von LEN (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
STRING	Eingang	Eingangsdatentyp
INT	Ausgang	Länge der Zeichenkette

☞

- **Bei Kleinsteuerungen wie FP-e oder FP0 müssen Sie darauf achten, dass die Länge der Ergebniszeichenkette mindestens so groß ist wie die Länge der Ausgangszeichenkette.**
- **Wenn die Länge der Zeichenkette, die für die Eingangsvariable (input_string) im Feld "Typ" definiert ist, überschritten wird, tritt ein Fehler auf (siehe Sondermerker zur Fehlerauswertung).**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Zeichenkette länger ist als die für die Eingangsvariable im Feld "Typ" definierte Länge
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_string	STRING[12]	'Panasonic'	sample string
1	VAR	output_value	INT	0	result: here 9

In diesem Beispiel wurde die Eingangsvariable **input_string** deklariert. Stattdessen können Sie die Zeichenkette (**'Panasonic'**) auch direkt an den Eingang der Funktion schreiben. Die Zeichenkette muss im POE-Kopf und an den Funktionseingängen in Anführungszeichen stehen.

Rumpf Die Länge (9) der Zeichenkette **input_string** (**'Panasonic'**) wird in **output_value** geschrieben.

KOP

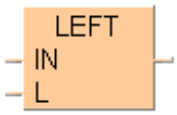
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=LEN(input_value);
```

LEFT — Zeichen von links

Erklärung LEFT kopiert, ausgehend von links, **n** Zeichen der als erste Eingangsvariable angegebenen Zeichenkette in die Ausgangsvariable. Die Anzahl der zu liefernden Zeichen **n** legen Sie mit der zweiten Eingangsvariable fest.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞

- **Wenn die Anzahl der zu liefernden Zeichen größer ist als die Eingangs-Zeichenkette, wird die gesamte Zeichenkette in die Ausgangsvariable output_string kopiert.**
- **Wenn die Ausgangs-Zeichenkette länger ist als die für die Ausgangsvariable im Feld "Typ" definierte Länge, werden nur so viele Zeichen von links kopiert, wie in die Ausgangsvariable passen. Sondermerker R9009 (%MX0.900.9) wird gesetzt.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

SPS-Typen **Verfügbarkeit von LEFT (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
STRING	Eingang 1	Eingangszeichenkette
INT	Eingang 2	Anzahl der Zeichen der Eingangszeichenkette, die von links kopiert werden
STRING	Ausgang	kopierte Zeichenkette

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Zeichenkette länger ist als die für die Eingangsvariable im Feld "Typ" definierte Länge
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ Ausgangs-Zeichenkette länger ist als die für die Ausgangsvariable im Feld "Typ" definierte Länge

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

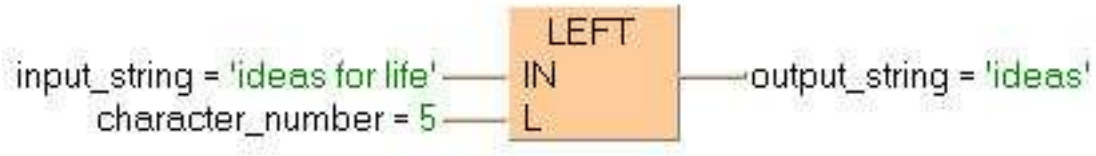
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_string	STRING[15]	'Ideas for life'	sample string
1	VAR	output_string	STRING[5]	''	result: here 'Ideas'
2	VAR	character_number	INT	5	characters to be delivered

In diesem Beispiel wurden die Eingangsvariablen (**input_string** und **character_number**) deklariert. Stattdessen können Sie die Eingangs-Zeichenkette **'Ideas for life'** und die Anzahl der zu liefernden Zeichen auch direkt an die Eingänge der Funktion schreiben. Die Zeichenkette muss im POE-Kopf und an den Funktionseingängen in Anführungszeichen stehen.

Rumpf Aus **input_string** (**'Ideas for life'**) werden, ausgehend von links, die mit **character_number** festgelegte Anzahl Zeichen (5) in **output_string** kopiert (**'Ideas'**).

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

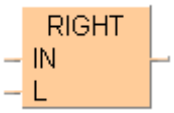
```
output_string:=LEFT(IN:=input_string, L:=character_number);
```

RIGHT

Zeichen von rechts

Erklärung RIGHT kopiert, ausgehend von rechts, n Zeichen der als erste Eingangsvariable angegebenen Zeichenkette in die Ausgangsvariable. Die Anzahl der zu liefernden Zeichen n legen Sie mit der zweiten Eingangsvariable fest.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von RIGHT (s. S. 1196)**

Datentypen

Datentyp	E/A	Funktion
STRING	Eingang 1	Eingangszeichenkette
INT	Eingang 2	Anzahl der Zeichen der Eingangszeichenkette, die von rechts kopiert werden
STRING	Ausgang	kopierte Zeichenkette

☞

- **Wenn die Anzahl der zu liefernden Zeichen größer ist als die Eingangs-Zeichenkette, wird die gesamte Zeichenkette in die Ausgangsvariable output_string kopiert.**
- **Wenn die Ausgangs-Zeichenkette länger ist als die für die Ausgangsvariable im Feld "Typ" definierte Länge, werden nur so viele Zeichen von rechts kopiert, wie in die Ausgangsvariable passen. Sondermerker R9009 (%MX0.900.9) wird gesetzt.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe. (Bis zu 200 Schritte).**

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Zeichenkette länger ist als die für die Eingangsvariable im Feld "Typ" definierte Länge
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ Ausgangs-Zeichenkette länger ist als die für die Ausgangsvariable im Feld "Typ" definierte Länge

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_string	STRING[15]	'ideas for life'	sample string
1	VAR	character_number	INT	4	characters to be delivered
2	VAR	output_string	STRING[4]	''	result here= 'life'

In diesem Beispiel wurden die Eingangsvariablen **input_string** und **character_number** deklariert. Stattdessen können Sie die Eingangs-Zeichenkette **'Ideas for life'** und die Anzahl der zu liefernden Zeichen auch direkt an die Eingänge der Funktion schreiben. Die Zeichenkette muss im POE-Kopf und an den Funktionseingängen in Anführungszeichen stehen.

Rumpf Aus **input_string** (**'Ideas for life'**) werden, ausgehend von rechts, die mit **character_number** festgelegte Anzahl Zeichen (4) in **output_string** kopiert (**'Ideas'**).

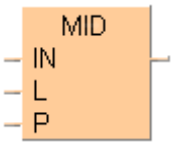
KOP

MID

Zeichen ab Position

Erklärung MID kopiert, ausgehend von der Position **P**, **L** Zeichen der als erste Eingangsvariable angegebenen Zeichenkette in die Ausgangsvariable. Die Anzahl der zu liefernden Zeichen **L** legen Sie mit der zweiten, die Startposition **P** mit der dritten Eingangsvariable fest.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von MID (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
STRING	Eingang 1	Eingangszeichenkette
INT	Eingang 2	Anzahl der zu kopierenden Zeichen der Eingangszeichenkette
INT	Eingang 3	Position, an der das Kopieren beginnt
STRING	Ausgang	kopierte Zeichenkette

☞

- **Die Summe aus Startposition und Anzahl zu liefernder Zeichen sollte nicht größer sein als die Eingangs-Zeichenkette. Wenn z.B. aus einer 10 Zeichen langen Zeichenkette 5 Zeichen ab der Position 7 geliefert werden sollen, erhalten Sie nur die letzten 4 Zeichen.**
- **Wenn die Ausgangs-Zeichenkette länger ist als die für die Ausgangsvariable (output_string) im Feld "Typ" definierte Länge, werden nur so viele Zeichen kopiert, wie in die Ausgangsvariable passen. Sondermerker R9009 (%MX0.900.9) wird gesetzt.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe. (Bis zu 200 Schritte).**

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Eingangs-Zeichenkette länger ist als die für die Eingangsvariable im Feld "Typ" definierte Länge
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ Ausgangs-Zeichenkette länger ist als die für die Ausgangsvariable im Feld "Typ" definierte Länge

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

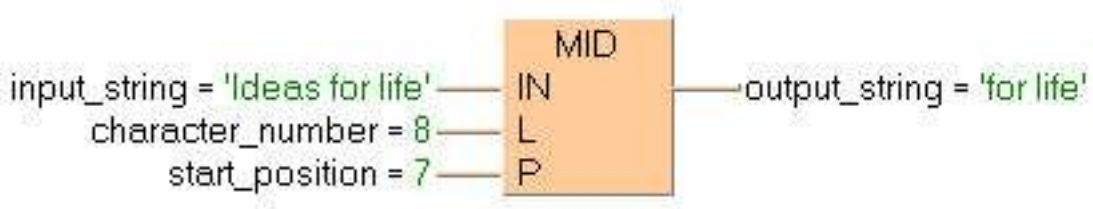
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	input_string	STRING[15]	'Ideas for life'	sample string
1	VAR	character_number	INT	8	characters to be delivered
2	VAR	start_position	INT	0	position to start copying
3	VAR	output_string	STRING[5]	''	result here: 'for life'

In diesem Beispiel wurden die Eingangsvariablen (**input_string**, **character_number** und **start_position**) deklariert. Stattdessen können Sie die Eingangs-Zeichenkette **'Ideas for life'**, die Anzahl der zu liefernden Zeichen und die Startposition auch direkt an die Eingänge der Funktion schreiben. Die Zeichenkette muss im POE-Kopf und an den Funktionseingängen in Hochkommata stehen.

Rumpf Ausgehend von der Variable **start_position** (7), wird die mit **character_number** festgelegte Anzahl Zeichen (8) von der Eingangsvariablen **input_string** (**'Ideas for life'**) in die Ausgangsvariable **output_string** kopiert (**'for life'**).

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_string:=MID(IN:=input_string, L:=character_number,
P:=start_position);
```

CONCAT

Zeichenketten zusammenfügen

Erklärung CONCAT fügt die als zweite und folgende Eingangsvariable (IN1 + IN2 + ...) angegebenen Zeichenketten an die als erste Eingangsvariable angegebene Zeichenkette an und schreibt die resultierende Zeichenkette in die Ausgangsvariable.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von CONCAT (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
STRING	Eingang 1	Erste Eingangszeichenkette
STRING	Eingang 2	Zeichenkette, die an die erste Zeichenkette angehängt wird
STRING	Ausgang	Ergebniszeichenkette

☞

- **Wenn die Ausgangs-Zeichenkette länger ist als die für die Ausgangsvariable (output_string) im Feld "Typ" definierte Länge, werden nur so viele Zeichen von links kopiert, wie in die Ausgangsvariable passen. Sondermerker R9009 (%MX0.900.9) wird gesetzt.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe.**

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Zeichenkette länger ist als die für die Eingangsvariable im Feld "Typ" definierte Länge
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ Ausgangs-Zeichenkette länger ist als die für die Ausgangsvariable im Feld "Typ" definierte Länge

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

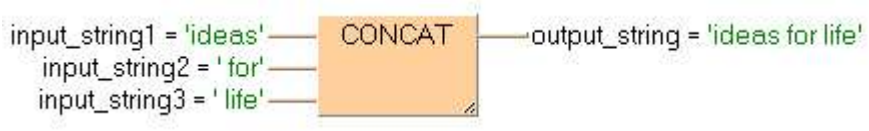
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	input_string1	STRING[32]	'Ideas '
1	VAR	input_string2	STRING[32]	'for'
2	VAR	input_string3	STRING[32]	'life'
3	VAR	output_string	STRING[32]	''

In diesem Beispiel wurden die Eingangsvariablen (**input_string1, input_string2** und **input_string3**) deklariert. Stattdessen können Sie die Zeichenketten (**'Ideas'**, **' for' und ' life'**) auch direkt an die Eingänge der Funktion schreiben. Die Zeichenketten müssen im POE-Kopf und in der Funktion in Hochkommata stehen.

Rumpf **Input_string3** (**' life'**) wird an **input_string2** (**' for'**) und diese Zeichenkette wird an **input_string1** (**'Ideas'**) angehängt. Die resultierende Zeichenkette (**'Ideas for life'**) wird in **output_string** geschrieben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_string:=CONCAT(input_string1, input_string2, input_string3);
```

DELETE

Zeichen aus einer Zeichenkette löschen

Erklärung

DELETE löscht **L** Zeichen ab der Position **P** aus einer als erste Eingangsvariable angegebenen Zeichenkette und schreibt die resultierende Zeichenkette in die Ausgangsvariable. Die Anzahl der zu löschenden Zeichen **L** legen Sie mit der zweiten, die Startposition **P** mit der dritten Eingangsvariable fest.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DELETE (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
STRING	Eingang 1	Eingangszeichenkette
INT	Eingang 2	Anzahl der Zeichen der Eingangszeichenkette, die gelöscht werden
INT	Eingang 3	Position, an der das Löschen beginnt
STRING	Ausgang	Ergebniszeichenkette

☞

- **Wenn die Ausgangs-Zeichenkette länger ist als die für die Ausgangsvariable (output_string) im Feld "Typ" definierte Länge, werden nur so viele Zeichen von links kopiert, wie in die Ausgangsvariable passen. Sondermerker R9009 (%MX0.900.9) wird gesetzt.**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe. (Bis zu 200 Schritte).**

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Zeichenkette länger ist als die für die Eingangsvariable im Feld "Typ" definierte Länge
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ Ausgangs-Zeichenkette länger ist als die für die Ausgangsvariable im Feld "Typ" definierte Länge

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

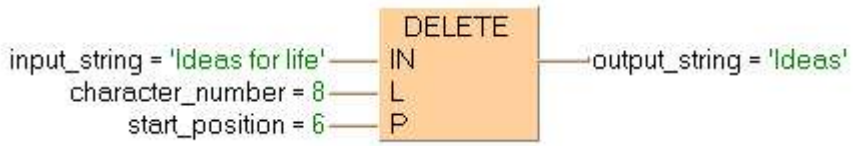
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	input_string	STRING[15]	'Ideas for life'
1	VAR	character_number	INT	8
2	VAR	start_position	INT	6
3	VAR	output_string	STRING[5]	''

In diesem Beispiel wurden die Eingangsvariablen (**input_string**, **character_number** und **start_position**) deklariert. Stattdessen können Sie die Eingangs-Zeichenkette **'Ideas for life'**, die Anzahl der zu löschenden Zeichen und die Startposition auch direkt an die Eingänge der Funktion schreiben. Die Zeichenkette muss im POE-Kopf und an den Funktionseingängen in Hochkommata stehen.

Rumpf Ausgehend von der Variable **start_position (6)**, wird die mit **character_number** festgelegte Anzahl Zeichen **(8)** von der Eingangsvariablen **input_string** (**'Ideas for life'**) gelöscht. Die resultierende Zeichenkette (**'Ideas'**) wird in **output_string** geschrieben.

KOP



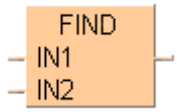
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_string:=DELETE(input_string, character_number, start_position);
```

FIND

Zeichenkette suchen

Erklärung FIND sucht in der ersten Eingangszeichenkette nach der zweiten Eingangszeichenkette und gibt die Position aus, an der die zweite Eingangszeichenkette gefunden wurde. Das Ergebnis wird in die Ausgangsvariable geschrieben. Wenn die zweite Eingangszeichenkette in der ersten Eingangszeichenkette nicht vorkommt, wird der Wert 0 ausgegeben.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von FIND (s. S. 1192)**

Datentypen

Datentyp	E/A	Funktion
STRING	Eingang 1	Eingangszeichenkette
STRING	Eingang 2	Zeichenkette, die in der Eingangszeichenkette durchsucht wird
INT	Ausgang	Position, an der die gesuchte Zeichenkette gefunden wurde

☞

- **Wenn die Zeichenketten länger sind als die, die für die Länge der Eingangsvariablen (input_string_1 und input_string_2) im Feld "Typ" definiert sind, tritt ein Fehler auf (siehe Sondermerker zur Fehlerauswertung).**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe. (Bis zu 200 Schritte).**

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Eingangszeichenketten länger sind als die für die Eingangsvariable im Feld "Typ" definierte Länge
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

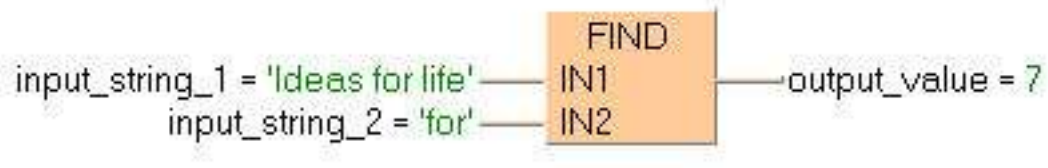
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	input_string_1	STRING[15]	'ideas for l...
1	VAR	input_string_2	STRING[3]	'for'
2	VAR	output_value	INT	0

In diesem Beispiel wurden die Eingangsvariablen (**input_string_1** und **input_string_2**) deklariert. Stattdessen können Sie die Zeichenketten (**'Ideas for life'** und **'for'** auch direkt an die Eingänge der Funktion schreiben. Die Zeichenketten müssen im POE-Kopf und in der Funktion in Hochkommata stehen.

Rumpf Die zweite Eingangszeichenkette **Input_string_2** (**'for'**) wird innerhalb der ersten Eingangszeichenkette **input_string_1** (**'Ideas for life'**) gesucht. Die Position des ersten Vorkommens (**7**) wird in **output_value** geschrieben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

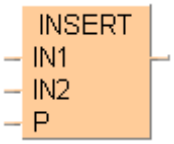
```
output_value := FIND(input_string_1, input_string_2);
```

INSERT

Einfügen einer Zeichenkette

Erklärung INSERT fügt die am Eingang **IN2** eingegebene Zeichenkette in die am Eingang **IN1** eingegebene Zeichenkette ab Position **P** ein. Das Ergebnis wird in die Ausgangsvariable geschrieben.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von INSERT (s. S. 1193)**

Datentypen

Datentyp	E/A	Funktion
STRING	Eingang 1	Eingangszeichenkette
STRING	Eingang 2	Zeichenkette, die in die Eingangszeichenkette eingefügt wird
INT	Eingang 3	Position, an der die Zeichenkette eingefügt wird
STRING	Ausgang	Ergebniszeichenkette

☞

- **Wenn die Zeichenketten länger sind als die, die für die Länge der Eingangsvariablen (input_string_1 und input_string_2) im Feld "Typ" definiert sind, tritt ein Fehler auf (siehe Sondermerker zur Fehlerauswertung).**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe. (Bis zu 200 Schritte).**

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	Eingangszeichenketten länger sind als die für die Eingangsvariable im Feld "Typ" definierte Länge
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

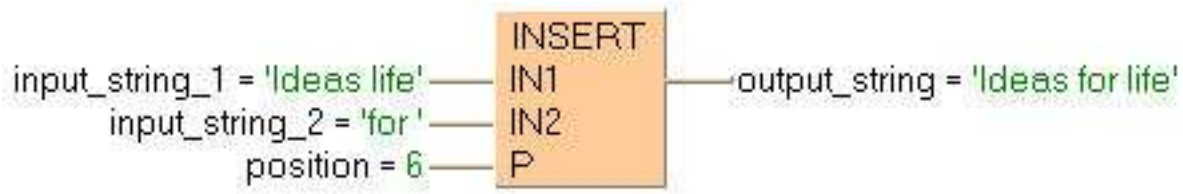
	Klasse	Bezeichner	Typ	Initial
0	VAR	input_string1	STRING[32]	'ideas life'
1	VAR	input_string2	STRING[32]	'for'
2	VAR	position	INT	6
3	VAR	output_string	STRING[32]	''

Rumpf

In diesem Beispiel wurden die Eingangsvariablen **input_string1**, **input_string2** und **position** deklariert. Statt dessen können Sie die Werte direkt an die Eingänge der Funktion schreiben. Die Zeichenketten müssen im POE-Kopf und an den Funktionseingängen in Hochkommata stehen. Die zweite Eingangszeichenkette **input_string2** ('for ') wird in die erste Eingangszeichenkette **input_string1** ('Ideas life') an Position 6 eingefügt. Das Ergebnis ('Ideas for life') wird in die Ausgangsvariable **output_value** geschrieben.

Im KOP-Beispiel wurde das Fernglassymbol im Online-Modus aktiviert, deshalb können Sie das Ergebnis sofort sehen.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

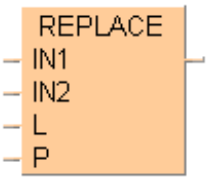
```
output_value:=INSERT(IN1:=input_string1, IN2:=input_string2, P:=6);
```

REPLACE

ersetzt Zeichen in einer Zeichenkette

Erklärung

Die am Eingang **IN2** eingegebene Zeichenkette ersetzt bestimmte Zeichen in der am Eingang **IN1** eingegebenen Zeichenkette. Die Zahl der Zeichen, d.h. die Länge L der zu ersetzenden Zeichenkette, wird am Eingang **L** eingegeben. Die Position, an der die Ersetzung beginnen soll, wird am Eingang **P** eingegeben. Das Ergebnis wird in die Ausgangsvariable geschrieben.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von REPLACE (s. S. 1196)

- **Wenn die Zeichenketten länger sind als die, die für die Länge der Eingangsvariablen (input_string_1 und input_string_2) im Feld "Typ" definiert sind, tritt ein Fehler auf (siehe Sondermerker zur Fehlerauswertung).**
- **Die Anzahl der Schritte kann sich je nach verwendeter SPS und den Parametern ändern, siehe auch Programmschritttabelle für code-intensive Befehle in der Online-Hilfe. (Bis zu 200 Schritte).**

Datentypen

Datentyp	E/A	Funktion
STRING	Eingang 1	Eingangszeichenkette
STRING	Eingang 2	Ersatzzeichenkette
INT	Eingang 3	Anzahl der zu ersetzenden Zeichen in der Eingangszeichenkette
INT	Eingang 4	Position, ab der Zeichen ersetzt werden
STRING	Ausgang	Ergebniszeichenkette

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Eingangszeichenketten länger sind als die für die Eingangsvariable im Feld "Typ" definierte Länge
R9008	%MX0.900.8	kurzzeitig	

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

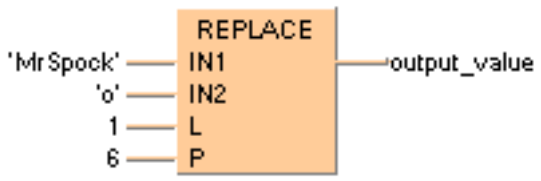
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	output_value	STRING[32]	''	result: 'MrSpook'

Rumpf

In diesem Beispiel werden Konstante direkt an die Eingänge der Funktion geschrieben. Statt dessen können Sie Variablen im POE-Kopf deklarieren. Die Zeichenketten müssen in Anführungszeichen im POE-Kopf und an den Funktionseingängen stehen. Hier wird 'c' in der Zeichenkette 'MrSpock' mit 'o' ersetzt; das Ergebnis ist 'MrSpook'.

KOP



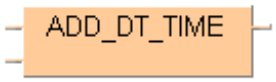
Kapitel 10

Arithmetische Funktionen für Datentypen der Zeit

ADD_DT_TIME

TIME zu DATE_AND_TIME addieren

Erklärung ADD_DT_TIME addiert den Wert einer Variable vom Datentyp TIME zu Datum und Uhrzeit in der Variable vom Datentyp DATE_AND_TIME. Das Ergebnis wird in einer Variable vom Datentyp DATE_AND_TIME gespeichert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von ADD_DT_TIME (s. S. 1184)**

Datentypen

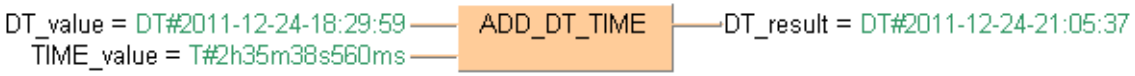
Datentyp	E/A	Funktion
DATE_AND_TIME	Eingang 1	Augend
TIME	Eingang 2	Summand
DATE_AND_TIME	Ausgang	Summe

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DT_value	DATE_AND_TIME	DT#2011-12-24-18:29:59
1	VAR	TIME_value	TIME	T#2h35m38s560ms
2	VAR	DT_result	DATE_AND_TIME	DT#2001-01-01-00:00:00

KOP



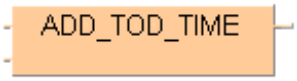
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DT_result := ADD_DT_TIME (DT_value, TIME_value);
```

ADD_TOD_TIME

TIME zu TIME_OF_DAY addieren

Erklärung ADD_TOD_TIME addiert eine Variable vom Datentyp TIME zur Uhrzeit .. Das Ergebnis wird in einer Variable vom Datentyp TIME_OF_DAY gespeichert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von ADD_TOD_TIME (s. S. 1184)**

Datentypen

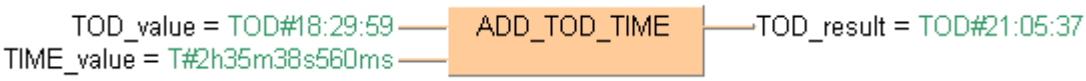
Datentyp	E/A	Funktion
TIME_OF_DAY	Eingang 1	Augend
TIME	Eingang 2	Summand
TIME_OF_DAY	Ausgang	Summe

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	TOD_value	TIME_OF_DAY	TOD#18:29:59	
1	VAR	TIME_value	TIME	T#2h35m38s560ms	
2	VAR	TOD_result	TIME_OF_DAY	TOD#00:00:00	

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

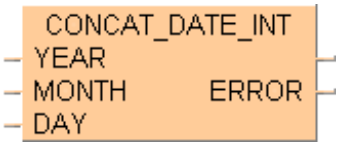
```
TOD_result := ADD_TOD_TIME (TOD_value, TIME_value);
```

CONCAT_DATE_INT

INT-Werte verketten, um ein Datum zu bilden

Erklärung

CONCAT_DATE_INT verkettet die INTEGER-Werte für Jahr, Monat und Tag. Das Ergebnis wird in der Ausgangsvariable vom Datentyp DATE gespeichert. Der Boolesche ERROR-Ausgang ist gesetzt, wenn die Eingangswerte ungültige Datums- oder Zeitwerte sind.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von CONCAT_DATE_INT (s. S. 1184)

Datentypen

Datentyp	E/A	Funktion
INT	Eingang 1 Eingang 2 Eingang 3	Jahr Monat Tag
DATE	Ausgang	Ergebnis
BOOL	Ausgang	Der Boolesche ERROR-Ausgang ist gesetzt, wenn die Eingangswerte ungültige Datums- oder Zeitwerte sind.

Beispiel

In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DATE_value	DATE	D#2001-01-01
1	VAR	YEAR_value	INT	2011
2	VAR	MONTH_value	INT	12
3	VAR	DAY_value	INT	24
4	VAR	ERROR	BOOL	FALSE

KOP

ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

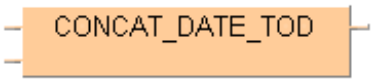
```
DATE_value := CONCAT_DATE_INT(YEAR := YEAR_value,
                              MONTH := MONTH_value,
                              DAY := DAY_value,
                              ERROR => ERROR);
```

CONCAT_DATE_TOD

Datum mit Uhrzeit verketten

Erklärung

CONCAT_DATE_TOD verkettet einen Wert vom Datentyp DATE mit einem Wert vom Datentyp TIME_OF_DAY. Das Ergebnis wird in der Ausgangsvariable vom Datentyp DATE_AND_TIME gespeichert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von CONCAT_DATE_TOD (s. S. 1184)

Datentypen

Datentyp	E/A	Funktion
DATE	Eingang 1	Datum
TIME_OF_DAY	Eingang 2	Uhrzeit
DATE_AND_TIME	Ausgang	Ergebnis

Beispiel

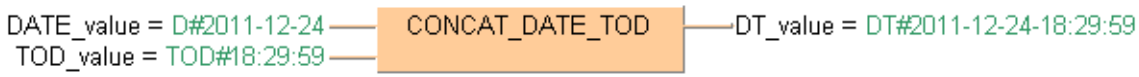
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DT_value	DATE_AND_TIME	DT#2001-01-01-00:00:00
1	VAR	DATE_value	DATE	D#2011-12-24
2	VAR	TOD_value	TOD	TOD#18:29:59

KOP



ST

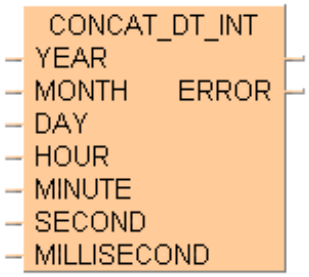
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DT_value := CONCAT_DATE_TOD (DATE_value, TOD_value);
```

CONCAT_DT_INT

INT-Werte verketten, um Datum und Uhrzeit zu bilden

Erklärung CONCAT_DT_INT verkettet die INT-Werte für Jahr, Monat, Tag, Stunde, Minute, Sekunde und Millisekunde. Das Ergebnis wird in der Ausgangsvariable vom Datentyp DATE_AND_TIME gespeichert. Der Boolesche ERROR-Ausgang ist gesetzt, wenn die Eingangswerte ungültige Datums- oder Zeitwerte sind.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von CONCAT_DT_INT (s. S. 1184)**

Datentypen

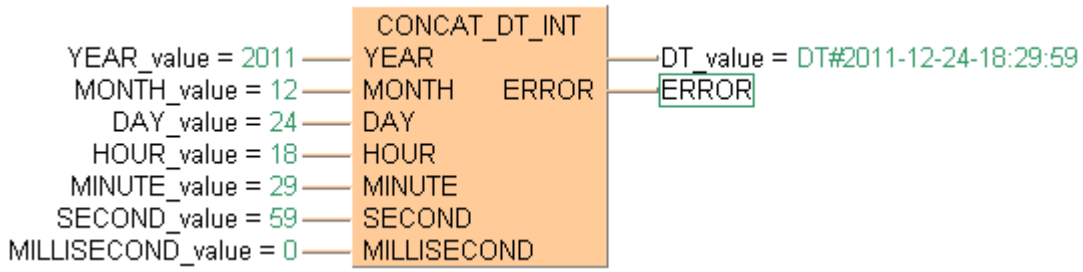
Datentyp	E/A	Funktion
INT	Eingang 1 Eingang 2 Eingang 3 Eingang 4 Eingang 5 Eingang 6 Eingang 7	Jahr Monat Tag Stunde Minute Sekunde Millisekunde
DATE_AND_TIME	Ausgabe	Ergebnis
BOOL	Ausgabe	Der Boolesche ERROR-Ausgang ist gesetzt, wenn die Eingangswerte ungültige Datums- oder Zeitwerte sind.

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DT_value	DATE_AND_TIME	DT#2001-01-01-00:00:00
1	VAR	YEAR_value	INT	2011
2	VAR	MONTH_value	INT	12
3	VAR	DAY_value	INT	24
4	VAR	HOUR_value	INT	18
5	VAR	MINUTE_value	INT	29
6	VAR	SECOND_value	INT	59
7	VAR	MILLISECOND_value	INT	0
8	VAR	ERROR	BOOL	FALSE

KOP



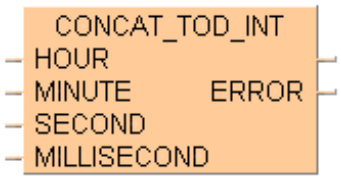
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DT_value := CONCAT_DT_INT(YEAR := YEAR_value,
                MONTH := MONTH_value,
    DAY := DAY_value,
    HOUR := HOUR_value,
    MINUTE := MINUTE_value,
    SECOND := SECOND_value,
    MILLISECOND := MILLISECOND_value,
    ERROR => ERROR);
```

CONCAT_TOD_INT

INT-Werte verketten, um Uhrzeit zu bilden

Erklärung CONCAT_TOD_INT verkettet die INTEGER-Werte für Stunde, Minute, Sekunde und Millisekunde. Das Ergebnis wird in der Ausgangsvariable vom Datentyp TIME_OF_DAY gespeichert. Der Boolesche ERROR-Ausgang ist gesetzt, wenn die Eingangswerte ungültige Datums- oder Zeitwerte sind.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von CONCAT_TOD_INT (s. S. 1184)**

Datentypen

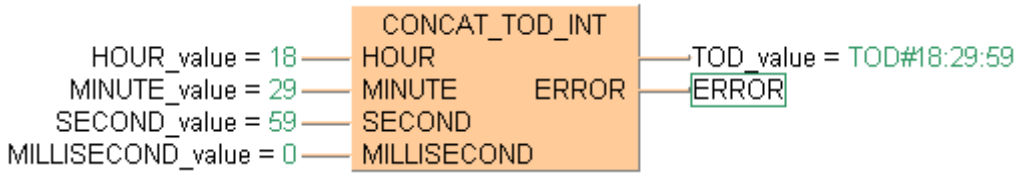
Datentyp	E/A	Funktion
INT	Eingang 1 Eingang 2 Eingang 3 Eingang 4	Stunde Minute Sekunde Millisekunde
TIME_OF_DAY	Ausgang	Ergebnis
BOOL	Ausgang	Der Boolesche ERROR-Ausgang ist gesetzt, wenn die Eingangswerte ungültige Datums- oder Zeitwerte sind.

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	TOD_value	TIME_OF_DAY	TOD#00:00:00
1	VAR	HOUR_value	INT	18
2	VAR	MINUTE_value	INT	29
3	VAR	SECOND_value	INT	59
4	VAR	MILLISECOND_value	INT	0
5	VAR	ERROR	BOOL	FALSE

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
TOD_value := CONCAT_TOD_INT(HOUR := HOUR_value,
                            MINUTE := MINUTE_value,
                    SECOND := SECOND_value,
                    MILLISECOND := MILLISECOND_value,
                    ERROR => ERROR);
```

GET_RTC_DTBCD

Auswerten der Echtzeituhr

Erklärung GET_RTC_DT liest den Wert der Echtzeituhr für die Uhr-/Kalenderfunktion der SPS aus. Verfügt die SPS über keine Echtzeituhr, oder ist die Uhr-/Kalenderfunktion außer Betrieb, ist das Ergebnis ein ungültiger Datums- und Zeitdauerwert.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von GET_RTC_DT (s. S. 1192)**

Datentypen

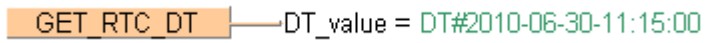
Datentyp	E/A	Funktion
DATE_AND_TIME	Ausgang	Datum und Uhrzeit

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeic...	Typ	Initial
0	VAR	bSetEdge	BOOL	FALSE
1	VAR	DT_value	DT	DT#2001-01-01-00:00:00

KOP



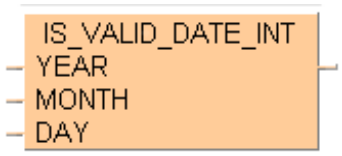
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DT_value := GET_RTC_DT();
```

IS_VALID_DATE_INT

Gültigkeitsprüfung des DATE-Werts

Erklärung IS_VALID_DATE_INT prüft, ob die Kombination der INT-Werte für Jahr, Monat und Tag einen gültigen DATE-Wert ergibt. Wenn das Datum gültig ist, wird der Boolesche Ausgangsmerker gesetzt.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von IS_VALID_DATE_INT (s. S. 1193)**

Datentypen

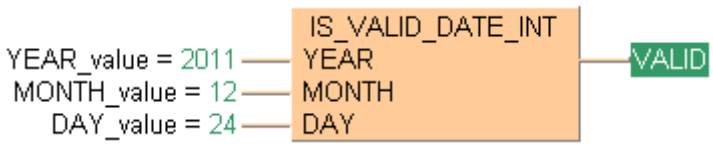
Datentyp	E/A	Funktion
INT	Eingang 1 Eingang 2 Eingang 3	Jahr Monat Tag
BOOL	Ausgang	TRUE, wenn der Ergebniswert für das Datum gültig ist.

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	YEAR_value	INT	2011
1	VAR	MONTH_value	INT	12
2	VAR	DAY_value	INT	24
3	VAR	VALID	BOOL	FALSE

KOP



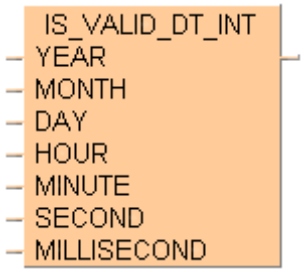
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
VALID := IS_VALID_DATE_INT(YEAR := YEAR_value,
                           MONTH := MONTH_value,
                 DAY := DAY_value);
```

IS_VALID_DT_INT

Gültigkeitsprüfung des DATE_AND_TIME-Werts

Erklärung IS_VALID_DT prüft, ob die Kombination der INT-Werte für Jahr, Monat, Tag, Stunde, Minute, Sekunde und Millisekunde einen gültigen Wert für Datum und Uhrzeit ergibt. Wenn die Werte für Datum und Uhrzeit gültig sind, wird der Boolesche Ausgangsmerker gesetzt.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von IS_VALID_DT_INT (s. S. 1193)**

Datentypen

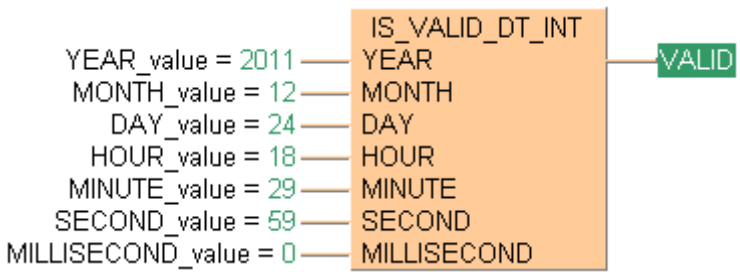
Datentyp	E/A	Funktion
INT	Eingang 1 Eingang 2 Eingang 3 Eingang 4 Eingang 5 Eingang 6 Eingang 7	Jahr Monat Tag Stunde Minute Sekunde Millisekunde
BOOL	Ausgang	TRUE, wenn die Ergebniswerte für Datum und Uhrzeit gültig sind.

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	YEAR_value	INT	2011
1	VAR	MONTH_value	INT	12
2	VAR	DAY_value	INT	24
3	VAR	HOUR_value	INT	18
4	VAR	MINUTE_value	INT	29
5	VAR	SECOND_value	INT	59
6	VAR	MILLISECOND_value	INT	0
7	VAR	VALID	BOOL	FALSE

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

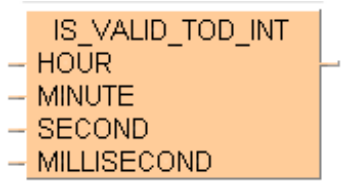
```
VALID := IS_VALID_DT_INT(YEAR := YEAR_value,
                MONTH := MONTH_value,
    DAY := DAY_value,
    HOUR := HOUR_value,
    MINUTE := MINUTE_value,
    SECOND := SECOND_value,
    MILLISECOND := MILLISECOND_value);
```

IS_VALID_TOD_INT

Gültigkeitsprüfung des TIME_OF_DAY-Werts

Erklärung

IS_VALID_TOD_INT prüft, ob die Kombination der INT-Werte für Jahr, Monat, Tag, Stunde, Minute, Sekunde und Millisekunde einen gültigen Wert für die Uhrzeit ergibt. Wenn der Wert für die Uhrzeit gültig ist, wird der Boolesche Ausgangsmerker gesetzt.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Verfügbarkeit von IS_VALID_TOD_INT (s. S. 1193)

Datentypen

Datentyp	E/A	Funktion
INT	Eingang 1 Eingang 2 Eingang 3 Eingang 4	Stunde Minute Sekunde Millisekunde
BOOL	Ausgang	TRUE, wenn der Ergebniswert für die Uhrzeit gültig ist.

Beispiel

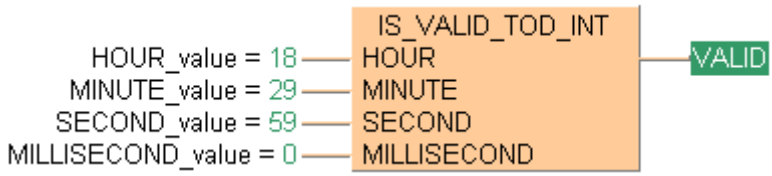
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	HOUR_value	INT	18
1	VAR	MINUTE_value	INT	29
2	VAR	SECOND_value	INT	59
3	VAR	MILLISECOND_value	INT	0
4	VAR	VALID	BOOL	FALSE

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
VALID := IS_VALID_TOD_INT(HOUR := HOUR_value,
                          MINUTE := MINUTE_value,
                          SECOND := SECOND_value,
                          MILLISECOND := MILLISECOND_value);
```

SET_RTC_DT

Einstellen der Uhr-/Kalenderfunktion

Erklärung SET_RTC_DT stellt den Wert der Echtzeituhr für die Uhr-/Kalenderfunktion der SPS ein. Verfügt die SPS über keine Echtzeituhr, oder ist die Uhr-/Kalenderfunktion außer Betrieb, ist das Ergebnis ein ungültiger Datums- und Zeitdauerwert.

SPS-Typen **Verfügbarkeit von SET_RTC_DT (s. S. 1196)**

Datentypen

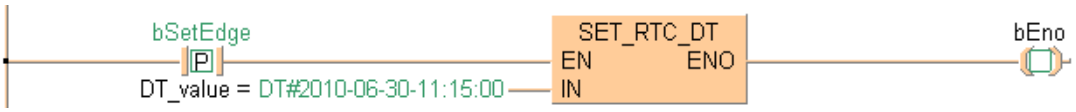
Datentyp	E/A	Funktion
DATE_AND_TIME	Eingang	Datum und Uhrzeit

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bSetEdge	BOOL	FALSE
1	VAR	DT_value	DT	DT#2010-06-30-11:15:00
2	VAR	bEno	BOOL	FALSE

KOP



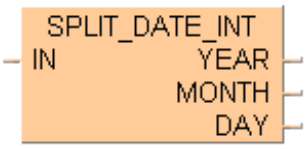
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(bSetEdge) THEN
  SET_RTC_DT(DT_value);
END_IF;
```

SPLIT_DATE_INT

DATE in INT-Werte aufteilen

Erklärung SPLIT_DATE_INT teilt einen Wert vom Datentyp DATE in INT-Werte für Jahr, Monat und Tag auf.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von SPLIT_DATE_INT (s. S. 1196)**

Datentypen

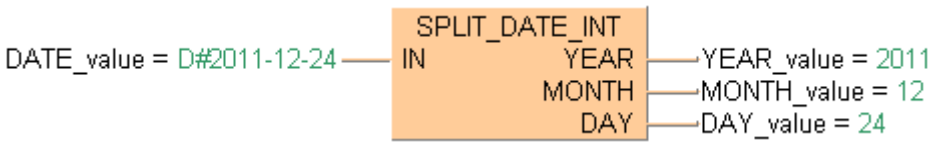
Datentyp	E/A	Funktion
DATE	Eingang	Datum
INT	Ausgang 1 Ausgang 2 Ausgang 3	Jahr Monat Tag

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DATE_value	DATE	D#2011-12-24
1	VAR	YEAR_value	INT	0
2	VAR	MONTH_value	INT	0
3	VAR	DAY_value	INT	0

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
SPLIT_DATE_INT(IN := DATE_value,
               YEAR => YEAR_value,
               MONTH => MONTH_value,
               DAY => DAY_value);
```

SPLIT_DT_INT

DATE_AND_TIME in INT-Werte aufteilen

Erklärung SPLIT_DT_INT teilt einen Wert vom Datentyp DATE_AND_TIME in INT-Werte für Jahr, Monat, Tag, Stunde, Minute, Sekunde und Millisekunde auf.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von SPLIT_DT_INT (s. S. 1196)**

Datentypen

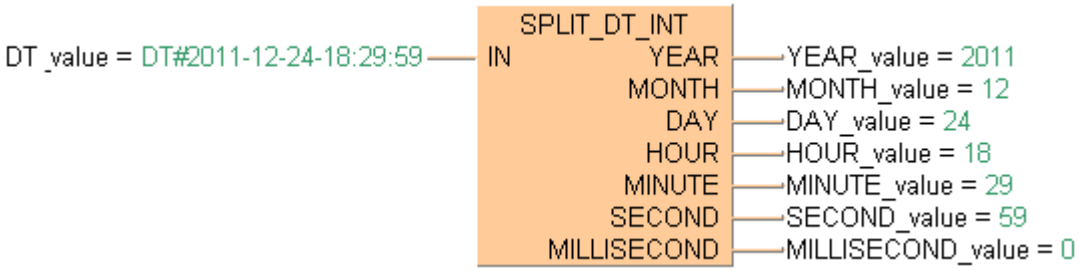
Datentyp	E/A	Funktion
DATE_AND_TIME	Eingang	Datum und Uhrzeit
INT	Ausgang 1	Jahr
	Ausgang 2	Monat
	Ausgang 3	Tag
	Ausgang 4	Stunde
	Ausgang 5	Minute
	Ausgang 6	Sekunde
	Ausgang 7	Millisekunde

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DT_value	DATE_AND_TIME	DT#2011-12-24-18:29:59
1	VAR	YEAR_value	INT	0
2	VAR	MONTH_value	INT	0
3	VAR	DAY_value	INT	0
4	VAR	HOUR_value	INT	0
5	VAR	MINUTE_value	INT	0
6	VAR	SECOND_value	INT	0
7	VAR	MILLISECOND_value	INT	0

KOP



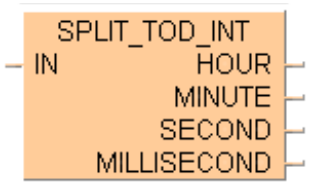
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
SPLIT_DT_INT(IN := DT_value,
             YEAR => YEAR_value,
    MONTH => MONTH_value,
    DAY => DAY_value,
    HOUR => HOUR_value,
    MINUTE => MINUTE_value,
    SECOND => SECOND_value,
    MILLISECOND => MILLISECOND_value);
```

SPLIT_TOD_INT

TIME_OF_DAY in INT-Werte aufteilen

Erklärung SPLIT_TOD_INT teilt einen Wert vom Datentyp TIME_OF_DAY in INT-Werte für Stunde, Minute, Sekunde und Millisekunde auf.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von SPLIT_TOD_INT (s. S. 1196)**

Datentypen

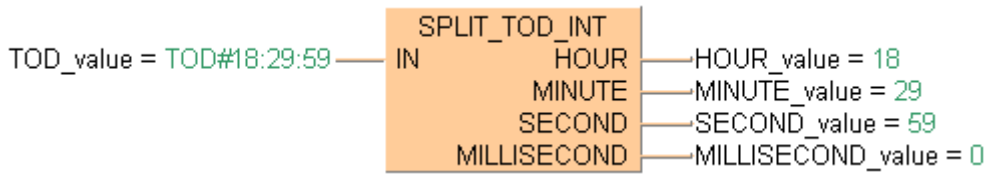
Datentyp	E/A	Funktion
TIME_OF_DAY	Eingang	Uhrzeit
INT	Ausgang 1 Ausgang 2 Ausgang 3 Ausgang 4	Stunde Minute Sekunde Millisekunde

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	TOD_value	TIME_OF_DAY	TOD#18:29:59
1	VAR	HOUR_value	INT	0
2	VAR	MINUTE_value	INT	0
3	VAR	SECOND_value	INT	0
4	VAR	MILLISECOND_value	INT	0

KOP



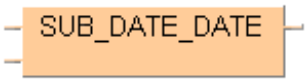
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
SPLIT_TOD_INT(IN := TOD_value,
              HOUR => HOUR_value,
    MINUTE => MINUTE_value,
    SECOND => SECOND_value,
    MILLISECOND => MILLISECOND_value);
```

SUB_DATE_DATE

DATE von DATE subtrahieren

Erklärung SUB_DATE_DATE subtrahiert einen Wert vom Datentyp DATE von einem anderen DATE-Wert. Das Ergebnis wird in der Ausgangsvariable vom Datentyp TIME gespeichert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von SUB_DATE_DATE (s. S. 1197)**

☞ **Das TIME-Ergebnis ist nur dann gültig, wenn die Differenz zwischen Minuend und Subtrahend kleiner oder gleich der maximal erlaubten TIME-Zeitdauer ist. Andernfalls ensteht ein Überlauf der TIME-Ergebnisvariable und der CARRY-Merker wird gesetzt.**

Datentypen

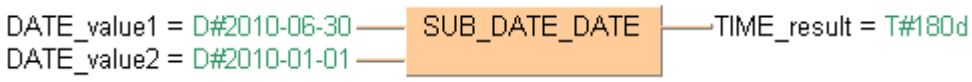
Datentyp	E/A	Funktion
DATE	Eingang 1	Minuend
DATE	Eingang 2	Subtrahend
TIME	Ausgang	Ergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DATE_value1	DATE	D#2010-07-11
1	VAR	DATE_value2	DATE	D#2010-07-03
2	VAR	TIME_result	TIME	T#0s

KOP



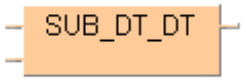
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
TIME_result := SUB_DATE_DATE(DATE_value1, DATE_value2);
```

SUB_DT_DT

DATE_AND_TIME von DATE_AND_TIME subtrahieren

Erklärung SUB_DT_DT subtrahiert einen Wert vom Datentyp DATE_AND_TIME von einem anderen DATE_AND_TIME-Wert. Das Ergebnis wird in der Ausgangsvariable vom Datentyp TIME gespeichert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von SUB_DT_DT (s. S. 1197)**

☞ **Das TIME-Ergebnis ist nur dann gültig, wenn die Differenz zwischen Minuend und Subtrahend kleiner oder gleich der maximal erlaubten TIME-Zeitdauer ist. Andernfalls ensteht ein Überlauf der TIME-Ergebnisvariable und der CARRY-Merker wird gesetzt.**

Datentypen

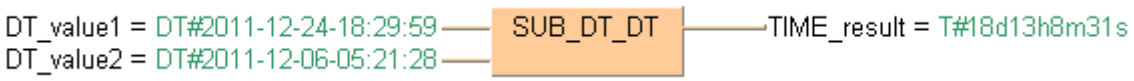
Datentyp	E/A	Funktion
DATE_AND_TIME	Eingang 1	Minuend
DATE_AND_TIME	Eingang 2	Subtrahend
TIME	Ausgang	Ergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DT_value1	DATE_AND_TIME	DT#2011-12-24-18:29:59
1	VAR	DT_value2	DATE_AND_TIME	DT#2011-12-06-05:21:28
2	VAR	TIME_result	TIME	T#0s

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

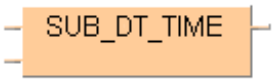
```
TIME_result := SUB_DT_DT (DT_value1, DT_value2);
```

SUB_DT_TIME

TIME von DATE_AND_TIME subtrahieren

Erklärung

SUB_DT_TIME subtrahiert einen Wert vom Datentyp TIME von einem Wert vom Datentyp DATE_AND_TIME. Das Ergebnis wird in der Ausgangsvariable vom Datentyp TIME_OF_DAY. gespeichert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von SUB_DT_TIME**

Datentypen

Datentyp	E/A	Funktion
DATE_AND_TIME	Eingang 1	Minuend
TIME	Eingang 2	Subtrahend
DATE_AND_TIME	Ausgang	Ergebnis

Beispiel

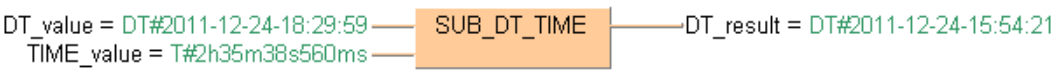
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	DT_value	DATE_AND_TIME	DT#2011-12-24-18:29:59
1	VAR	TIME_value	TIME	T#2h35m38s560ms
2	VAR	DT_result	DATE_AND_TIME	DT#2001-01-01-00:00:00

KOP



ST

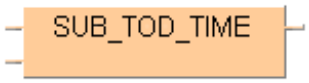
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
DT_result := SUB_DT_TIME (DT_value, TIME_value);
```

SUB_TOD_TIME

TIME von TIME_OF_DAY subtrahieren

Erklärung SUB_TOD_TIME subtrahiert einen TIME-Wert von einem Wert vom Datentyp TIME_OF_DAY. Das Ergebnis wird in der Ausgangsvariable vom Datentyp TIME_OF_DAY. gespeichert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von SUB_TOD_TIME**

Datentypen

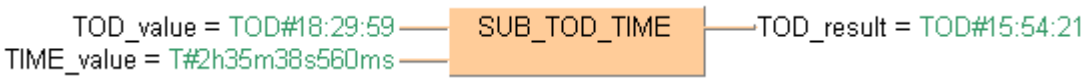
Datentyp	E/A	Funktion
TIME_OF_DAY	Eingang 1	Minuend
TIME	Eingang 2	Subtrahend
TIME_OF_DAY	Ausgang	Ergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	TOD_value	TIME_OF_DAY	TOD#18:29:59
1	VAR	TIME_value	TIME	T#2h35m38s560ms
2	VAR	TOD_result	TIME_OF_DAY	TOD#00:00:00

KOP



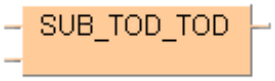
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
TOD_result := SUB_TOD_TIME (TOD_value, TIME_value);
```

SUB_TOD_TOD

TIME_OF_DAY von TIME_OF_DAY subtrahieren

Erklärung SUB_TOD_TOD subtrahiert einen Wert vom Datentyp TIME_OF_DAY von einem anderen TIME_OF_DAY-Wert. Das Ergebnis wird in der Ausgangsvariable vom Datentyp TIME gespeichert.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von SUB_TOD_TOD (s. S. 1197)**

Datentypen

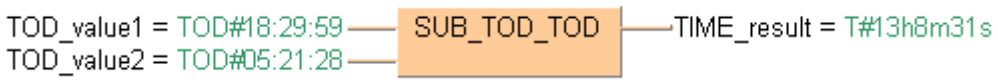
Datentyp	E/A	Funktion
TIME_OF_DAY	Eingang 1	Minuend
TIME_OF_DAY	Eingang 2	Subtrahend
TIME	Ausgang	Ergebnis

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	TOD_value1	TIME_OF_DAY	TOD#18:29:59
1	VAR	TOD_value2	TIME_OF_DAY	TOD#05:21:28
2	VAR	TIME_result	TIME	T#0s

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
TIME_result := SUB_TOD_TOD (TOD_value1, TOD_value2);
```

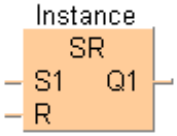
Kapitel 11

Bistabile Funktionsbausteine

SR

Bistabiler FB (vorrangig Setzen)

Erklärung Mit dem Funktionsbaustein SR (set/reset) können Sie einen Ausgang sowohl setzen als auch zurücksetzen.



Symbol:

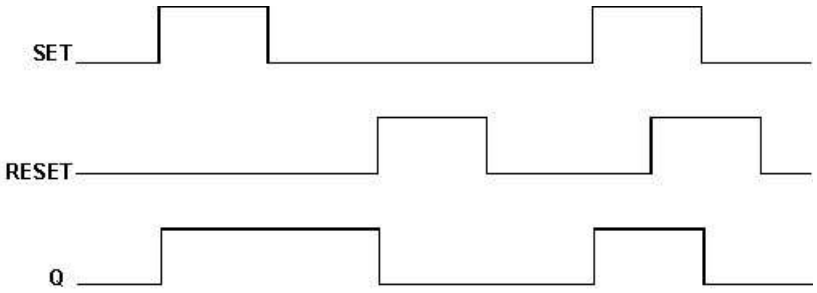
Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Für den SR deklarieren Sie:

SET (S1)	Setzen bei jeder steigenden Flanke an SET wird der Ausgang Q gesetzt
RESET (R)	Zurücksetzen bei jeder steigenden Flanke an RESET wird der Ausgang Q zurückgesetzt, außer wenn SET gesetzt ist (siehe Zeitdiagramm)
Q (Q1)	Signalausgang wird gesetzt, wenn eine steigende Flanke an SET anliegt; wird zurückgesetzt, wenn eine steigende Flanke an RESET anliegt und der SET nicht gesetzt ist

☞ **Die Namen in Klammern geben die gültige Bezeichnung der Parameter des ST-Editors an.**

Zeitdia-gramm:



☞
- **Wenn an beiden Eingängen (Setzen und Zurücksetzen) eine steigende Flanke anliegt, wird Q gesetzt.**
- **Q hat bei der Initialisierung immer den Status Null (zurückgesetzt).**

SPS-Typen **Verfügbarkeit von SR (s. S. 1197)**

Datentypen

Datentypen	E/A	Funktion
BOOL	Eingang 1	Setzen
BOOL	Eingang 2	Zurücksetzen
BOOL	Ausgang	Setzen oder Zurücksetzen abhängig vom Eingang

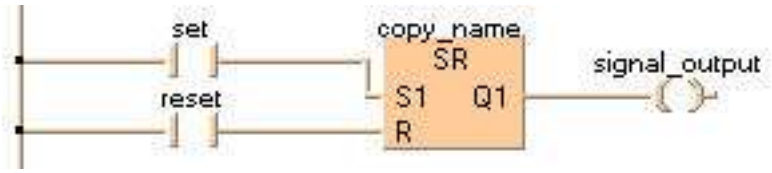
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins SR verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert und ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	copy_name	SR		under this identifier a copy of
1	VAR	set	BOOL	FALSE	set input
2	VAR	reset	BOOL	FALSE	reset input
3	VAR	signal_output	BOOL	FALSE	

Rumpf Wenn **set** gesetzt ist (Status = TRUE), wird **signal_output** gesetzt. Wenn nur **reset** gesetzt ist, wird **signal_output** zurückgesetzt (Status = FALSE). Wenn sowohl **set** als auch **reset** gesetzt sind, wird **signal_output** gesetzt.

KOP



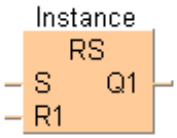
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
copy_name( SET:= set, RESET:= reset);
        signal_output:= signal_output;
```

RS

Bistabiler FB (vorrangig Zurücksetzen)

Erklärung Mit dem Funktionsbaustein RS (reset/set) können Sie einen Ausgang sowohl zurücksetzen als auch setzen.



Symbol:

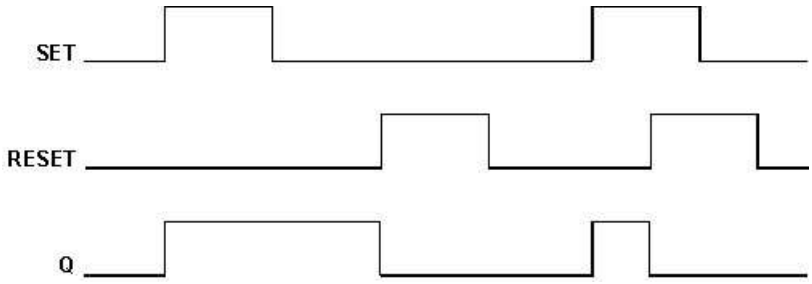
Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Für den RS deklarieren Sie:

SET (S1)	Setzen bei jeder steigenden Flanke an SET wird der Ausgang Q gesetzt, wenn RESET nicht gesetzt ist
RESET (R)	Zurücksetzen bei jeder steigenden Flanke an RESET wird der Ausgang Q zurückgesetzt
Q (Q1)	Signalausgang wird gesetzt, wenn eine steigende Flanke an SET anliegt und RESET nicht gesetzt ist; wird zurückgesetzt, wenn eine steigende Flanke an RESET anliegt

☞ **Die Namen in Klammern geben die gültige Bezeichnung der Parameter des ST-Editors an.**

Zeitdiagramm:



☞ **Wenn an beiden Eingängen eine steigende Flanke anliegt, wird Q zurückgesetzt.**

SPS-Typen **Verfügbarkeit von RS (s. S. 1196)**

Datentypen

Datentypen	E/A	Funktion
BOOL	Eingang 1	Setzen
BOOL	Eingang 2	Zurücksetzen
BOOL	Ausgang	Setzen oder Zurücksetzen abhängig vom Eingang

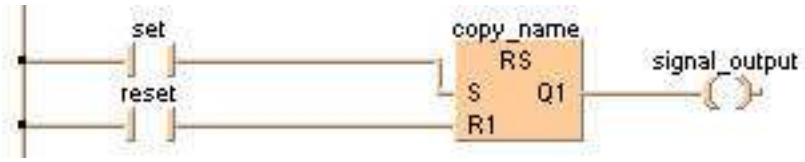
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins RS verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert und ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	copy_name	RS		under this identifier a copy of
1	VAR	set	BOOL	FALSE	set input
2	VAR	reset	BOOL	FALSE	reset input
3	VAR	signal_output	BOOL	FALSE	

Rumpf Wenn **set** gesetzt ist (Status = TRUE), wird **signal_output** gesetzt. Wenn nur **reset** gesetzt ist, wird **signal_output** zurückgesetzt (Status = FALSE). Wenn sowohl **set** als auch **reset** gesetzt sind, wird **signal_output** auf FALSE zurückgesetzt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
copy_name( SET:= set, RESET:= reset);
        signal_output:= signal_output;
```

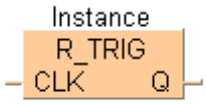
Kapitel 12

Flankenerkennung

R_TRIG

Erkennen einer steigenden Flanke

Erklärung Mit dem Funktionsbaustein R_TRIG (rising edge trigger) können Sie an einem Eingang eine steigende Flanke erkennen.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Für den R_TRIG deklarieren Sie:

CLK	**Signaleingang** bei jeder steigenden Flanke am Signaleingang wird der Ausgang Q gesetzt (clk = clock = Takt)
Q	**Signalausgang** wird gesetzt, wenn eine steigende Flanke an CLK anliegt

SPS-Typen **Verfügbarkeit von R_TRIG (s. S. 1195)**

☞ **Der Ausgang Q eines R_TRIG Funktionsbausteins bleibt nach dem Auftreten einer steigenden Flanke (Statuswechsel FALSE -> TRUE) am CLK-Eingang einen SPS-Zyklus lang gesetzt und wird im darauf folgenden Zyklus wieder zurückgesetzt.**

Datentypen

Datentypen	E/A	Funktion
BOOL	Eingang CLK	Erkennt steigende Flanke
BOOL	Ausgang Q	Gesetzt, falls steigende Flanke am Eingang erkannt wird

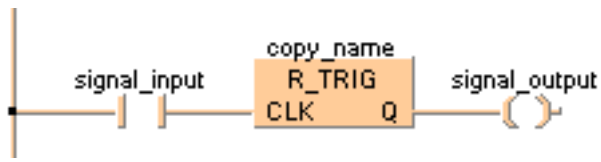
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins R_TRIG verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert und ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial
0	VAR	copy_name	R_TRIG	
1	VAR	signal_input	BOOL	FALSE
2	VAR	signal_output	BOOL	FALSE

Rumpf Wenn an **signal_input** eine steigende Flanke anliegt, wird **signal_output** gesetzt.

KOP



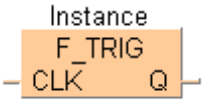
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
copy_name ( CLK := signal_input ,
            Q => signal_output );
```

F_TRIG

Erkennen einer fallenden Flanke

Erklärung Mit dem Funktionsbaustein F_TRIG (falling edge trigger) können Sie an einem Eingang eine fallende Flanke erkennen.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Für den F_TRIG deklarieren Sie:

CLK	**Signaleingang** bei jeder fallenden Flanke am Signaleingang wird der Ausgang Q gesetzt (clk = clock = Takt)
Q	**Signalausgang** wird gesetzt, wenn eine fallende Flanke an CLK anliegt

SPS-Typen **Verfügbarkeit von F_TRIG (s. S. 1186)**

☞ **Der Ausgang Q eines F_TRIG Funktionsbausteins bleibt nach dem Auftreten einer fallenden Flanke (Statuswechsel TRUE -> FALSE) am CLK-Eingang einen SPS-Zyklus lang gesetzt und wird im darauf folgenden Zyklus wieder zurückgesetzt.**

Datentypen

Datentypen	E/A	Funktion
BOOL	Eingang CLK	Erkennt fallende Flanke am Eingang CLK
BOOL	Ausgang Q	Wird gesetzt, wenn eine fallende Flanke an CLK anliegt

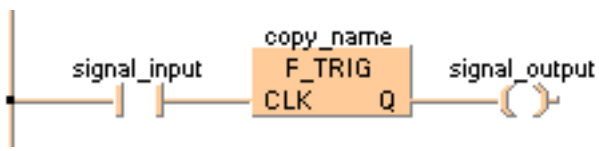
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins F_TRIG verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert und ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial
0	VAR	copy_name	F_TRIG	
1	VAR	signal_input	BOOL	FALSE
2	VAR	signal_output	BOOL	FALSE

Rumpf Wenn an **signal_input** eine fallende Flanke anliegt, wird **signal_output** gesetzt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
copy_name( CLK:= signal_input ,
           Q=> signal_output );
```

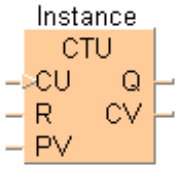
Kapitel 13

Zähler

CTU

Aufwärtszähler

Erklärung Mit dem Funktionsbaustein CTU (count up) können Sie Zählvorgänge programmieren.



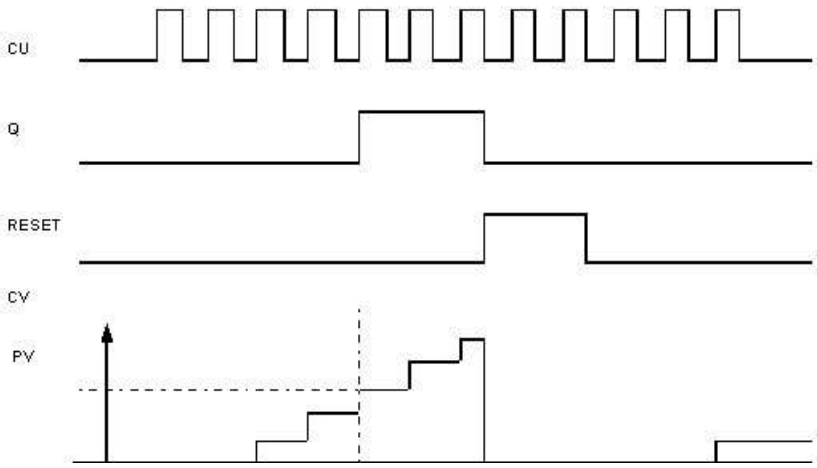
Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Für den CTU deklarieren Sie:

CU	**Taktgeber** bei jeder steigenden Flanke an CU wird der Wert 1 zu CV addiert, außer RESET ist gesetzt
RESET (R)	**Zurücksetzen** bei jeder steigenden Flanke an RESET wird CV auf Null zurückgesetzt
PV	**Sollwert** wenn PV (preset value) erreicht wird, wird Q gesetzt
Q	**Signalausgang** wird gesetzt, wenn CV größer/gleich PV
CV	**Istwert** enthält das Additionsergebnis (CV = current value)

☞ **Die Namen in Klammern geben die gültige Bezeichnung der Parameter des ST-Editors an.**

Zeitdiagramm:



SPS-Typen **Verfügbarkeit von CTU (s. S. 1185)**

Datentypen

Datentypen	E/A	Funktion
BOOL	Eingang CU	Erkennt steigende Flanke, addiert 1 zu CV
BOOL	Eingang RESET	Setzt CV an der steigenden Flanke zurück auf 0
INT	Eingang PV	Sollwert
BOOL	Ausgang Q	Gesetzt, falls CV >= PV
INT	Ausgang CV	Istwert

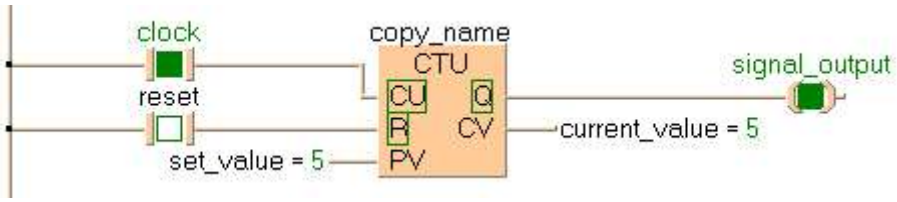
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins CTU verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert. Für diese Kopie wird ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	copy_name	CTU		under this identifier a copy of the function block CTU is declared
1	VAR	clock	BOOL	FALSE	up counter input
2	VAR	reset	BOOL	FALSE	reset input (reset to 0)
3	VAR	set_value	INT	5	default (PV=preset value)
4	VAR	signal_output	BOOL	FALSE	
5	VAR	current_value	INT	0	current counter value (EV=elapsed value)

Rumpf Wenn **reset** gesetzt ist (Status = TRUE), wird **current_value** (CV) zurückgesetzt. Wenn an **clock** eine steigende Flanke erkannt wird, wird der Wert 1 zu **current_value** addiert. Vorgang wird bei jeder steigenden Flanke an **clock** so oft wiederholt, bis **current_value** größer/gleich **set_value** ist. Dann wird **signal_output** gesetzt.

KOP



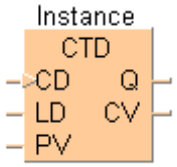
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
copy_name ( CU := clock, RESET := reset, PV := set_value, Q => signal_output, CV =>
current_value );
```

CTD

Abwärtszähler

Erklärung Mit dem Funktionsbaustein CTD (count down) können Sie Zählvorgänge programmieren.



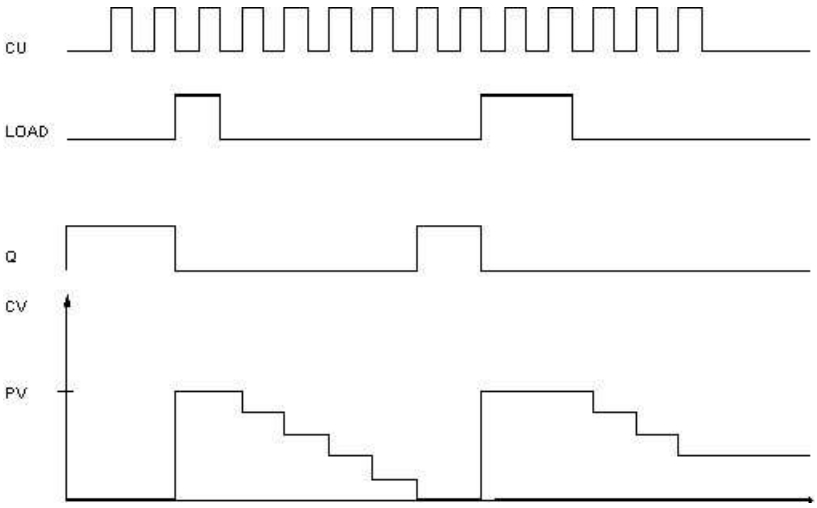
Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Für den CTD deklarieren Sie:

CD	**Taktgeber-Eingang** bei jeder steigenden Flanke an CD wird der Wert 1 vom aktuellen Wert in CV abgezogen, außer LOAD ist gesetzt oder CV hat den Wert Null erreicht
LOAD (LD)	**Setzen** mit LOAD wird der Zählerstand auf PV zurückgesetzt
PV	**Ausgangswert** ist der Wert, von dem beim ersten Zählvorgang abgezogen wird
Q	**Signalausgang** wird gesetzt, wenn CV = Null
CV	**Istwert** enthält das aktuelle Subtraktionsergebnis (CV = current value)

☞ **Die Namen in Klammern geben die gültige Bezeichnung der Parameter des ST-Editors an.**

Zeit-diagramm:



SPS-Typen **Verfügbarkeit von CTD (s. S. 1185)**

Datentypen

Datentypen	E/A	Funktion
BOOL	Eingang CD	Subtrahiert 1 von CV an der steigenden Flanke
BOOL	Eingang LOAD	Setzt Zähler zurück auf PV
INT	Eingang PV	Ausgangswert
BOOL	Ausgang Q	Signalausgang gesetzt, falls CV = PV
INT	Ausgang CV	Istwert

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

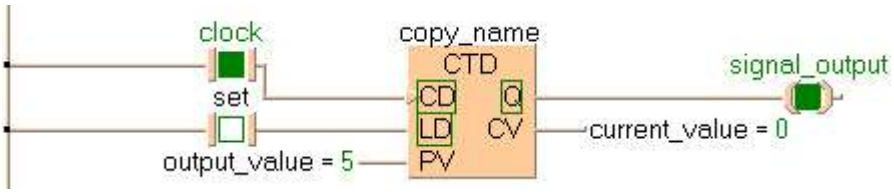
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins CTD verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert und ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	copy_name	CTD		under this identifier a copy of the function block ...
1	VAR	clock	BOOL	FALSE	down counter input
2	VAR	set	BOOL	FALSE	set input (set to preset value (PV))
3	VAR	output_value	INT	0	minuend
4	VAR	signal_output	BOOL	FALSE	
5	VAR	current_value	INT	0	current counter value

Rumpf

Wenn **set** gesetzt ist (Status = TRUE), wird **preset_value** (PV) in den **current_value** (CV) geladen. Jedesmal, wenn an **clock** eine steigende Flanke anliegt, wird von **current_value** der Wert 1 abgezogen. Dieser Vorgang wird so oft wiederholt, bis **current_value** kleiner/gleich Null ist. Dann wird **signal_output** gesetzt.

KOP



ST

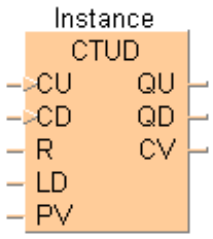
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF set THEN        (* first cycle *)
     load:=TRUE;          (* load has to be TRUE,
                           to set current_value to output_value *)
     clock:=FALSE;
END_IF;
copy_name(CD:= clock, LOAD:= set, PV:= output_value, Q=> signal_output, CV=>
current_value);
load:=FALSE;      (* now current_value got the right value, load doesn't need
to be *)
                  (* TRUE any longer *);
```

CTUD

Auf-/Abwärtszähler

Erklärung Mit dem Funktionsbaustein CTUD (count up/down) können Sie Zählvorgänge (auf- und abwärts) programmieren.



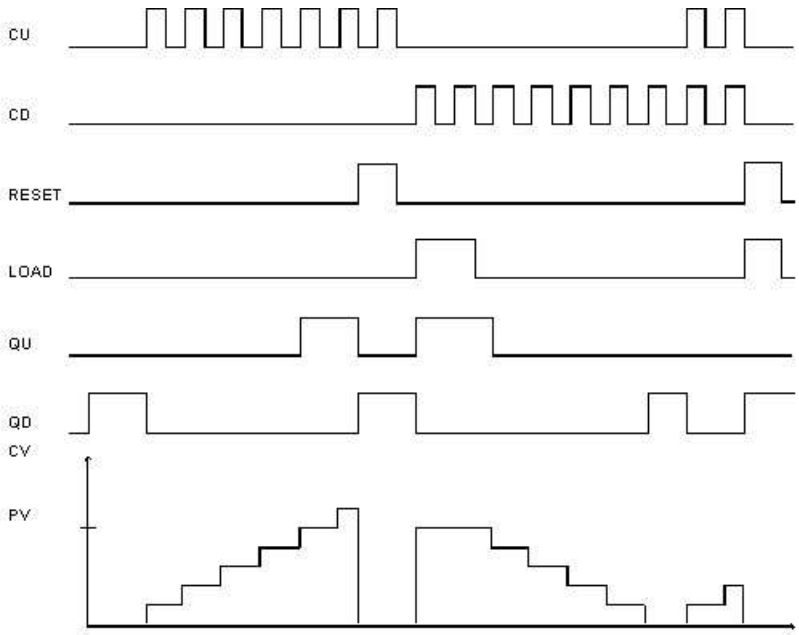
Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Für den CTUD deklarieren Sie:

CU	**Aufwärtszähler** bei jeder steigenden Flanke an CU wird der Wert 1 zum aktuellen CV addiert, außer RESET und/oder LOAD ist/sind gesetzt
CD	**Abwärtszähler** bei jeder steigenden Flanke an CD wird der Wert 1 vom aktuellen CV abgezogen, außer RESET und/oder LOAD ist/sind gesetzt und wenn CU und CD gleichzeitig gesetzt sind. Im letzten Fall wird aufwärtsgezählt.
RESET (R)	**Zurücksetzen** wenn RESET gesetzt wird, wird CV zurückgesetzt
LOAD (LD)	**Setzen** wenn LOAD gesetzt wird, wird PV in CV geladen. Das gilt nicht, wenn RESET gleichzeitig gesetzt wird. In diesem Fall wird LOAD ignoriert.
PV	**Ausgangswert** enthält den Sollwert, der bei der Addition bzw. Subtraktion erreicht werden soll (PV = preset value)
QU	**Signalausgang - Aufwärtszähler** wird gesetzt, wenn CV größer/gleich PV
QD	**Signalausgang - Abwärtszähler** wird gesetzt, wenn CV = Null
CV	**Istwert** ist das Additions-/Subtraktionsergebnis (CV = current value)

☞ **Die Namen in Klammern geben die gültige Bezeichnung der Parameter des ST-Editors an.**

Zeitdiagramm:



SPS-Typen **Verfügbarkeit von CTUD (s. S. 1185)**

Datentypen

Datentypen	E/A	Funktion
BOOL	Eingang CU	Aufwärtszähler
BOOL	Eingang CD	Abwärtszähler
BOOL	Eingang RESET	Falls gesetzt, setzt CV zurück
BOOL	Eingang LOAD	Lädt PV nach CV
INT	Eingang PV	Sollwert
BOOL	Ausgang QU	Signalausgang Aufwärtszähler
BOOL	Ausgang QD	Signalausgang Abwärtszähler
INT	Ausgang CV	Istwert

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins CTUD verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert. Für diese Kopie wird ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial	Kommentar
	VAR	copy_name	CTUD		under this identifier a copy of the function block CTUD is declared
	VAR	up_clock	BOOL	FALSE	upward counter input
	VAR	down_clock	BOOL	FALSE	downward counter input
	VAR	reset	BOOL	FALSE	reset input (reset to 0)
	VAR	set	BOOL	FALSE	set input (set to set_value)
	VAR	set_value	INT	0	default
	VAR	output_up	BOOL	FALSE	
	VAR	output_down	BOOL	FALSE	
	VAR	current_value	INT	0	current counter value
	VAR	enable	BOOL	FALSE	

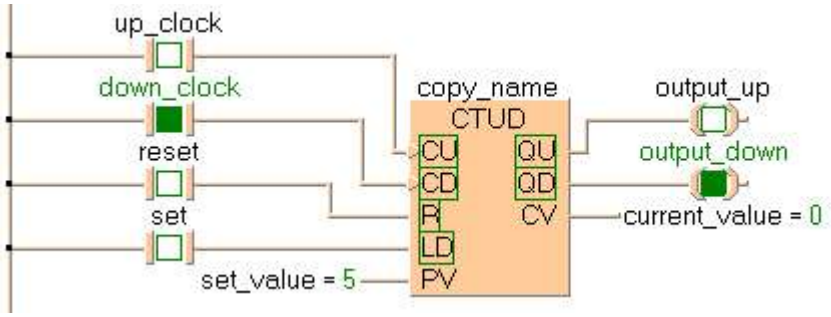
Rumpf Aufwärtszählen:

Wenn **reset** gesetzt ist, wird **current_value** (CV) zurückgesetzt. Wenn **up_clock** gesetzt ist, wird der Wert 1 zu **current_value** addiert. Dieser Vorgang wird bei jeder steigenden Flanke an **up_clock** so oft wiederholt, bis **current value** größer/gleich **set_value** ist. Dann wird **output_up** gesetzt. Der Vorgang wird nicht durchgeführt, wenn **reset** und/oder **set** gesetzt ist/sind.

Abwärtszählen:

Wenn **set** gesetzt ist (Status = TRUE), wird **set_value** in den **current_value** (CV) geladen. Wenn **down_clock** gesetzt ist, wird bei jedem Takt von **set_value** der Wert 1 abgezogen. Dieser Vorgang wird bei jedem Takt so oft wiederholt, bis **current_value** kleiner/gleich Null ist. Dann wird **signal_output** gesetzt. Der Vorgang wird nicht durchgeführt, wenn **reset** und/oder **set** gesetzt ist/sind bzw. wenn CU und CV gleichzeitig gesetzt sind. Im letzten Fall wird abwärtsgezählt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

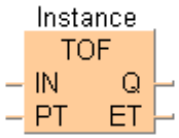
```
copy_name(CU:= up_clock, CD:= down_clock, RESET:= reset, LOAD:= set, PV:=
set_value,
                QU=> output_up, QD=> output_down, CV=> current_value);
```

Kapitel 14

Zeitgeber

TOF Ausschaltverzögerung

Erklärung Mit dem Funktionsbaustein TOF können Sie eine Ausschaltverzögerung programmieren, z.B. den Lüfter für ein Gerät später abschalten als das Gerät selbst.

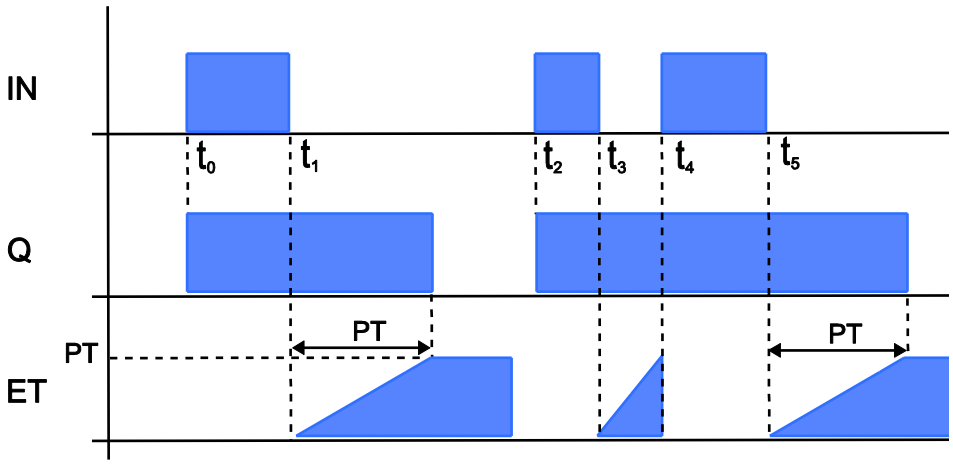


Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Für den TOF deklarieren Sie:

IN	**Zeitgeber EIN** bei einer fallenden Flanke an IN wird ein interner Zeitmesser gestartet. Wenn eine steigende Flanke an IN anliegt, bevor PT seinen Wert erreicht hat, wird Q nicht abgeschaltet
PT	**Ausschaltverzögerung** (16 Bit: 0 - 327,27s, 32 Bit: 0 - 21 474 836,47s; Auflösung jeweils 10ms) hier wird die gewünschte Ausschaltverzögerung definiert (PT = preset time)
Q	**Signalausgang** wird zurückgesetzt, wenn PT = ET
ET	**Istwert** ist die tatsächlich abgelaufene Zeit (ET = elapsed time)

Zeitdia-gramm



Q wird um die Zeit, die in **PT** definiert worden ist, verzögert ausgeschaltet. Das Einschalten verläuft verzögerungsfrei.

Wenn **IN** (wie im Zeitdiagramm oben bei t3 bis t4) wieder gesetzt wird, bevor die Verzögerungszeit **PT** abgelaufen ist, bleibt **Q** gesetzt (Zeitdiagramm bei t2 bis t3).

SPS-Typen **Verfügbarkeit von TOF (s. S. 1198)**

Datentypen

Datentypen	E/A	Funktion
BOOL (IN)	Eingang	Bei fallender Flanke wird ein interner Zeitmesser gestartet
TIME (PT)	Eingang	Ausschaltverzögerung
BOOL (Q)	Ausgang	Signalausgang wird zurückgesetzt, wenn PT = ET
TIME (ET)	Ausgang	Istwert

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

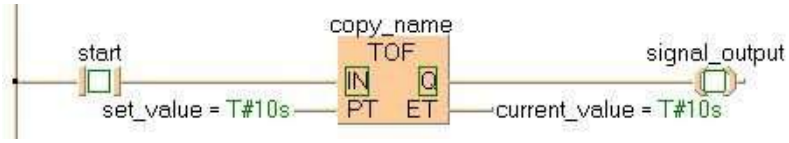
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins TOF verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert. Für diese Kopie wird ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial
0	VAR	copy_name	TOF	
1	VAR	start	BOOL	FALSE
2	VAR	set_value	TIME	T#0s
3	VAR	signal_output	BOOL	FALSE
4	VAR	current_value	TIME	T#0s

Rumpf

Wenn **start** zurückgesetzt wird, wird dieses Signal um die Zeitspanne **set_value** verzögert an **signal_output** weitergegeben.

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

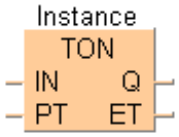
```
copy_name( IN:= start ,
           PT:= set_value ,
           Q=> signal_output ,
           ET=> current_value );
```

TON

Einschaltverzögerung

Erklärung

Mit dem Funktionsbaustein TON können Sie eine Einschaltverzögerung programmieren.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Für den TON deklarieren Sie:

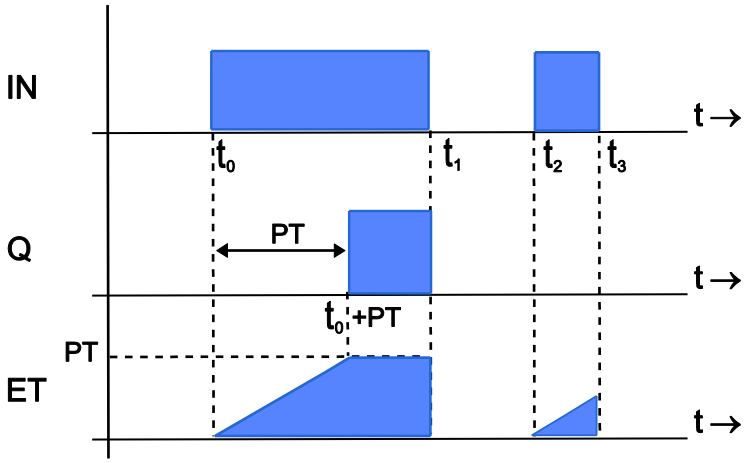
IN **Zeitgeber EIN**
bei jeder steigenden Flanke an IN wird ein interner Zeitmesser gestartet

PT **Einschaltverzögerung**
(16 Bit: 0 - 327,27s, 32 Bit: 0 - 21 474 836,47s; Auflösung jeweils 10ms) hier wird die gewünschte Einschaltverzögerung definiert (PT = preset time)

Q **Signalausgang**
wird gesetzt, wenn PT = ET

ET **Istwert**
ist die tatsächlich abgelaufene Zeit (ET = elapsed time)

Zeitdia-gramm



Q wird um die Zeit, die in **PT** definiert worden ist, verzögert gesetzt. Das Zurücksetzen verläuft verzögerungsfrei.

Wenn der Eingang **IN** nur so lange oder gar kürzer gesetzt wird, als die Verzögerungszeit **PT** dauert (t3 - t2 < PT), dann wird **Q** nicht gesetzt.

SPS-Typen **Verfügbarkeit von TON (s. S. 1198)**

Datentypen

Datentypen	E/A	Funktion
BOOL (IN)	Eingang	Bei steigender Flanke wird ein interner Zeitmesser gestartet
TIME (PT)	Eingang	Einschaltverzögerung
BOOL (Q)	Ausgang	Signalausgang wird gesetzt, wenn PT = ET
TIME (ET)	Ausgang	Istwert

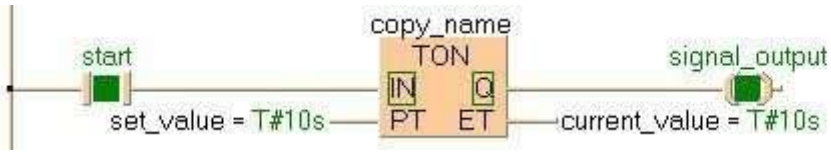
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins TON verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert. Für diese Kopie wird ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial
0	VAR	copy_name	TON	
1	VAR	start	BOOL	FALSE
2	VAR	set_value	TIME	T#0s
3	VAR	signal_output	BOOL	FALSE
4	VAR	current_value	TIME	T#0s

Rumpf Wenn **start** gesetzt wird (Status = TRUE), wird das Eingangssignal um die Zeitspanne **set_value** verzögert an **signal_output** weitergegeben.

KOP

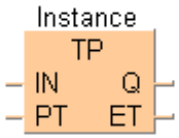


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
copy_name( IN:= start ,
           PT:= set_value ,
           Q=> signal_output ,
           ET=> current_value );
```

TP Impuls-Zeitgeber

Erklärung Mit dem Funktionsbaustein TP können Sie einen Impulsgeber mit einer definierten Impulsdauer programmieren.



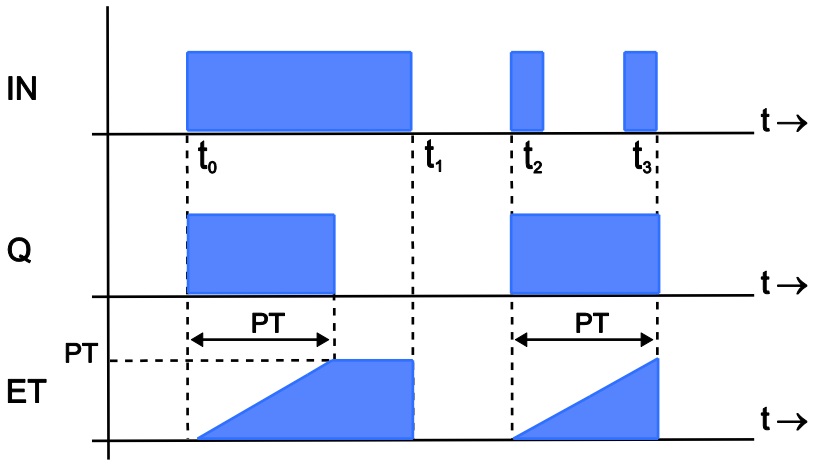
Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Für den TP deklarieren Sie:

IN	**Impulsgeber** bei einer steigenden Flanke an IN wird ein Impuls von der Dauer erzeugt, die in PT definiert worden ist
PT	**Impulsdauer** (16 Bit: 0 - 327,27s, 32 Bit: 0 -21 474 836,47s; Auflösung jeweils 10ms) jeder steigenden Flanke an **IN** wird ein Impuls von der Dauer **PT** ausgelöst. der Impulsdauer löst auch eine neue steigende Flanke an **IN** keinen neuen Impuls aus
Q	**Signalausgang** wird für die Dauer von PT gesetzt, sobald eine steigende Flanke an **IN** anliegt
ET	**Istwert** enthält die abgelaufene Impulsdauer. Wenn **PT** = **ET**, wird **Q** zurückgesetzt

☞ **FP2, FP2SH und FP10SH benötigen für PT den 32-Bit Wert.**

Zeitdia-gramm



Unabhängig von der Einschaltdauer des IN-Signals, wird am Ausgang Q ein Impuls von der Länge erzeugt, der in PT definiert worden ist. Der Funktionsbaustein TP wird gestartet (getriggert), wenn am Eingang IN eine steigende Flanke anliegt.

Während PT abgearbeitet wird, hat eine steigende Flanke am Eingang IN keinen Einfluss.

SPS-Typen **Verfügbarkeit von TP (s. S. 1198)**

Datentypen

Datentypen	E/A	Funktion
BOOL	Eingang IN	Bei steigender Flanke wird ein Impuls mit definierter Dauer erzeugt
TIME	Eingang PT	Impulsdauer
BOOL	Ausgang Q	Signalausgang
TIME	Ausgang ET	Istwert

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

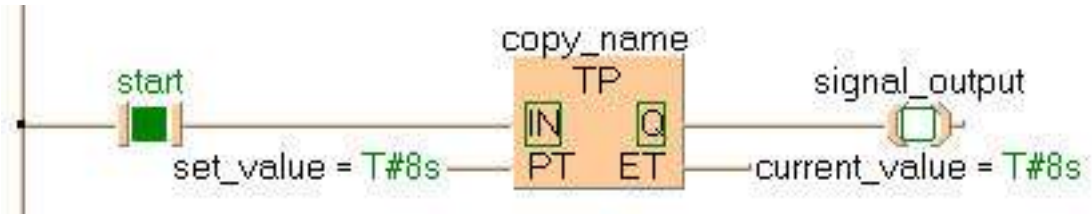
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins TP verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert. Für diese Kopie wird ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial
0	VAR	copy_name	TP	
1	VAR	start	BOOL	FALSE
2	VAR	set_value	TIME	T#0s
3	VAR	signal_output	BOOL	FALSE
4	VAR	current_value	TIME	T#0s

Rumpf

Wenn **start** gesetzt ist (Status = TRUE), wird der Impuls an **signal_output** so lange ausgegeben, bis **set_value** für die Impulsdauer erreicht worden ist.

KOP



ST

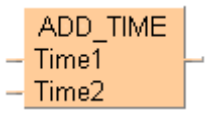
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
copy_name( IN:= start ,
           PT:= set_value ,
           Q=> signal_output ,
           ET=> current_value );
```

ADD_TIME

Zeiten addieren

Erklärung ADD_TIME addiert die Zeiten der beiden Eingangsvariablen und schreibt die Summe in die Ausgangsvariable.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von ADD_TIME (s. S. 1184)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang 1	Augend
TIME	Eingang 2	Summand
TIME	Ausgang	Summe

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	time_value_1	TIME	T#0s
1	VAR	time_value_2	TIME	T#0s
2	VAR	time_value_3	TIME	T#0s

In diesem Beispiel wurden die Eingangsvariablen (**time_value_1** und **time_value_2**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Time_value_1** und **time_value_2** werden addiert. Das Ergebnis wird in **time_value_3** geschrieben.

KOP

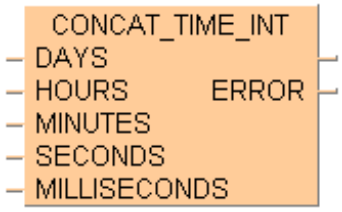
ST
```
time_value_3:=ADD_TIME(time_value_1, time_value_2);
```

CONCAT_TIME_INT

INT-Werte verketten, um eine Uhrzeit zu bilden

Erklärung

CONCAT_TIME_INT verkettet die INTEGER-Werte für Tag, Stunde, Minute, Sekunde und Millisekunde.Das Ergebnis wird in der Ausgangsvariable vom Datentyp TIME gespeichert. Der Boolesche ERROR-Ausgang ist gesetzt, wenn die Eingangswerte ungültige Datums- oder Zeitwerte sind. Die höchste Zeiteinheit ungleich Null kann ihre scheinbare Grenze übersteigen, z.B. ist T#25h ein gültiger Zeitwert und T#1d25h nicht.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von CONCAT_TIME_INT (s. S. 1184)

Datentypen

Datentyp	E/A	Funktion
INT	Eingang 1 Eingang 2 Eingang 3 Eingang 4 Eingang 5	Tage Stunden Minuten Sekunden Millisekunden
TIME	Ausgang	Ergebnis
BOOL	Ausgang	Der Boolesche ERROR-Ausgang ist gesetzt, wenn die Eingangswerte ungültige Datums- oder Zeitwerte sind.

Beispiel

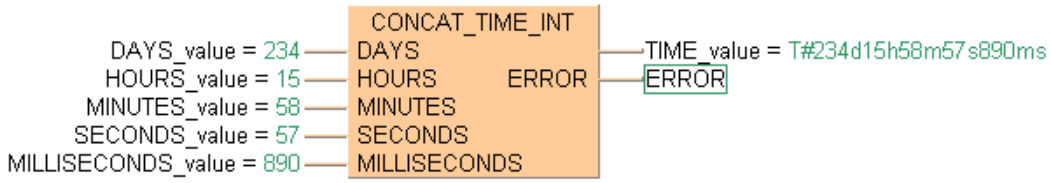
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	TIME_value	TIME	T#0s
1	VAR	DAYS_value	INT	234
2	VAR	HOURS_value	INT	15
3	VAR	MINUTES_value	INT	58
4	VAR	SECONDS_value	INT	57
5	VAR	MILLISECONDS_value	INT	890
6	VAR	ERROR	BOOL	FALSE

KOP



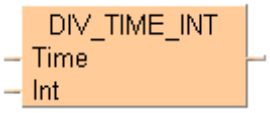
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
TIME_value := CONCAT_TIME_INT(DAYS := DAYS_value,
                              HOURS := HOURS_value,
                        MINUTES := MINUTES_value,
                        SECONDS := SECONDS_value,
                        MILLISECONDS := MILLISECONDS_value,
                        ERROR => ERROR);
```

DIV_TIME_INT

INTEGER Zeiten dividieren

Erklärung DIV_TIME_INT dividiert den Wert der ersten Eingangsvariablen durch den Wert der zweiten und schreibt das Ergebnis in die Ausgangsvariable.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DIV_TIME_INT (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang 1	Dividend
INT	Eingang 2	Divisor
TIME	Ausgang	Ergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

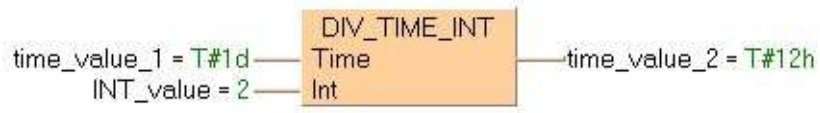
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	time_value_1	TIME	T#0s
1	VAR	time_value_2	TIME	T#0s
2	VAR	INT_value	INT	0

In diesem Beispiel wurden die Eingangsvariablen (**time_value_1** und **INT_value**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Time_value_1** wird durch **INT_value** geteilt. Das Ergebnis wird in **time_value_2** geschrieben.

KOP



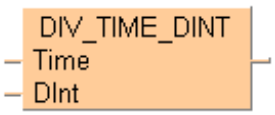
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
time_value_2:=DIV_TIME_INT(time_value_1, INT_value);
```

DIV_TIME_DINT

DOUBLE INTEGER Zeiten dividieren

Erklärung DIV_TIME_DINT dividiert den Wert der ersten Eingangsvariablen durch den Wert der zweiten und schreibt das Ergebnis in die Ausgangsvariable.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DIV_TIME_DINT (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang 1	Dividend
DINT	Eingang 2	Divisor
TIME	Ausgang	Ergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	time_value_1	TIME	T#2h	
1	VAR	time_value_2	TIME	T#0s	result: T#20m
2	VAR	DINT_value	DINT	6	

In diesem Beispiel wurden die Eingangsvariablen (**time_value_1** und **DINT_value**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Time_value_1** wird durch **DINT_value** geteilt. Das Ergebnis wird in **time_value_2** geschrieben.

KOP

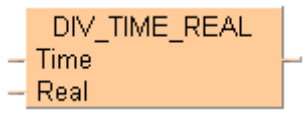
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
time_value_2:=DIV_TIME_DINT(time_value_1, INT_value);
```

DIV_TIME_REAL

REAL Zeiten dividieren

Erklärung Mit DIV_TIME_REAL wird der erste Eingangsvariablenwert vom Typ TIME durch den zweiten Eingangsvariablenwert vom Typ REAL geteilt. Der REAL-Wert wird auf eine ganze Zahl auf- bzw. abgerundet. Das Ergebnis wird in die Ausgangsvariable geschrieben.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von DIV_TIME_REAL (s. S. 1185)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang 1	Dividend
REAL	Eingang 2	Divisor
TIME	Ausgang	Ergebnis

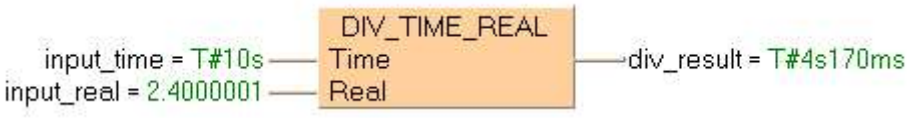
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	input_time	TIME	T#10s
1	VAR	input_real	REAL	2.4
2	VAR	div_result	TIME	T#0s

Rumpf Der Wert der Variablen **input_time** wird durch den Wert der Variablen **input_real** geteilt. Das Ergebnis wird in **div_result** geschrieben. In diesem Beispiel wurden die Eingangsvariablen im POE-Kopf deklariert. Statt dessen können Sie im Rumpf Konstanten direkt an die Eingänge der Funktion schreiben.

KOP



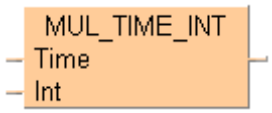
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
div_result:=DIV_TIME_REAL(input_time, input_real);
```

MUL_TIME_INT

Zeiten multiplizieren

Erklärung MUL_TIME_INT multipliziert die Werte der beiden Eingangsvariablen miteinander und schreibt das Ergebnis in die Ausgangsvariable.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von MUL_TIME_INT (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang 1	Multiplikand
INT	Eingang 2	Multiplikator
TIME	Ausgang	Ergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

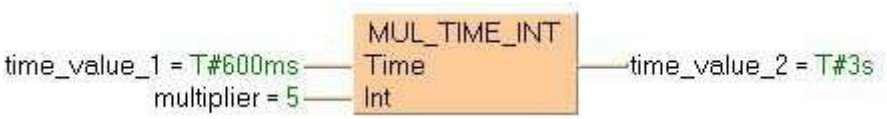
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	time_value_1	TIME	T#0s
1	VAR	multiplier	INT	0
2	VAR	time_value_2	TIME	T#0s

In diesem Beispiel wurden die Eingangsvariablen (**time_value_1** und **multiplier**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Time_value_1** wird mit **multiplier** multipliziert. Das Ergebnis wird in **time_value_2** geschrieben.

KOP



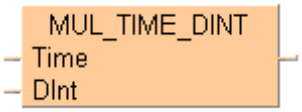
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
time_value_2:=MUL_TIME_INT(time_value_1, multiplier);
```

MUL_TIME_DINT

DOUBLE INTEGER Zeiten multiplizieren

Erklärung MUL_TIME_DINT multipliziert die Werte der beiden Eingangsvariablen miteinander und schreibt das Ergebnis in die Ausgangsvariable.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von MUL_TIME_DINT (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang 1	Multiplikand
DINT	Eingang 2	Multiplikator
TIME	Ausgang	Ergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

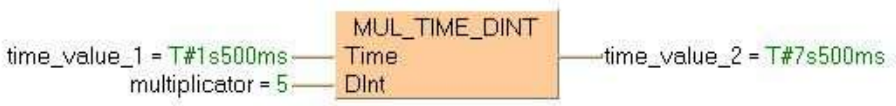
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	time_value_1	TIME	T#1s500ms	
1	VAR	multiplier	DINT	5	
2	VAR	time_value_2	TIME	T#0s	result: T#7s500ms

In diesem Beispiel wurden die Eingangsvariablen **time_value** und **multiplier** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **time_value_1** wird mit **multiplier** multipliziert. Das Ergebnis wird in **time_value_2** geschrieben.

KOP



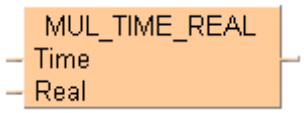
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
time_value_2 :=MUL_TIME_DINT (time_value_1, multiplier);
```

MUL_TIME_REAL

REAL Zeiten multiplizieren

Erklärung MUL_TIME_REAL multipliziert die Werte der beiden Eingangsvariablen miteinander. Der REAL-Wert wird auf eine ganze Zahl auf- bzw. abgerundet. Das Ergebnis wird in die Ausgangsvariable geschrieben.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von MUL_TIME_REAL (s. S. 1194)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang 1	Multiplikand
REAL	Eingang 2	Multiplikator
TIME	Ausgang	Ergebnis

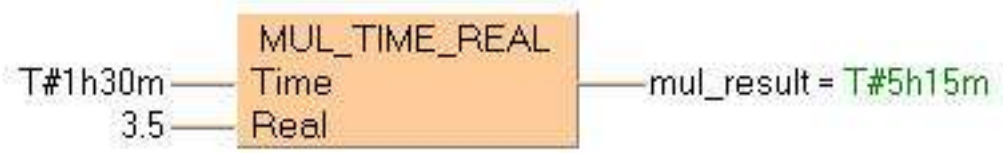
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	mul_result	TIME	T#0s

Rumpf Die Konstante **T#1h30m** wird mit dem Wert **3,5** multipliziert. Das Ergebnis wird in **mul_result** geschrieben. Wenn Sie das Fernglassymbol im Online-Modus anklicken, können Sie sofort das Ergebnis **T#6h0m0s0.00ms** sehen.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

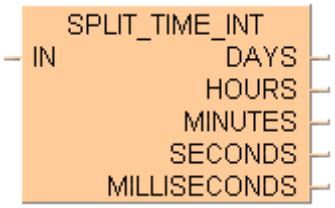
```
mul_result:=MUL_TIME_REAL(T#1h30m, 3.5);
```

SPLIT_TIME_INT

TIME in INT-Werte aufteilen

Erklärung

SPLIT_TIME_INT teilt einen Wert vom Datentyp TIME in INT -Werte für Tage, Stunden, Minuten, Sekunden und Millisekunden auf. Die höchste Zeiteinheit ungleich Null kann ihre scheinbare Grenze übersteigen, z.B. ist T#25h ein gültiger Zeitwert und T#1d25h nicht.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von SPLIT_TIME_INT (s. S. 1196)

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang	Dauer
INT	Ausgang 1 Ausgang 2 Ausgang 3 Ausgang 4 Ausgang 5	Tage Stunden Minuten Sekunden Millisekunden

Beispiel

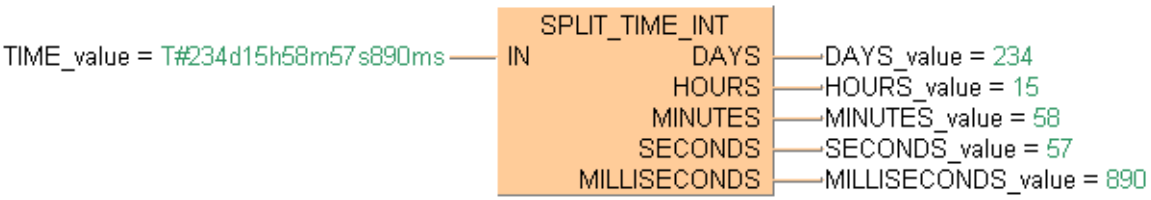
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	TIME_value	TIME	T#234d15h58m57s890ms
1	VAR	DAYS_value	INT	0
2	VAR	HOURS_value	INT	0
3	VAR	MINUTES_value	INT	0
4	VAR	SECONDS_value	INT	0
5	VAR	MILLISECONDS_value	INT	0

KOP



ST

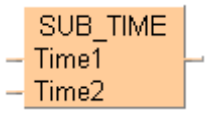
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
SPLIT_TIME_INT(IN := TIME_value,
               DAYS => DAYS_value,
               HOURS => HOURS_value,
               MINUTES => MINUTES_value,
               SECONDS => SECONDS_value,
               MILLISECONDS => MILLISECONDS_value);
```

SUB_TIME

Zeiten subtrahieren

Erklärung SUB_TIME subtrahiert den Wert der zweiten Eingangsvariablen vom Wert der ersten und schreibt das Ergebnis in die Ausgangsvariable.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen **Verfügbarkeit von SUB_TIME (s. S. 1197)**

Datentypen

Datentyp	E/A	Funktion
TIME	Eingang 1	Minuend
TIME	Eingang 2	Subtrahend
TIME	Ausgang	Ergebnis

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

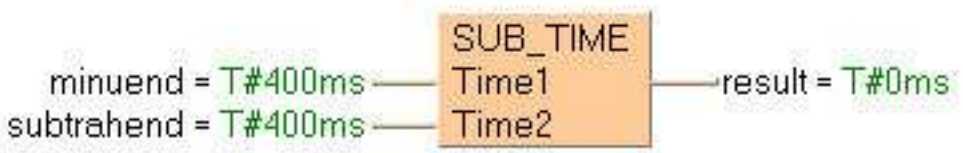
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	minuend	TIME	T#0s
1	VAR	subtrahend	TIME	T#0s
2	VAR	result	TIME	T#0s

In diesem Beispiel wurden die Eingangsvariablen (**minuend** und **subtrahend**) deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf **Subtrahend** wird von **minuend** abgezogen. Das Ergebnis wird in **result** geschrieben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

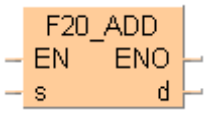
```
result := SUB_TIME(minuend, subtrahend);
```

Kapitel 15

Arithmetikbefehle

F20_ADD 16-Bit-Addition

Erklärung Der 16-Bit-Wert des Speicherregisters **s** oder eine Konstante s und der 16-Bit-Wert des Speicherregisters **d** werden addiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert. Alle 16-Bit-Werte werden als Integer-Werte behandelt.



Beispielwert: 27

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0000	0000	0001	1011

Beispielwert: 16

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0000	0000	0001	0000

Ergebnis: 43 wenn Trigger AN ist

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0000	0000	0010	1011

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden ADD (s. S. 61). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

☞ **Wenn dieser Befehl verwendet wird, wird der Bereich von Augend d vom hinzugefügten Ergebnis überschrieben. Soll das Überschreiben vermieden werden, empfiehlt es sich, den Befehl F22_ADD2 (s. S. 339) zu verwenden.**

SPS-Typen **Verfügbarkeit von F20_ADD s. S. 1189**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	Summand
d		Augend und Ergebnis

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 16-Bit-Daten liegt (Überlauf oder Unterlauf).

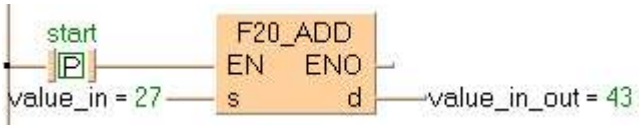
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value_in	INT	27	the value, that will be added
2	VAR	value_in_out	INT	16	result after a 0->1 leading edge from start: 43
3	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

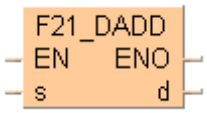


ST
```
IF start THEN
    F20_ADD(value_in, value_in_out);
END_IF;
```

F21_DADD

32-Bit-Addition

Erklärung Der 32-Bit-Wert des Speicherregisters **s** oder eine Konstante s und der 32-Bit-Wert des Speicherregisters **d** werden addiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert. Alle 32-Bit-Werte werden als Double Integer gewertet.



Example value 1312896

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0000	0000	0001	0100	0000	1000	1000	0000

32-bit area

Example value 558144

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0000	0000	0000	1000	1000	0100	0100	0000

Result value 1871040 if trigger is ON

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0000	0000	0001	1100	1000	1100	1100	0000

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden ADD (s. S. 61). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

☞ **Wenn dieser Befehl verwendet wird, wird der Bereich von Augend d vom hinzugefügten Ergebnis überschrieben. Soll das Überschreiben vermieden werden, empfiehlt es sich, den Befehl F23_DADD2 (s. S. 341) zu verwenden.**

SPS-Typen **Verfügbarkeit von F21_DADD (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s	ANY32	Summand
d		Augend und Ergebnis

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 32-Bit-Daten liegt (Überlauf oder Unterlauf).

Beispiel

In diesem Beispiel wird die Funktion F21_DADD im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

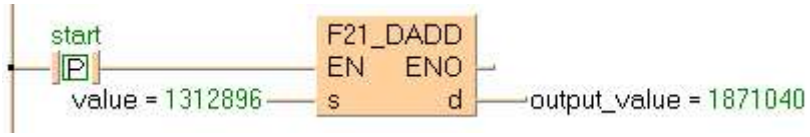
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	value	DINT	27
2	VAR	output_value	DINT	16

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



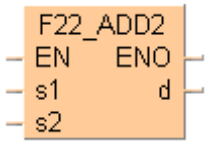
ST

```
IF start THEN
    F21_DADD(value, output_value);
END_IF;
```

F22_ADD2

16-Bit-Addition mit Transfer

Erklärung Der 16-Bit-Wert des Speicherregisters oder die zu 16-Bit äquivalente Konstante, die im 32-Bit-Wert Speicherregister als **s1** und **s2** definiert sind, werden addiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert. Alle 16-Bit-Werte werden als Integer-Werte behandelt.



Beispielwert: 27

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	1 0 1 1

Beispielwert: 16

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 0

Ergebnis: 43 wenn Trigger AN ist

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 1 0	1 0 1 1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden ADD (s. S. 61). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen **Verfügbarkeit von F22_ADD2 (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Augend
s2		Summand
d		Ergebnis

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 16-Bit-Daten liegt (Überlauf oder Unterlauf).

Beispiel

In diesem Beispiel wird die Funktion F22_ADD2 im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

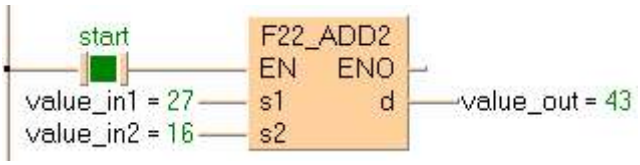
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value_in1	INT	27	
2	VAR	value_in2	INT	16	
3	VAR	value_out	INT	0	result after a 0->1 leading edge from start: 43

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



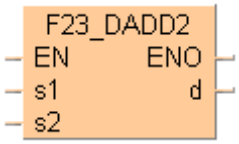
ST

```
IF start THEN
    F22_ADD2(value_in1, value_in2, value_out);
END_IF;
```

F23_DADD2 — 32-Bit-Addition mit Transfer

Erklärung

Der 32-Bit-Wert des Speicherregisters oder die zu 32-Bit äquivalente Konstante, die im 32-Bit-Wert Speicherregister als **s1** und **s2** definiert sind, werden addiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert. Alle 32-Bit-Werte werden als Double Integer gewertet.



Example value 1312896

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0000	0000	0001	0100	0000	1000	1000	0000

32-bit area

Example value 558144

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0000	0000	0000	1000	1000	0100	0100	0000

Result value 1871040 if trigger is ON

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0000	0000	0001	1100	1000	1100	1100	0000

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden ADD (s. S. 61). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen **Verfügbarkeit von F23_DADD2 (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Augend
s2		Summand
d		Ergebnis

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 32-Bit-Daten liegt (Überlauf oder Unterlauf).

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	value_in1	DINT	1312896
2	VAR	value_in2	DINT	558144
3	VAR	value_out	DINT	0

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

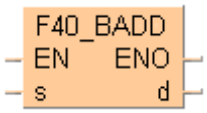
ST

```
IF start THEN
    F23_DADD2(value_in1, value_in2, value_out);
END_IF;
```

F40_BADD

4-Digit-BCD-Addition

Erklärung Der BCD-codierte 16-Bit-Wert des Speicherregisters **s** oder eine Konstante **s** und der BCD-codierte 16-Bit-Wert des Speicherregisters **d** werden addiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 16#2111 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 1 0	0 0 0 1	0 0 0 1	0 0 0 1
16# (BCD)	2	1	1	1

Example value 16#0011 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1
16# (BCD)	0	0	1	1

Result value 16#2122 (BCD) if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 1 0	0 0 0 1	0 0 1 0	0 0 1 0
16# (BCD)	2	1	2	2

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

☞ **Wenn dieser Befehl verwendet wird, wird der Bereich von Augend d vom hinzugefügten Ergebnis überschrieben. Soll das Überschreiben vermieden werden, empfiehlt es sich, den Befehl F41_DBADD (s. S. 345) zu verwenden.**

SPS-Typen **Verfügbarkeit von F40_BADD (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s	WORD	Summand, 16-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
d	WORD	Augend und Ergebnis, 16-Bit-Bereich für BCD-codierte Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn

R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Überlauf).

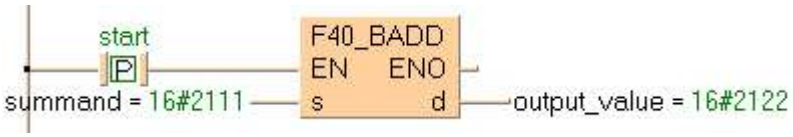
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	summand	WORD	16#2111	this value will be added
2	VAR	output_value	WORD	16#0011	result after 0->1 leading edge from start: 16#2122
3	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

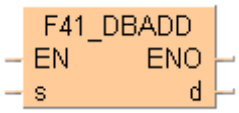


ST

```
IF start THEN
    F40_BADD(summand, output_value);
END_IF;
```

F41_DBADD 8-Digit-BCD-Addition

Erklärung Der BCD-codierte 32-Bit-Wert des Speicherregisters **s** oder eine Konstante **s** und der BCD-codierte 32-Bit-Wert des Speicherregisters **d** werden addiert. Dazu muss der Trigger auf **EN** gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 16#12342000 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0
16# BCD	1	2	3	4	2	0	0	0

32-bit area

Example value 16#00003678 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1	0 1 1 0	0 1 1 1	1 0 0 0
16# BCD	0	0	0	0	3	6	7	8

Result value 16#12345678 (BCD) if trigger is ON

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0
16# BCD	1	2	3	4	5	6	7	8

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

☞ **Wenn dieser Befehl verwendet wird, wird der Bereich von Augend d vom hinzugefügten Ergebnis überschrieben. Soll das Überschreiben vermieden werden, empfiehlt es sich, den Befehl F43_DBADD2 (s. S. 349) zu verwenden.**

SPS-Typen **Verfügbarkeit von F41_DBADD (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s	DWORD	Summand, 32-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
d	DWORD	Augend und Ergebnis, 32-Bit-Bereich für BCD-codierte Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Überlauf).

Beispiel

In diesem Beispiel wird die Funktion F41_DBADD im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	summand	DWORD	16#12342000	this value will be added
2	VAR	output_value	DWORD	16#00003678	result after 0->1 leading edge from start: 16#12345678
3	VAR				

Rumpf

Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

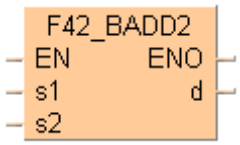
KOP

ST

```
IF DF(start) THEN
    F41_DBADD(summand, output_value);
END_IF;
```

F42_BADD2 — 4-Digit-BCD-Addition mit Transfer

Erklärung Der BCD-codierte 16-Bit-Wert des Speicherregisters **s1** oder eine Konstante **s1** und der BCD-codierte 16-Bit-Wert des Speicherregisters **s2** oder eine Konstante **s s2** werden addiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 16#4321 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s1	0 1 0 0	0 0 1 1	0 0 1 0	0 0 0 1
16# (BCD)	4	3	2	1

Example value 16#1234 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s2	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0
16# (BCD)	1	2	3	4

Result value 16#5555 (BCD) if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1
16# (BCD)	5	5	5	5

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F42_BADD2 (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	WORD	Augend, 16-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
s2	WORD	Summand, 16-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
d	WORD	Summe, 16-Bit-Bereich für BCD-codierte Daten

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Überlauf).

Beispiel

In this example the function F42_BADD2 is programmed in ladder diagram (LD) and structured text (ST). The same POU header is used for all programming languages.

POU Header

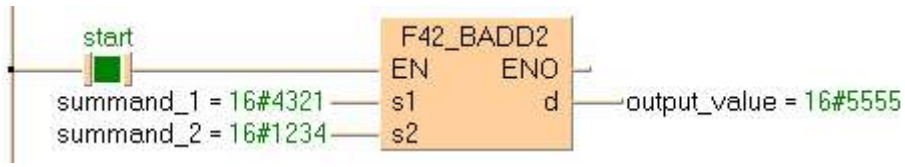
In the POU header, all input and output variables are declared that are used for programming this function.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	summand_1	WORD	16#4321	first summand
2	VAR	summand_2	WORD	16#1234	second summand
3	VAR	output_value	WORD	0	result after a 0->1 leading edge from start: 16#5555
4	VAR				

Body

When the variable **start** changes from FALSE to TRUE, the function is executed.

KOP

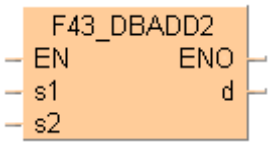


ST

```
IF start THEN
    F42_BADD2(summand_1, summand_2, output_value);
END_IF;
```

F43_DBADD2 — 8-Digit-BCD-Addition mit Transfer

Erklärung Der BCD-codierte 32-Bit-Wert des Speicherregisters **s1** oder eine Konstante **s1** und der BCD-codierte 16-Bit-Wert des Speicherregisters **s2** oder eine Konstante **s2** werden addiert. Dazu muss der Trigger EN auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 16#12345678 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s1	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0
16# BCD	1	2	3	4	5	6	7	8

32-bit area

Example value 16#87654321 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s2	1 0 0 0	0 1 1 1	0 1 1 0	0 1 0 1	0 1 0 0	0 0 1 1	0 0 1 0	0 0 0 1
16# BCD	8	7	6	5	4	3	2	1

Result value 16#99999999 (BCD) if trigger is ON

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	1 0 0 1	1 0 0 1	1 0 0 1	1 0 0 1	1 0 0 1	1 0 0 1	1 0 0 1	1 0 0 1
16# BCD	9	9	9	9	9	9	9	9

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F43_DBADD2 (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	DWORD	Augend, 32-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
s2	DWORD	Summand, 32-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
d	DWORD	Summe, 32-Bit-Bereich für BCD-codierte Daten

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Überlauf).

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

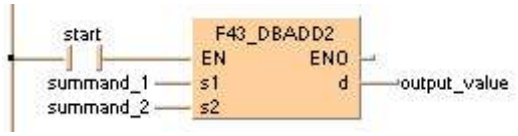
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	summand_1	DWORD	16#12345678	first summand
2	VAR	summand_2	DWORD	16#87654321	second summand
3	VAR	output_value	DWORD	0	result after a 0->1 leading edge from start: 16#99999999
4	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



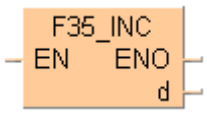
ST

```
IF start THEN
    F43_DBADD2( summand_1, summand_2, output_value);
END_IF;
```

F35_INC

16-Bit-Inkrement

Erklärung Zum 16-Bit-Wert des Speicherregisters **d** wird der Wert "1" addiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 17

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1

Result value 18 if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	0 0 1 0

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F35_INC (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
d	INT, WORD	16-Bit-Datenbereich, der um 1 erhöht wird

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 16-Bit-Daten liegt (Überlauf).

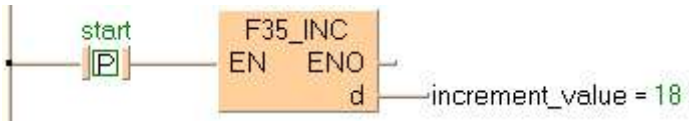
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	increment_value	INT	17	result after a 0->1 leading
2	VAR				edge from start: 18

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

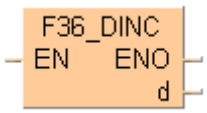


ST
```
IF DF(start) THEN
    F35_INC(increment_value);
END_IF;
```

F36_DINC

32-Bit-Inkrement

Erklärung Zum 32-Bit-Wert des Speicherregisters **d** wird der Wert "1" addiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 131081

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0000	0000	0000	0010	0000	0000	0000	1001

32-bit area

Result value 131082 if trigger is ON

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0000	0000	0000	0010	1000	0000	0000	1010

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F36_DINC (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
d	ANY32	32-Bit-Datenbereich, der um 1 erhöht wird

Operanden

Für	Merker				T/C		Register			Konstante
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 32-Bit-Daten liegt (Überlauf).

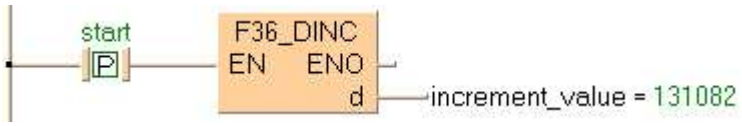
Beispiel In diesem Beispiel wird die Funktion F36_DINC im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	increment_value	DINT	17

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



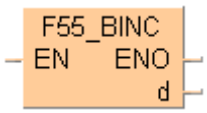
ST

```
IF DF(start) THEN
    F36_DINC(increment_value);
END_IF;
```

F55_BINC

4-Digit-BCD-Inkrementieren

Erklärung Zum BCD-codierten 16-Bit-Wert des Speicherregisters **d** wird der Wert "1" addiert. Dazu muss der Trigger auf **EN** gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 16#4320 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 1 0 0	0 0 1 1	0 0 1 0	0 0 0 0
16# BCD	4	3	2	0

Result value 16#4321 (BCD) if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 1 0 0	0 0 1 1	0 0 1 0	0 0 0 1
16# BCD	4	3	2	1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F55_BINC (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
d	WORD	16-Bit-Datenbereich für 4-stellige BCD-Daten, die um 1 erhöht werden sollen

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Überlauf).

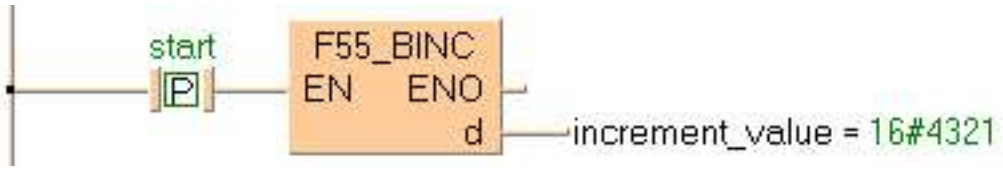
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	increment_value	WORD	16#4320	result after a 0->1 leding edge from start: 16#4321
2	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

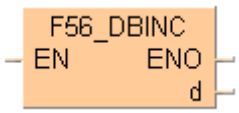


ST
```
IF DF(start) THEN
    F55_BINC(increment_value);
END_IF;
```

F56_DBINC

8-Digit-BCD-Inkrementieren

Erklärung Zum BCD-codierten 16-Bit-Wert des Speicherregisters **d** wird der Wert "1" addiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 16#87654320 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	1 0 0 0	0 1 1 1	0 1 1 0	0 1 0 1	0 1 0 0	0 0 1 1	0 0 1 0	0 0 0 0
16# BCD	8	7	6	5	4	3	2	0

32-bit area

Result value 16#87654321 (BCD) if trigger is ON

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	1 0 0 0	0 1 1 1	0 1 1 0	0 1 0 1	0 1 0 0	0 0 1 1	0 0 1 0	0 0 0 1
16# BCD	8	7	6	5	4	3	2	1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F56_DBINC (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
d	DWORD	32-Bit-Datenbereich für 8-stellige BCD-Daten, die um 1 erhöht werden sollen

Operanden

Für	Merker				T/C		Register			Konstante
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Überlauf).

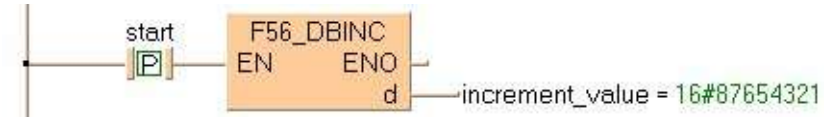
Beispiel In diesem Beispiel wird die Funktion F56_DBINC im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	increment_value	DWORD	16#87654320	result after a 0->1 leding edge from start: 16#87654321
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

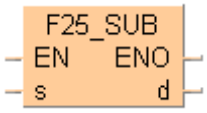
KOP



ST
```
IF DF(start) THEN
    F56_DBINC(increment_value);
END_IF;
```

F25_SUB 16-Bit-Subtraktion

Erklärung Der binäre 16-Bit-Wert des Speicherregisters **s** oder eine Konstante s wird vom 16-Bit-Wert des Speicherregisters **d** subtrahiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis der Subtraktion überschreibt den Inhalt des Speicherregisters **d**. Der Inhalt des Speicherregisters s bleibt erhalten. Alle 16-Bit-Werte werden als Integer-Werte behandelt.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden SUB (s. S. 62).

Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

Example value 16

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	1 0 1 1

Example value 27

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 0

Result value -11 if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	1 1 1 1	1 1 1 1	1 1 1 1	0 1 0 1

SPS-Typen **Verfügbarkeit von F25_SUB (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	Subtrahend
d		Minuend und Ergebnis

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 16-Bit-Daten liegt (Überlauf oder Unterlauf).

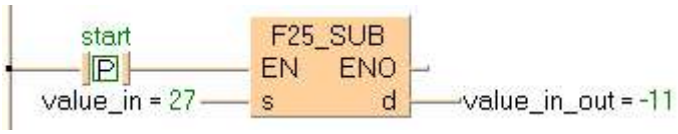
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value_in	INT	27	the value, that will be subtracted
2	VAR	value_in_out	INT	16	result after a 0->1 leading edge from start: -11
3	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

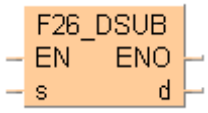
KOP



ST
```
IF start THEN
    F25_SUB(value_in, value_in_out);
END_IF;
```

F26_DSUB 32-Bit-Subtraktion

Erklärung Der binäre 32-Bit-Wert des Speicherregisters **s** oder einer Konstante s wird vom 32-Bit-Wert des Speicherregisters **d** subtrahiert. Dazu muss der Trigger auf **EN** EIN gesetzt sein. Das Ergebnis der Subtraktion überschreibt den Inhalt des Speicherregisters **d**. Der Inhalt des Speicherregisters s bleibt erhalten. Alle 32-Bit-Werte werden als Double Integer gewertet.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden SUB (s. S. 62).

Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

Example value 16778109

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1	0 1 1 1	1 1 0 1

32-bit area

Example value 524740

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	0 0 0 0	1 0 0 0	0 0 0 0	0 0 0 1	1 1 0 0	0 1 0 0

Result value 16253369 if trigger is ON

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	1 1 1 1	1 0 0 0	0 0 0 0	0 0 0 1	1 0 1 1	1 0 0 1

SPS-Typen **Verfügbarkeit von F26_DSUB (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s	ANY32	Subtrahend
d		Minuend und Ergebnis

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 32-Bit-Daten liegt (Überlauf oder Unterlauf).

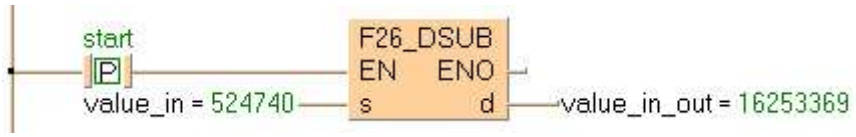
Beispiel In diesem Beispiel wird die Funktion F26_DSUB im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	value_in	DINT	27
2	VAR	value_in_out	DINT	16

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

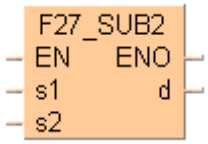


ST
```
IF start THEN
    F26_DSUB(value_in, value_in_out);
END_IF;
```

F27_SUB2

16-Bit-Subtraktion mit Transfer

Erklärung Der 16-Bit-Wert des Speicherregisters **s2** oder eine Konstante s2 wird vom 16-Bit-Wert des Speicherregisters s1 oder von einer Konstante **s1** subtrahiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert. Alle 16-Bit-Werte werden als Integer-Werte behandelt.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden SUB (s. S. 62).

Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

Example value 27

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 0

Example value 16

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	1 0 1 1

Result value 11 if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	0 0 1 1

SPS-Typen **Verfügbarkeit von F27_SUB2 (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Minuend
s2		Subtrahend
d		Ergebnis

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 16-Bit-Daten liegt (Überlauf oder Unterlauf).

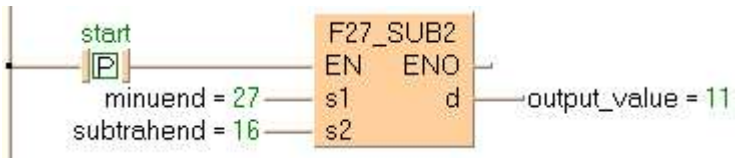
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	minuend	INT	27	minuend
2	VAR	subtrahend	INT	16	subtrahend
3	VAR	output_value	INT	0	result after a 0->1 leading edge from start: 11
4	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

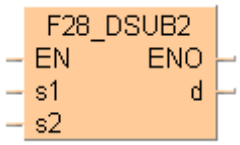


ST
```
IF start THEN
    F27_SUB2(minuend, subtrahend, output_value);
END_IF;
```

F28_DSUB2

32-Bit-Subtraktion mit Transfer

Erklärung Der 32-Bit-Wert des Speicherregisters **s2** oder eine Konstante s2 wird vom 32-Bit-Wert des Speicherregisters s1 oder einer Konstanten **s1** subtrahiert. Dazu muss der Trigger auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert. Alle 32-Bit-Werte werden als Double Integer gewertet.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden SUB (s. S. 62).

Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

Example value 16809984

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s1	0 0 0 0	0 0 0 1	0 0 0 0	0 0 0 0	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

32-bit area

Example value 525312

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s2	0 0 0 0	0 0 0 0	0 0 0 0	1 0 0 0	0 0 0 0	0 1 0 0	0 0 0 0	0 0 0 0

Result value 16284672 if trigger is ON

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	1 1 1 1	1 0 0 0	0 1 1 1	1 1 0 0	0 0 0 0	0 0 0 0

SPS-Typen **Verfügbarkeit von F28_DSUB2 (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Minuend
s2		Subtrahend
d		Ergebnis

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 32-Bit-Daten liegt (Überlauf oder Unterlauf).

Beispiel

In diesem Beispiel wird die Funktion F28_DSUB2 im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	minuend	DINT	27
2	VAR	subtrahend	DINT	16
3	VAR	output_value	DINT	0

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

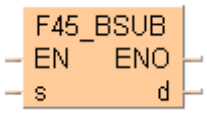
ST

```
IF start THEN
    F28_DSUB2(minuend, subtrahend, output_value);
END_IF;
```

F45_BSUB

4-Digit-BCD-Subtraktion

Erklärung Der BCD-codierte 16-Bit-Wert des Speicherregisters **s** oder eine Konstante **s** wird vom BCD-codierten 16-Bit-Wert des Speicherregisters **d** subtrahiert. Dazu muss der Trigger auf **EN** gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 16#2111 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 1 0	0 0 0 1	0 0 0 1	0 0 0 1
16# (BCD)	2	1	1	1

Example value 16#0011 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	0 0 0 1	0 1 0 1
16# (BCD)	0	0	1	1

Trigger: ON

Result value 16#2100 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 1 0	0 0 0 1	0 0 0 0	0 0 0 0
16# (BCD)	2	1	0	0

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F45_BSUB (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s	WORD	Minuend, 16-Bit-Bereich für vierstellige BCD-Daten oder äquivalente Konstante
d	WORD	Minuend und Ergebnis, 16-Bit-Bereich für BCD-codierte Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Überlauf).

Beispiel

In diesem Beispiel wird die Funktion F45_DSUB im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	subtrahend	WORD	16#0011	this value will be subtracted
2	VAR	output_value	WORD	16#2111	result after 0->1 leading edge from start: 16#2100
3	VAR				

Rumpf

Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

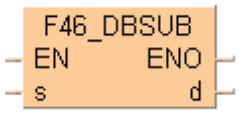
ST

```
IF DF(start) THEN
    F45_BSUB(subtrahend, output_value);
END_IF;
```

F46_DBSUB

8-Digit-BCD-Subtraktion

Erklärung Der achtstellige BCD-codierte 32-Bit-Wert des Speicherregisters **s** oder eine Konstante **s** wird vom BCD-codierte 32-Bit-Wert des Speicherregisters **d** subtrahiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 16#23210044 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 1 0	0 0 1 1	0 0 0 1	0 0 0 1	0 0 0 0	0 0 0 0	0 1 0 0	0 1 0 0
16# BCD	2	3	2	1	0	0	4	4

32-bit area

Example value 16#00210011 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	0 0 1 0	0 0 0 1	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1
16# BCD	0	0	2	1	0	0	1	1

Trigger: ON

Result value 16#23000033 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 1 0	0 0 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1	0 0 1 1
16# BCD	2	3	0	0	0	0	3	3

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F46_DBSUB (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s	DWORD	Subtrahen, 32-Bit-Bereich für achtstellige BCD-Daten oder äquivalente Konstante
d	DWORD	Minuend und Ergebnis, 32-Bit-Bereich für BCD-codierte Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Überlauf).

Beispiel In diesem Beispiel wird die Funktion F46_DBSUB im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	subtrahend	DWORD	16#00210011	this value will be subtracted
2	VAR	output_value	DWORD	16#23210044	result after 0->1 leading edge from start: 16#23000033
3	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

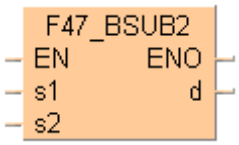
ST

```
IF DF(start) THEN
    F46_DBSUB(subtrahend, output_value);
END_IF;
```

F47_BSUB2

4-Digit-BCD-Subtraktion mit Transfer

Erklärung Der BCD-codierte 16-Bit-Wert des Speicherregisters **s2** oder eine Konstante **s2** wird vom BCD-codierten 16-Bit-Wert des Speicherregisters **s1** oder von einer Konstanten **s1** subtrahiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 16#16 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s1	0 0 0 0	0 0 0 0	0 0 0 1	0 1 1 0
16# (BCD)	0	0	1	6

Example value 16#4 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s2	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0
16# (BCD)	0	0	0	4

Trigger: ON

Result value 16#12 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	0 0 1 0
16# (BCD)	0	0	1	2

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F47_BSUB2 (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	WORD	Minuend, 16-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
s2	WORD	Minuend, 16-Bit-Bereich für vierstellige BCD-Daten oder äquivalente Konstante
d	WORD	Ergebnis, 16-Bit-Bereich für BCD-codierte Daten

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Überlauf).

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

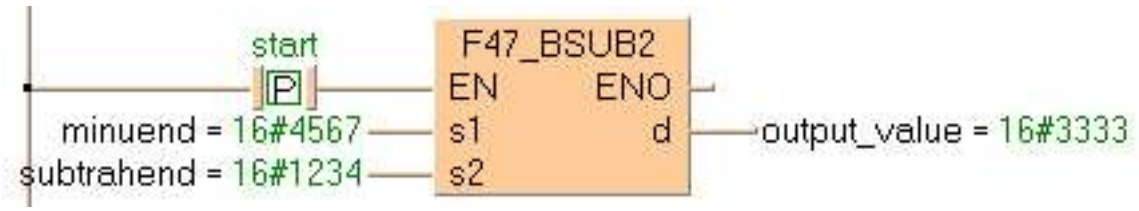
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	minuend	WORD	16#4567	minuent
2	VAR	subtrahend	WORD	16#1234	subtrahent
3	VAR	output_value	WORD	0	result after a 0->1 leading edge from start: 16#3333
4	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



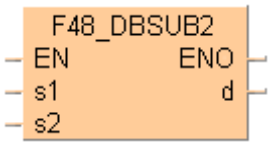
ST

```
IF start THEN
    F47_BSUB2(minuend, subtrahend, output_value);
END_IF;
```

F48_DBSUB2

8-Digit-BCD-Subtraktion mit Transfer

Erklärung Der BCD-codierte 32-Bit-Wert des Speicherregisters **s2** oder eine Konstante **s2** wird vom BCD-codierten 32-Bit-Wert des Speicherregisters **s1** oder von einer Konstanten **s1** subtrahiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 16#33555588 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s1	0 0 0 1	0 0 1 0	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1	1 0 0 0	1 0 0 0
16# BCD	3	3	5	5	5	5	8	8

32-bit area

Example value 16#00110022 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s2	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1	0 0 0 0	0 0 0 0	0 0 1 0	0 0 1 0
16# BCD	0	0	1	1	0	0	2	2

Trigger: ON

Result value 16#33445566 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 1 1	0 0 1 1	0 1 0 0	0 1 0 0	0 1 0 1	0 1 0 1	0 1 1 0	0 1 1 0
16# BCD	3	3	4	4	5	5	6	6

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F48_DBSUB2 (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	DWORD	Minuend, 32-Bit-Bereich für achtstellige BCD-codierte Daten oder äquivalente Konstante
s2	DWORD	Subtrahen, 32-Bit-Bereich für achtstellige BCD-Daten oder äquivalente Konstante
d	DWORD	Ergebnis, 32-Bit-Bereich für achtstellige BCD-codierte Daten

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Überlauf).

Beispiel

In diesem Beispiel wird die Funktion F48_DBSUB2 im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

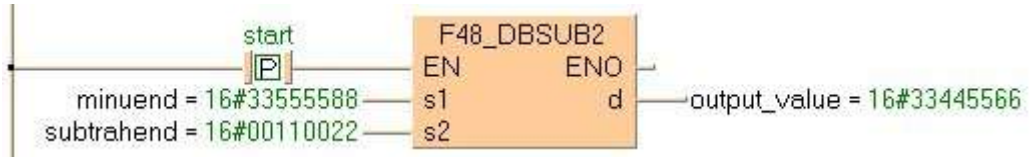
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	minuend	DWORD	16#33555588	minuent
2	VAR	subtrahend	DWORD	16#00110022	subtrahent
3	VAR	output_value	DWORD	0	result after a 0->1 leading edge from start: 16#33445566
4	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



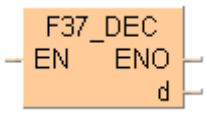
ST

```
IF start THEN
    F48_DBSUB2(minuend, subtrahend, output_value);
END_IF;
```

F37_DEC

16-Bit-Dekrementieren

Erklärung Vom 16-Bit-Wert des Speicherregisters **d** wird der Wert "1" subtrahiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 17

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1

Result value 16 if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 0

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F37_DEC (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
d	INT, WORD	16-Bit-Datenbereich, der um 1 erhöht wird

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 16-Bit-Daten liegt (Unterlauf).

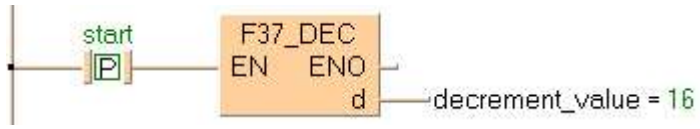
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	decrement_value	INT	17	result after a 0->1 leading edge from start: 16
2	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

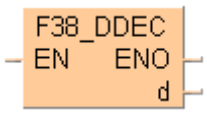


ST

```
IF DF(start) THEN
    F37_DEC(decrement_value);
END_IF;
```

F38_DDEC 32-Bit-Dekrementieren

Erklärung Ist **EN** gesetzt, wird vom 32-Bit-Wert des Speicherregisters **d** der Wert "1" subtrahiert. Das Ergebnis wird in **d** gespeichert.



Example value 131081

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0000	0000	0000	0010	0000	0000	0000	1001

32-bit area

Result 131080

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0000	0000	0000	0010	0000	0000	0000	1000

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F38_DDEC (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
d	ANY32	32-Bit-Datenbereich, der um 1 erhöht wird

Operanden

Für	Merker				T/C		Register			Konstante
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von 32-Bit-Daten liegt (Unterlauf).

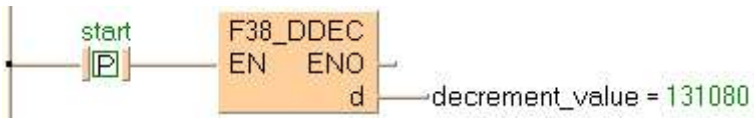
Beispiel In diesem Beispiel wird die Funktion F38_DDEC im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	decrement_value	DINT	17

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



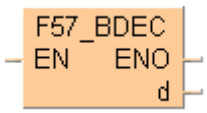
ST

```
IF DF(start) THEN
    F38_DDEC(decrement_value);
END_IF;
```

F57_BDEC

4-Digit-BCD-Dekrementieren

Erklärung Vom BCD-codierten 16-Bit-Wert des Speicherregisters **d** wird der Wert "1" subtrahiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 4322 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 1 0 0	0 0 1 1	0 0 1 0	0 0 1 0
16# BCD	4	3	2	2

Trigger: ON

Result value 4321 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 1 0 0	0 0 1 1	0 0 1 0	0 0 0 1
16# BCD	4	3	2	1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F57_BDEC (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
d	WORD	16-Bit-Datenbereich, der um 1 erhöht wird

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Unterlauf).

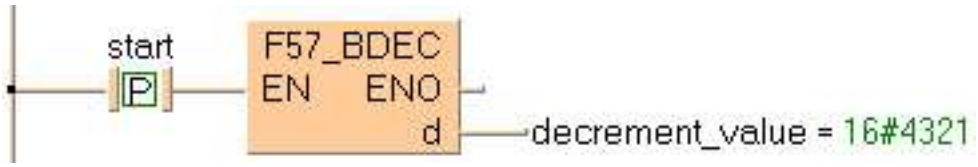
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	decrement_value	WORD	16#4322	result after a 0->1 leading edge from start: 16#4321
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

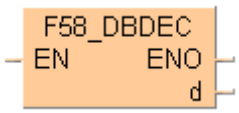
KOP



ST
```
IF DF(start) THEN
    F57_BDEC(decrement_value);
END_IF;
```

F58_DBDEC 8-Digit-BCD-Dekrementieren

Erklärung Vom BCD-codierten 32-Bit-Wert des Speicherregisters **d** wird der Wert "1" subtrahiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d** gespeichert.



Example value 87654322 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	1000	0111	0110	0101	0100	0011	0010	0010
16# BCD	8	7	6	5	4	3	2	2

32-bit area

Trigger: ON

Result value 87654321 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	1000	0111	0110	0101	0100	0011	0010	0001
16# BCD	8	7	6	5	4	3	2	1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F58_DBDEC (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
d	DWORD	32-Bit-Datenbereich, der um 1 erhöht wird

Operanden

Für	Merker				T/C		Register			Konstante
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis außerhalb des Bereichs von BCD-codierten Daten liegt (Unterlauf).

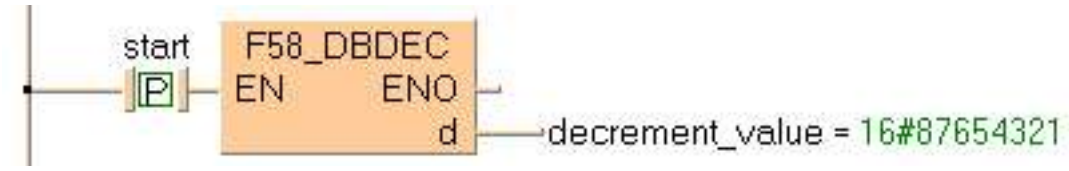
Beispiel In diesem Beispiel wird die Funktion F58_DBDEC im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	decrement_value	DWORD	16#87654322	result after a 0->1 leading edge from start: 16#87654321
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



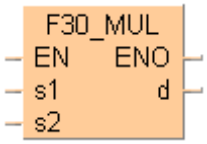
ST
```
IF DF(start) THEN
    F58_DBDEC(decrement_value);
END_IF;
```

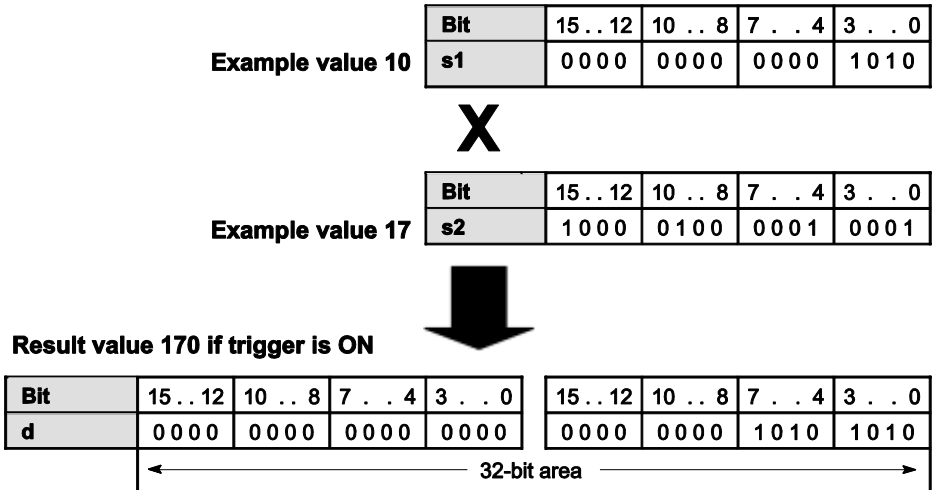
F30_MUL

16-Bit-Multiplikation mit Transfer

Erklärung

Der 16-Bit-Wert des Speicherregisters s1 oder eine Konstante **s1** und der 16-Bit-Wert des Speicherregisters **s2** oder eine Konstante s2 werden multipliziert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis der Multiplikation wird in das Speicherregister **d** (32-Bit-Bereich) geschrieben. Alle 16-Bit-Werte werden als Integer-Werte behandelt.





Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden MUL (s. S. 63). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen

Verfügbarkeit von F30_MUL (s. S. 1189)

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Multiplikand
s2		Multiplikator
d	ANY32	Ergebnis

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein (INT/DINT oder WORD/DWORD).

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

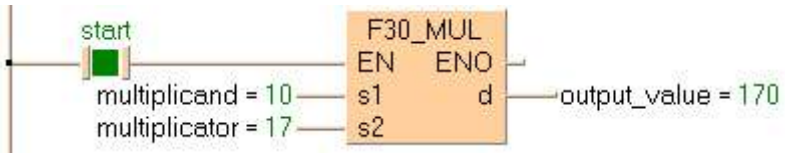
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the fuction
1	VAR	multiplicand	INT	10	multiplicant
2	VAR	multiplicator	INT	17	multiplicator
3	VAR	output_value	DINT	0	result after a 0->1 leading edge from start: 170
4	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

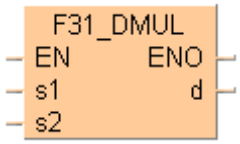


ST
```
IF start THEN
    F30_MUL(multiplicand, multiplicator, output_value);
END_IF;
```

F31_DMUL

32-Bit-Multiplikation mit Transfer

Erklärung Der 32-Bit-Wert des Speicherregisters **s1** oder eine Konstante s1 und der 32-Bit-Wert des Speicherregisters **s2** oder eine Konstante s2 werden multipliziert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird in **d[0]**, **d[1]** (64-Bit-Bereich) gespeichert. Alle 32-Bit-Werte werden als Double Integer gewertet.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden MUL (s. S. 63).

Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen **Verfügbarkeit von F31_DMUL (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Multiplikand
s2		Multiplikator
d	ARRAY [0..1] OF ANY32	Ergebnis

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

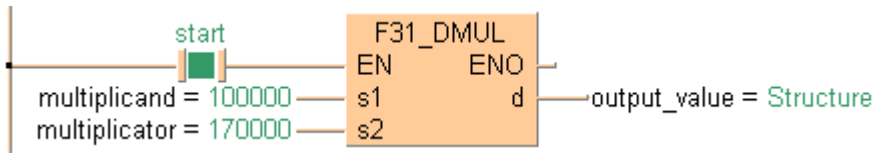
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the fuction
1	VAR	multiplicand	DINT	100000	multiplicant
2	VAR	multiplicator	DINT	170000	multiplicator
3	VAR	output_value	ARRAY [0..1] OF DINT	[2(0)]	result after a 0->1 leading edge from start: [170,0]

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



Ausgangswert[0] bzw. Ausgangswert[1] kann dann auf das Ergebnis zugegriffen werden.

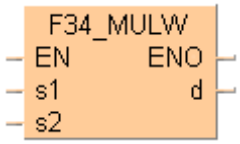
ST

```
IF start THEN
    F31_DMUL(multiplicand, multiplicator, output_value);
END_IF;
```

F34_MULW

16-Bit-Multiplikation

Erklärung Die Funktion multipliziert den Wert am Eingang **s1** mit dem Wert am Eingang **s2**. Das Funktionsergebnis wird am Ausgang **d** zurückgegeben. Achten Sie darauf, dass das Ergebnis am Ausgang **d** zwischen -32768 und 32767 (bzw. 16#0 und 16#FFFF) liegt. Alle 16-Bit-Werte werden als Integer-Werte behandelt.



Example value 6

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s1	0 0 0 0	0 0 0 0	0 0 0 0	0 1 1 0

X

Example value 5

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s2	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 1

Result value 30 if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	1 1 1 0

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F34_MULW (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Multiplikand
s2		Multiplikator
d		Ergebnis

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ das berechnete Ergebnis den 16-Bit-Speicherbereich am Ausgang **d** überschreitet.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the fuction
1	VAR	input_value_1	INT	6	
2	VAR	output_value	INT	0	result: here 30

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

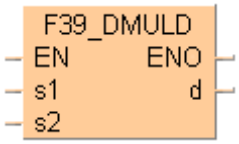
ST

```
IF start THEN
    F34_MULW(input_value_1, 5, output_value);
END_IF;
```

F39_DMULD

32-Bit-Multiplikation

Erklärung Die Funktion multipliziert den Wert am Eingang **s1** mit dem Wert am Eingang **s2**. Das Funktionsergebnis wird am Ausgang **d** zurückgegeben. Achten Sie darauf, dass das Ergebnis am Ausgang **d** zwischen -2147483648 und 2147483647 (bzw. 16#0 und 16#FFFFFFFF) liegt. Alle 32-Bit-Werte werden als Double Integer gewertet.



Example value 17

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1

Result value 18 if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 1	0 0 1 0

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F39_DMULD (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Multiplikand
s2		Multiplikator
d		Ergebnis

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ das berechnete Ergebnis den 32-Bit-Speicherbereich am Ausgang d überschreitet.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	input_value_1	DINT	1312896
2	VAR	input_value_2	DINT	10
3	VAR	output_value	DINT	0

Hier wurden die Eingangsvariablen **input_value_1** und **input_value _2** deklariert. Statt dessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

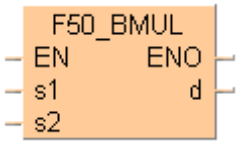
ST
```
IF start THEN
    F39_DMULD(input_value_1, input_value_2, output_value);
END_IF;
```

F50_BMUL

4-Digit-BCD-Multiplikation mit Transfer

Erklärung

Der BCD-codierte 16-Bit-Wert des Speicherregisters **s1** oder eine Konstante **s1** und der BCD-codierte 16-Bit-Wert des Speicherregisters **s2** oder eine Konstante **s2** werden multipliziert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis der Multiplikation wird in das Speicherregister **d** (8-Bit-Bereich) geschrieben.



Example value 16#20 BCD

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s1	0	0	2	0

×

Example value 16#2 BCD

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s2	0	0	0	2

Result value 16#40 if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0	0	0	0	0	0	4	0

32-bit area

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F50_BMUL (s. S. 1191)

Datentypen

Variable	Datentyp	Funktion
s1	WORD	Multiplikand, 16-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
s2	WORD	Multiplikator, 16-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
d	DWORD	Ergebnis, 32-Bit-Bereich für achtstellige BCD-codierte Daten

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	multiplicand	WORD	16#20	multiplicand
2	VAR	multiplicator	WORD	16#2	multiplicator
3	VAR	output_value	DWORD	0	result after a 0->1 leading edge from start: 16#40
4	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

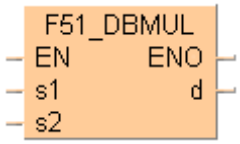
ST

```
IF start THEN
    F50_BMUL(multiplicand, multiplicator, output_value);
END_IF;
```

F51_DBMUL

8-Digit-BCD-Multiplikation mit Transfer

Erklärung Der BCD-codierte 32-Bit-Wert des Speicherregisters **s1** oder eine Konstante **s1** und der BCD-codierte 32-Bit-Wert des Speicherregisters **s2** oder eine Konstante **s2** werden multipliziert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis wird im ARRAY **d[0]**, **d[1]** (64-Bit-Bereich) gespeichert.



Example value 16#60008 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 0	0 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0	1 0 0 0
16# BCD	0	0	0	6	0	0	0	8

32-bit area

Example value 16#40002 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 0
16# BCD	0	0	0	4	0	0	0	2

Result value 16#2400440016 (BCD) if trigger is ON stored in the ARRAY [0..1] of DWORD

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d array[0]	0 0 0 0	0 0 0 0	0 1 0 0	0 1 0 0	0 0 0 0	0 0 0 0	0 0 0 1	0 1 1 0
16# BCD	0	0	4	4	0	0	1	6

output_array[0]

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d array[1]	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 0	0 1 0 0
16# BCD	0	0	0	0	0	0	2	4

output_array[1]

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F51_DBMUL (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	DWORD	Multiplikand, 32-Bit-Bereich für achtstellige BCD-codierte Daten oder äquivalente Konstante
s1	DWORD	Multiplikator, 32-Bit-Bereich für achtstellige BCD-codierte Daten oder äquivalente Konstante
d	ARRAY **[0..1] of** DWORD	Ergebnis

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

Beispiel

In diesem Beispiel wird die Funktion F51_DBMUL im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

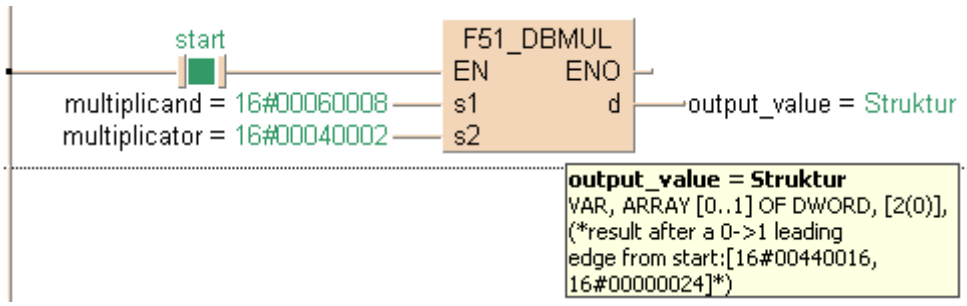
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	multiplicand	DWORD	16#60008	multiplicand
2	VAR	multiplicator	DWORD	16#40002	multiplicator
3	VAR	output_value	ARRAY [0..1] OF DWORD	[2(0)]	result after a 0->1 leading edge from start:[16#00440016, 16#00000024]
4	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST

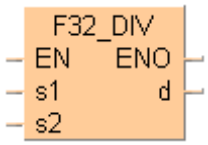
```
IF start THEN
    F51_DBMUL(multiplicand, multiplicator, output_value);
END_IF;
```

F32_DIV

16-Bit-Division mit Transfer

Erklärung

Der 16-Bit-Wert des Speicherregisters **s1** oder eine Konstante s2 wird durch den 16-Bit-Wert des Speicherregisters s1 oder einer Konstante **s2** dividiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Das Ergebnis der Division steht im Speicherregister **d**. Der Divisionsrest steht im Sonder-Datenregister DT9015 (DT90015 für FP2/2SH und FP10/10S/10SH). Alle 16-Bit-Werte werden als Integer-Werte behandelt.

Example value 36

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s1	0 0 0 0	0 0 0 0	0 0 1 0	0 1 0 0

Example value 17

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s2	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1

Result value 2 if trigger is ON

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 0

Remainder 2 stored in DT9015/90015

15 . . 12	10 . . 8	7 . . 4	3 . . 0
0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 0

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden DIV (s. S. 64). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen

Verfügbarkeit von F32_DIV (s. S. 1190)

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Dividend
s2		Divisor
d		Quotient

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.
R9009	%MX0.900.9	▪ kurzzeitig	▪ der negative Mindestwert -32768 (16#8000) wird durch -1 (16#FFFF) geteilt

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

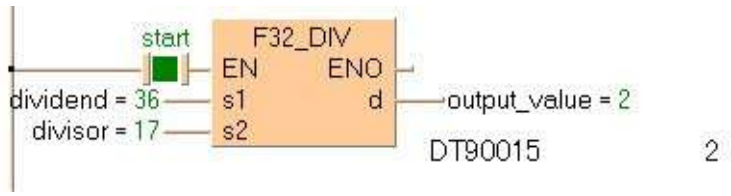
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the fuction
1	VAR	dividend	INT	36	dividend
2	VAR	divisor	INT	17	divisor
3	VAR	output_value	INT	0	result after a 0->1 leading edge from start: 2
4	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST

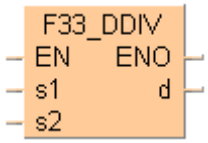
```
IF start THEN
    F32_DIV(dividend, divisor, output_value);
END_IF;
```

F33_DDIV

32-Bit-Division mit Transfer

Erklärung

Der 32-Bit-Wert des Speicherregisters **s1** oder eine Konstante s2 wird durch den 32-Bit-Wert des Speicherregisters s1 oder einer Konstante **s2** dividiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis der Division steht im Speicherregister **d**. Der Divisionsrest steht im Sonder-Datenregister DDT9015 (DDT90015 für FP2/2SH und FP10/10S/10SH). Alle 32-Bit-Werte werden als Double Integer gewertet.



Example value 16908416

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s1	0 0 0 0	0 0 0 1	0 0 0 0	0 0 1 0	0 0 0 0	0 0 0 0	1 0 0 0	0 0 0 0

32-bit area

÷

Example value 589828

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s2	0 0 0 0	0 0 0 0	0 0 0 0	1 0 0 1	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0

Result value 28 if trigger is ON

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1	1 1 0 0

Remainder 393232

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
	0 0 0 0	0 0 0 0	0 0 0 0	0 1 1 0	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 0

DT9016/DDT90016 | DT9015/DDT90015

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden DIV (s. S. 64). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F33_DDIV (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Dividend
s2		Divisor
d		Quotient

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

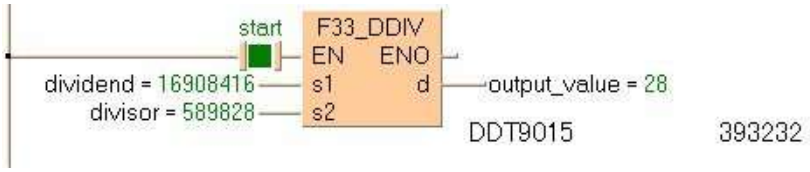
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	dividend	DINT	16908416	dividend
2	VAR	divisor	DINT	589828	divisor
3	VAR	output_value	DINT	0	result after a 0->1 leading edge from start: 28
4	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

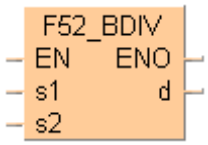


ST

```
IF start THEN
    F33_DDIV(dividend, divisor, output_value);
END_IF;
```

F52_BDIV — 4-Digit-BCD-Division mit Transfer

Erklärung Der BCD-codierte 16-Bit-Wert des Speicherregisters **s1** oder eine Konstante **s1** wird durch das den BCD-codierten 16-Bit-Wert des Speicherregisters **s2** oder eine Konstante **s2** dividiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Das Ergebnis der Division steht im Speicherregister **d**. Der Divisionsrest steht im Sonder-Datenregister DT9015 (DT90015 für FP2/2SH und FP10/10S/10SH).

Example value 16#0037 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 1 1	0 1 1 1
16# (BCD)	0	0	3	7

Example value 16#0015 (BCD)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	0 0 0 1	0 1 0 1
16# (BCD)	0	0	1	5

Trigger: ON

Result value 16#0002

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 0
16# (BCD)	0	0	0	2

Remainder 16#0007

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
DT9015	0 0 0 0	0 0 0 0	0 0 0 0	0 1 1 1
16# (BCD)	0	0	0	7

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F52_BDIV (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	WORD	Dividend, 16-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
s2	WORD	Divisor, 16-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
d	WORD	Quotient, 16-Bit-Datenbereich für BCD-Daten (Der Restbetrag wird in Sonderdatenregistern DT9015/DT90015 gespeichert)

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

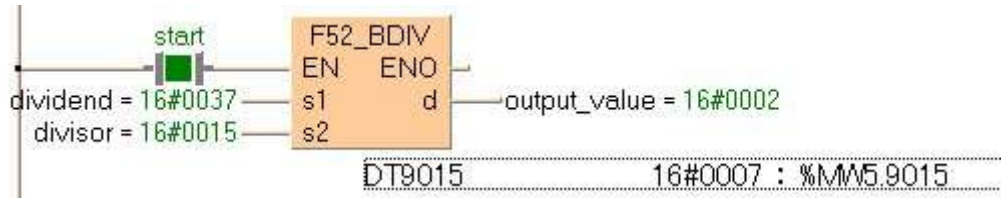
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the fuction
1	VAR	dividend	WORD	16#0037	dividend
2	VAR	divisor	WORD	16#0015	divisor
3	VAR	output_value	WORD	0	result after 0->1 leading edge from start: 16#0002
4	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

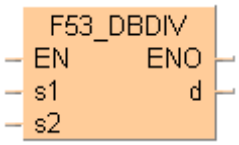


ST
```
IF start THEN
    F52_BDIV(dividend, divisor, output_value);
END_IF;
```

F53_DBDIV

8-Digit-BCD-Division mit Transfer

Erklärung Das Ergebnis der Division steht im Speicherregister **d**. Der Divisionsrest steht in den Sonder-Datenregistern DT9016 und DT9015 (DT90016 und DT90015 für FP2/2SH und FP10/10S/10SH).



Example value 16#00001110 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1	0 0 0 0
16# BCD	0	0	0	0	0	1	1	0

32-bit area

Example value 16#0000011 (BCD)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
s2	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1
16# BCD	0	0	0	0	0	0	1	1

Result value 16#00000100 (BCD) if trigger is ON

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 0	0 0 0 0
16# BCD	0	0	0	0	0	1	0	0

Remainder 16#00000010 (BCD) if trigger is ON stored in DT9015 to DT9016 (DDT90015 to DDT90016)

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 0
16# BCD	0	0	0	0	0	0	1	0
	DT9016/DDT90016				DT9015/DDT90015			

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F53_DBDIV (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	DWORD	Dividend, 32-Bit-Bereich für achtstellige BCD-codierte Daten oder äquivalente Konstante
s2	DWORD	Divisor, 32-Bit-Bereich für BCD-codierte Daten oder äquivalente Konstante
d	DWORD	Quotient, 32-Bit-Datenbereich für BCD-Daten (Der Restbetrag wird in den Sonderdatenregistern DT9016 und DT9015/DT90016 und DT90015 gespeichert)

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

Beispiel

In diesem Beispiel wird die Funktion F53_DBDIV im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

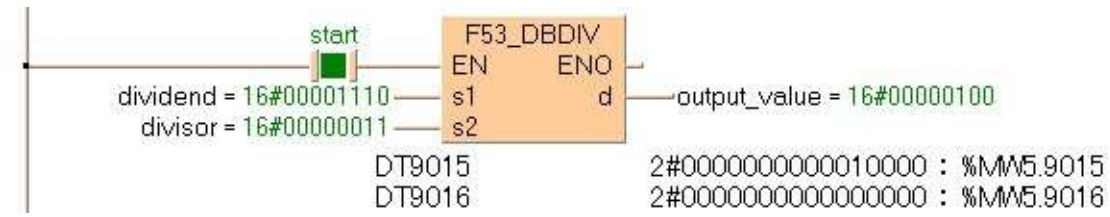
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	dividend	DWORD	16#00001110	dividend
2	VAR	divisor	DWORD	16#00000011	divisor
3	VAR	output_value	DWORD	0	result after 0->1 leading edge from start: 16#00000100
4	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

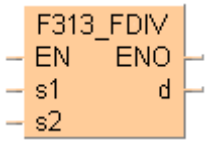


ST

```
IF start THEN
    F53_DBDIV(dividend, divisor, output_value);
END_IF;
```

F313_FDIV — Division von Fließkommawerten

Erklärung Wird EN gesetzt, dann wird der Fließkommawert, der mit **s1** festgelegt wird, durch den Fließkommawert, der mit **s2** festgelegt wird, dividiert. Das Ergebnis wird in **d** gespeichert.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden DIV (s. S. 64). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen **Verfügbarkeit von F313_FDIV (s. S. 1190)**

☞ **Dieser Befehl kann nicht im Interrupt-Programm programmiert werden.**

Datentypen

Variable	Datentyp	Funktion
s1	REAL	Fließkommawert, Dividend.
s2	REAL	Fließkommawert, Divisor.
d	REAL	32-Bit-Zielbereich

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Werte, die mit s1 und s2 festgelegt werden, keine Fließkommawerte sind. ▪ der Fließkommawert für den Divisor, der mit s2 festgelegt wird, ist "0,0".
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis einen Speicherüberlauf verursacht.

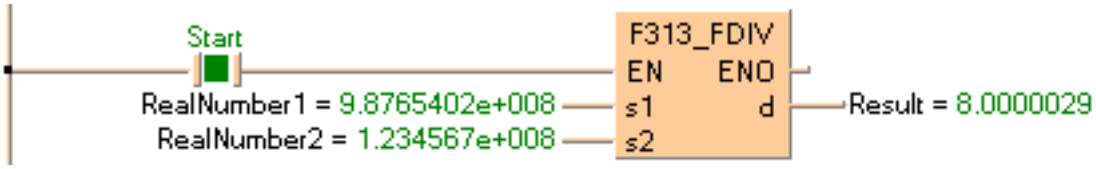
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Start	BOOL	FALSE
1	VAR	Result	REAL	0.0
2	VAR	Real Number1	REAL	987654321.0
3	VAR	Real Number2	REAL	123456789.0

Rumpf Wenn die Variable **Start** auf TRUE gesetzt wird, wird der Fließkommawert **RealNumber1** durch den Fließkommawert **RealNumber2** dividiert und das Ergebnis in den Adressbereich geschrieben, den der Compiler der Variablen **Result** zugewiesen hat. Das Monitorwertsymbol ist aktiviert.

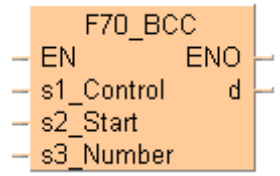
KOP

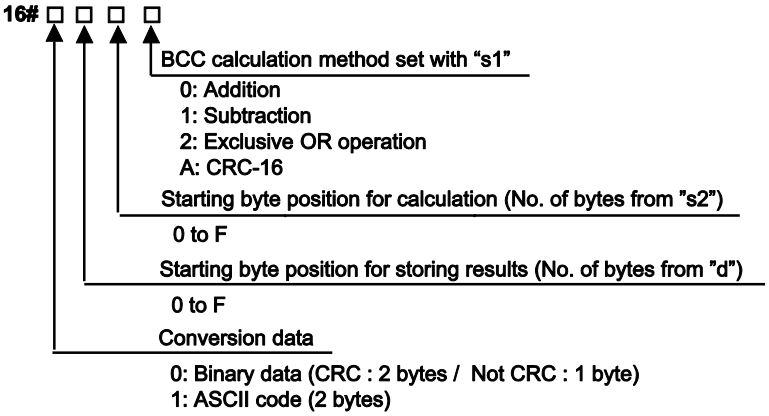

Start
F313_FDIV
EN
ENO
RealNumber1 = 9.8765402e+008
s1
d
Result = 8.0000029
RealNumber2 = 1.234567e+008
s2

F70_BCC

Prüfcode (Block Check Code) -Berechnung

Erklärung Berechnet entsprechend der am Eingang **s1** festgelegten Methode eine Prüfsumme (BBC-Block Check Code) über einen Speicherbereich, der über eine Anfangsadresse **s2** und eine Anzahl von Bytes **s3** definiert ist. Der Prüfcode (BCC) wird in das niederwertige Byte des Speicherregisters **d** geschrieben. (BCC ist ein Byte. Das höherwertige Byte von **d** ändert sich nicht.)





☞ **Wenn als Berechnungsmethode CRC-16 angegeben ist, kann als Umwandlungswert kein ASCII-Code ausgewählt werden.**

SPS-Typen **Verfügbarkeit von F70_BCC (s. S. 1192)**

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Datentypen

Variable	Datentyp	Funktion
s1	INT	Legt die Berechnungsmethode für den Prüfcode BCC fest: 0 = Addition, 1 = Subtraktion, 2 = XOR-Verknüpfung
s2	ANY16	Anfangsadresse für die Prüfsummenberechnung (BCC).
s3	INT	Legt die Anzahl der Bytes für die Prüfsummenberechnung fest.
d	ANY16	Speicherregister für die Prüfsumme (BCC).

Operanden

Für	Bitmerker				T/C		Register			Konstante
s1, s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	die Anzahl der ausgewählten Bytes für die Sollwerte die Grenze des ausgewählten Datenbereichs überschreitet.
R9008	%MX0.900.8	kurzzeitig	

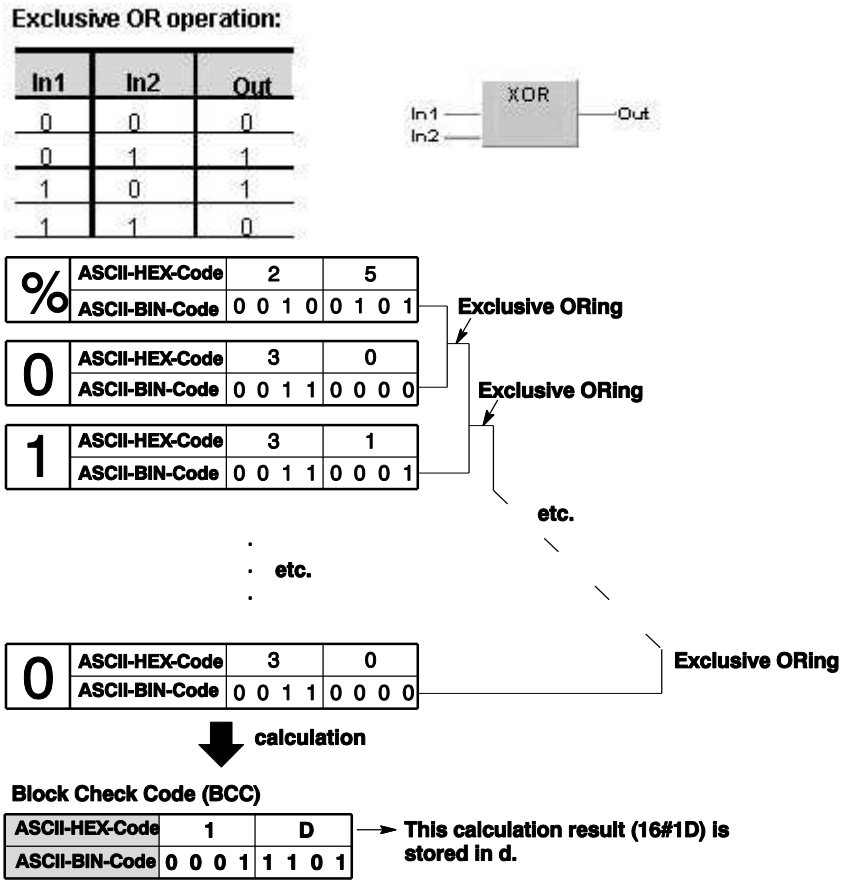
Beispiel In diesem Beispiel wird die Funktion F70_BCC im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Start	BOOL	FALSE	
1	VAR	BCC_Calc_Method	INT	2	0 = Addition
2	VAR	ASCII_String	STRING[32]	'%01#RCSX0000'	
3	VAR	BCC	WORD	0	result = 16#1D

Rumpf Wenn **Start** gesetzt (TRUE) ist , wird die Prüfsumme (BCC) aufgrund der Wertes, der für die Variable **ASCII_Folge** eingegeben wurde, berechnet. Für die BCC-Methode wurde die XOR-Verknüpfung ausgewählt.

Wie die Prüfsumme (BCC) mit Hilfe der XOR-Verknüpfung berechnet wird:



Der ASCII-BIN-Code der ersten beiden Zeichen wird miteinander verglichen, um ein XOR-Verknüpfungsergebnis mit 8 Zeichen Länge zu erhalten.

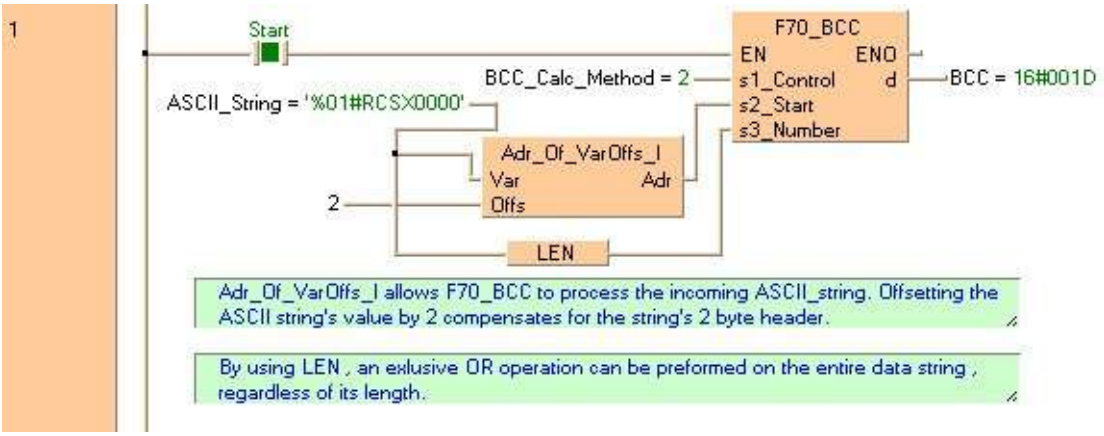
Signal für den Vergleich	ASCII-BIN-Code
%	00100101
0	00110000
Ergebnis der XOR-Verknüpfung	00010101

Dieses Ergebnis wird mit dem ASCII-BIN-Code des nächsten Zeichens verglichen, hier "1".

Signal für den Vergleich	ASCII-BIN-Code
Ergebnis der XOR-Verknüpfung	00010101
1	00110001
Nächste XOR-Verknüpfung	00100100

So wird bis zum letzten Zeichen verfahren.

KOP

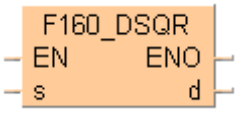


ST

```
IF start THEN
    F70_BCC( s1_Control:= BCC_Calc_Methode,
        s2_Start:= Adr_Of_VarOffs( Var:= ASCII_String,
        Offs:= 2),
        s3_Number:= LEN( ASCII_String),
        d=> BCC);
END_IF;
```

F160_DSQR Quadratwurzel einer 32-Bit-Zahl

Erklärung Aus dem binären 32-Bit-Wert des Speicherregisters **s** oder aus einer Konstanten s wird die Quadratwurzel gezogen. Das Ergebnis (Quadratwurzel) wird in **d** gespeichert.



Es erfolgt Doppelwortverarbeitung. Die Nachkommastellen werden gelöscht.

Example value 64

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
Binary	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0	0 0 0 0
Decimal	64							

32-bit area

Trigger: ON

Result value 8

Bit	31 . . 28	27 . . 24	23 . . 20	19 . . 16	15 . . 12	10 . . 8	7 . . 4	3 . . 0
Binary	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0
Decimal	8							

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden SQRT (s. S. 67). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen **Verfügbarkeit von F160_DSQR (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s	DINT, DWORD	Quelle, 32-Bit-Bereich zur Verarbeitung
d	DINT, DWORD	Quadratwurzel (Nachkommastellen gelöscht)

Die Variablen **s1** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Beispiel In diesem Beispiel wird die Funktion F160_DSQR im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	DINT	70	input_value:=70
2	VAR	output_value	DINT	0	result after a 0->1 leading edge from start: 8
3	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

ST

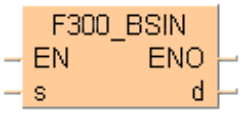
```
IF start THEN
    F160_DSQR(input_value, output_value);
END_IF;
```

F300_BSIN

Sinus-Funktion mit BCD-codierten Werten

Erklärung

Die Funktion berechnet den Sinus eines BCD-Winkelwertes (Eingang **s**) und speichert das Ergebnis (Ausgang **d**) als BCD-Wert in einem ARRAY mit drei Elementen.



BCD-Werte für den Eingang **s** liegen im Bereich von 0° bis 360° (16#0 bis 16#360) in 1° Schritten. Damit kann das am Ausgang **d** anliegende Ergebnis Werte im Bereich von -1.0000 bis 1.0000 annehmen. Das Funktionsergebnis wird wie folgt ausgegeben:

ARRAY[0]	Vorzeichen des Eingangswertes (0 bei positivem Vorzeichen, 1 bei negativem Vorzeichen).
ARRAY[1]	Vorkommawert als ganze Zahl (0 oder 1).
ARRAY[2]	Nachkommastellen als BCD-kodierter Wert auf 4 Stellen gerundet (16#0 bis 16#9999).

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F300_BSIN (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	WORD	16-Bit-Bereich zur Speicherung des Winkelwertes
d	ARRAY **[0..2] of** WORD	Ergebnis wird in 3 Wörtern gespeichert

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Eingangswert an s kein BCD-codierter Wert ist oder nicht zwischen 0° und 360° liegt.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	kurzzeitig	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe. Zusätzlich wird eine Auswertung programmiert, die das Ergebnis interpretiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

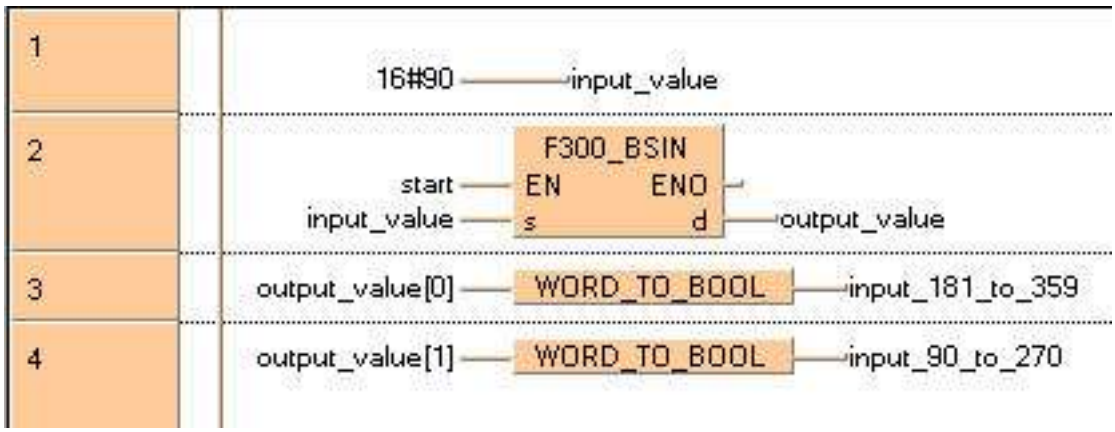
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	WORD	0	BCD value between
2	VAR	output_value	ARRAY [0..2] OF WORD	[3(0)]	number between -1.0000 and 1.0000
3	VAR	input_181_to_359	BOOL	FALSE	TRUE if input_value
4	VAR	input_90_or_270	BOOL	FALSE	TRUE if input_value 90° or 270°
5	VAR				

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Statt dessen können Sie im Rumpf auch eine Konstante (z.B. 16#45 für 45°) direkt an den Eingang der Funktion schreiben.

Rumpf Im Rumpf wird der Variablen **input_value** der Wert 90° zugewiesen. Wenn die Variable **start** auf den Wert TRUE gesetzt ist, wird die Funktion F300_BSIN ausgeführt. Sie speichert das Ergebnis in der Variablen **output_value**. Liegt **input_value** zwischen 181° und 359°, hat **output_value** ein negatives Vorzeichen. Die Funktion WORD_TO_BOOL setzt in diesem Fall die Variable **input_181_to_359** auf den Wert TRUE. Bei einem **input_value** von 90° oder 270° hat **output_value** 1 als Vorkommawert. Ist dies der Fall, dann setzt eine weitere Funktion WORD_TO_BOOL die Variable **input_90_or_270** auf den Wert TRUE.

KOP



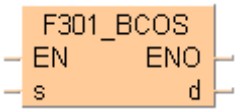
ST
```
input_value:=16#90;
IF start THEN
    F300_BSIN( input_value, output_value );
END_IF;
input_181_to_359:=WORD_TO_BOOL(output_value[0]);
input_90_or_270:=WORD_TO_BOOL(output_value[1]);
```

F301_BCOS

Cosinus-Funktion mit BCD-codierten Werten

Erklärung

Die Funktion berechnet den Sinus eines BCD-Winkelwertes (Eingang **s**) und speichert das Ergebnis (Ausgang **d**) als BCD-Wert in einem ARRAY mit drei Elementen.



BCD-Werte für den Eingang **s** liegen im Bereich von 0° bis 360° (16#0 bis 16#360) in 1° Schritten. Damit kann das am Ausgang **d** anliegende Ergebnis Werte im Bereich von -1.0000 bis 1.0000 annehmen. Das Funktionsergebnis wird wie folgt ausgegeben:

ARRAY[0]	Vorzeichen des Eingangswertes (0 bei positivem Vorzeichen, 1 bei negativem Vorzeichen).
ARRAY[1]	Vorkommawert als ganze Zahl (0 oder 1).
ARRAY[2]	Nachkommastellen als BCD-codierter Wert auf 4 Stellen gerundet (16#0 bis 16#9999).

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F301_BCOS (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	WORD	Bereich zur Speicherung des Winkelwertes
d	ARRAY **[0..2] of** WORD	Ergebnis wird in 3 Wörtern gespeichert

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Eingangswert an s kein BCD-codierter Wert ist oder nicht zwischen 0° und 360° liegt.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	kurzzeitig	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

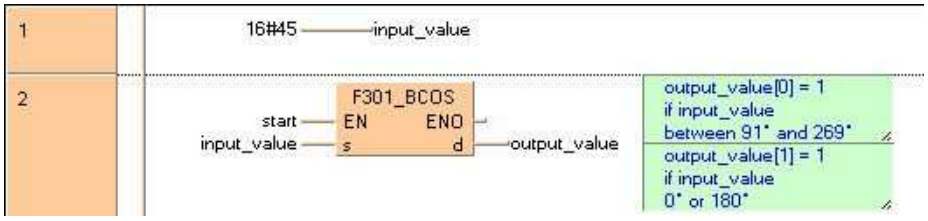
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	WORD	0	BCD value between number between -1.0000 and 1.0000
2	VAR	output_value	ARRAY [0..2] OF WORD	[3(0)]	output_value [0] = +/- sign output_value [1] = pre-decimal value output_value [2] = post-decimal point values
3	VAR				result: here +0.7071

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Statt dessen können Sie im Rumpf auch eine Konstante (z.B. 16#45 für 45°) direkt an den Eingang der Funktion schreiben.

Rumpf Um Rumpf wird der Variablen **input_value** der Wert 16#45° zugewiesen. Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Das Ergebnis lautet am Ausgang d: **output_value[0]** = 0, **output_value[1]** = 0, **output_value[2]** = 7071.

KOP



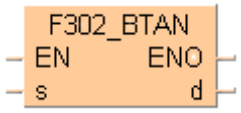
ST
```
input_value:=16#45;
IF start THEN
    F301_BCOS( input_value, output_value );
END_IF;
```

F302_BTAN

Tangens-Funktion mit BCD-codierten Werten

Erklärung

Die Funktion berechnet den Tangens eines BCD-Winkelwertes (Eingang **s**) und speichert das Ergebnis (Ausgang **d**) als BCD-Wert in einem ARRAY mit drei Elementen.



BCD-Werte für den Eingang **s** liegen im Bereich von 0° bis 360° (16#0 bis 16#360) in 1° Schritten. Damit kann das am Ausgang **d** anliegende Ergebnis Werte im Bereich von -57,2900 bis 57,2900 annehmen. Das Funktionsergebnis wird wie folgt ausgegeben:

ARRAY[0]	Vorzeichen des Eingangswertes (0 bei positivem Vorzeichen, 1 bei negativem Vorzeichen).
ARRAY[1]	Vorkommawert als BCD-Wert (6#0 bis 16#57).
ARRAY[2]	Nachkommastellen als BCD-codierter Wert auf 4 Stellen gerundet (16#0000 bis 16#9999).

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F302_BTAN (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	WORD	Bereich zur Speicherung des Winkelwertes
d	ARRAY **[0..2] of** WORD	Ergebnis wird in 3 Wörtern gespeichert

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Eingangswert an **s** kein BCD-codierter Wert ist ▪ **s** = 90° (16#90) oder 270° (16#270)
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

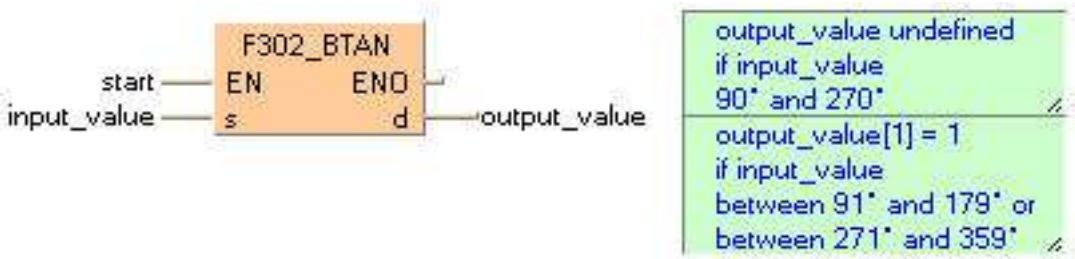
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	WORD	16#89	BCD value between
2	VAR	output_value	ARRAY [0..2] OF WORD	[3(0)]	number between -57.2900 and 57.2900 output_value[0] = +/- sign output_value[1] = pre-decimal point values output_value[2] = post-decimal point values result: here +57.2899
3	VAR				

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Statt dessen können Sie im Rumpf auch eine Konstante (z.B. 16#89 für 89°) direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Der **input_value** wurde im POE-Kopf mit dem Wert 16#89 (89°) initialisiert. Das Funktionsergebnis wird in den ARRAY **output_value** geschrieben. Hier steht im ersten Element von **output_value** = 16# (positives Vorzeichen). Im zweiten Element steht 16#57 als Vorkommastellen und im dritten Element 16#2899 als Nachkommastellen.

KOP



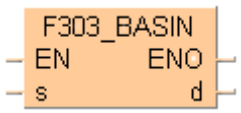
ST
```
IF start THEN
    F302_BTAN(input_value, output_value);
END_IF;
```

F303_BASIN

Arcus-Sinus-Funktion mit BCD-codierten Werten

Erklärung

Die Funktion berechnet den Arkurssinus eines BCD-Wertes, der in einem ARRAY mit drei Elementen dem Eingang **s** übergeben wird. Das Ergebnis wird als BCD-Wert im Bereich von 0° bis 360° (16#0 bis 16#360) am Ausgang **d** zurückgegeben.



BCD-Werte für den Eingang **s** liegen im Bereich von -1.0000 bis 1.0000. Sie werden wie folgt eingegeben:

ARRAY[0]	Vorzeichen des Eingangswertes (0 bei positivem Vorzeichen, 1 bei negativem Vorzeichen).
ARRAY[1]	Vorkommawert als ganze Zahl (0 oder 1).
ARRAY[2]	Nachkommastellen als BCD-codierter Wert auf 4 Stellen gerundet (16#0 bis 16#9999).

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F303_BASIN (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	ARRAY **[0..2] of** WORD	Bereich zur Speicherung des Winkelwertes
d	WORD	Ergebnis wird in 3 Wörtern gespeichert

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Eingangswert an s kein BCD-codierter Wert ist oder nicht zwischen -1.0000 und 1.0000 liegt.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

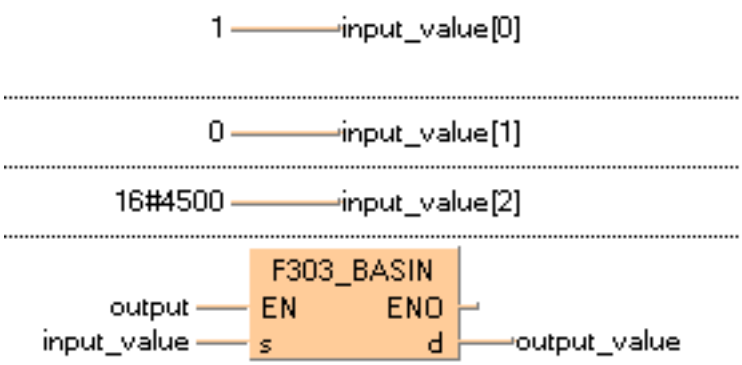
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	ARRAY [0..2] OF WORD	[3(0)]	number between -1.0000 and 1.0000
2	VAR	output_value	WORD	0	BCD value between 16#0 and 16#360 (0° and 360°) result: here 16#333
3	VAR				

Rumpf Dem ersten Element des ARRAY **input_value** wird der Wert 1 (negatives Vorzeichen) übergeben. In das zweite Element wird 0 als Vorkommawert und in das dritte Element 16#4500 als Nachkommawert geschrieben. Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Das Funktionsergebnis ist dann **output_value** = 16#333 (333°).

KOP



ST

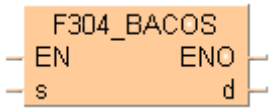
```
input_value[0]:=1;
input_value[1]:=0;
input_value[2]:=16#4500;
IF start THEN
    F303_BASIN(input_value, output_value);
END_IF;
```

F304_BACOS

Arcus-Cosinus-Funktion mit BCD-codierten Werten

Erklärung

Die Funktion berechnet den Arkuscosinus eines BCD-codierten Wertes, der in einem ARRAY mit drei Elementen dem Eingang **s** übergeben wird. Das Ergebnis wird als BCD-Wert im Bereich von 0° bis 360° (16#0 bis 16#360) am Ausgang **d** zurückgegeben.



BCD-Werte für den Eingang **s** liegen im Bereich von -1.0000 bis 1.0000. Sie werden wie folgt eingegeben:

ARRAY[0]	Vorzeichen des Eingangswertes (0 bei positivem Vorzeichen, 1 bei negativem Vorzeichen).
ARRAY[1]	Vorkommawert als ganze Zahl (0 oder 1).
ARRAY[2]	Nachkommastellen als BCD-codierter Wert auf 4 Stellen gerundet (16#0 bis 16#9999).

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F304_BACOS (s. S. 1190)

Datentypen

Variable	Datentyp	Funktion
s	ARRAY **[0..2] of** WORD	Bereich, in dem Winkelwerte in 3 Wörtern gespeichert werden
d	WORD	Ergebnis

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Eingangswert an s kein BCD-codierter Wert ist oder nicht zwischen -1.0000 und 1.0000 liegt.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

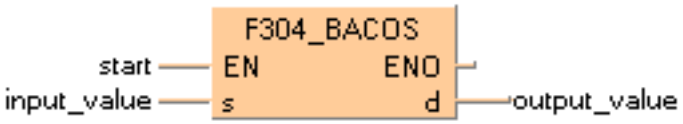
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	ARRAY [0..2] OF WORD	[2(0),16#8660]	number between -1.0000 and 1.0000
2	VAR	output_value	WORD	0	BCD value between 16#0 and 16#360 (0° and 360°) result: here 16#30
3	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Im ersten Element des ARRAY **input_value** steht 0 (positives Vorzeichen). Im zweiten Element steht 0 als Vorkommawert und im dritten Element die Nachkommastellen 8660. Die Funktion berechnet daraus den **output_value** = 16#30 (30°).

KOP



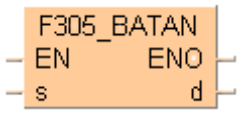
ST
```
IF start THEN
    F304_BACOS(input_value, output_value);
END_IF;
```

F305_BATAN

Arcus-Tangens-Funktion mit BCD-codierten Werten

Erklärung

Die Funktion berechnet den Arkustangens eines BCD-codierten Wertes, der in einem ARRAY mit drei Elementen dem Eingang **s** übergeben wird. Das Ergebnis wird als BCD-Wert im Bereich von 0° bis 90° (16#0 bis 16#90) oder von 270° bis 360° (16#270 bis 16#360) am Ausgang **d** zurückgegeben.



BCD-Werte für den Eingang **s** liegen im Bereich von -9999,9999 bis 9999,9999. Sie werden wie folgt eingegeben:

ARRAY[0]	Vorzeichen des Eingangswertes (0 bei positivem Vorzeichen, 1 bei negativem Vorzeichen).
ARRAY[1]	Vorkommawert als BCD-Wert (6#0 bis 16#9999).
ARRAY[2]	Nachkommastellen als BCD-codierter Wert auf 4 Stellen gerundet (16#0000 bis 16#9999).

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F305_BATAN (s. S. 1190)

Datentypen

Variable	Datentyp	Funktion
s	ARRAY **[0..2] of** WORD	Bereich, in dem Winkelwerte in 3 Wörtern gespeichert werden
d	WORD	Ergebnis

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Eingangswert an s kein BCD-Wert ist.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

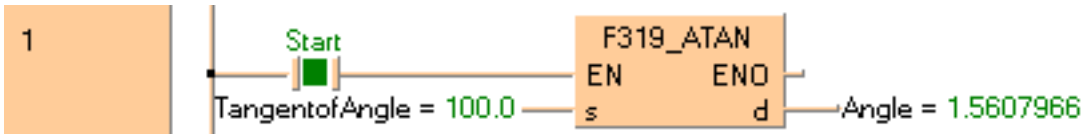
POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Start	BOOL	FALSE	
1	VAR	TangentofAngle	REAL	100.0	
2	VAR	Angle	REAL	0.0	range: -pi/2 to +pi/2 radians

Rumpf Wenn die Variable **Start** auf TRUE gesetzt wird, berechnet die Funktion den Arkustangens für die Variable **TangentofAngle**. Das Ergebnis wird in den Adressbereich geschrieben, den der Compiler der Variablen **Angle** zugewiesen hat.

KOP

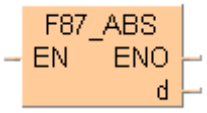


F87_ABS

16-Bit-Absolutbetrag

Erklärung

Vom 16-Bit-Wert des Speicherregisters **d** wird der absolute Betrag gebildet. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Das Ergebnis der Bildung des absoluten Betrags überschreibt den Inhalt des Speicherregisters **d**. Diesen Befehl können Sie zum Beispiel einsetzen, um bei Prozesssignalen mit wechselndem Vorzeichen immer eine positive Zahl zu bekommen.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden ABS (s. S. 65). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen **Verfügbarkeit von F87_ABS (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Speicherbereich für die Originaldaten und den Absolutbetrag

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der negative minimale Wert der der 16-Bit-Daten ist -32768 (16#8000).
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ der negative Wertebereich für 16-Bit-Daten liegt zwischen -1 und -32767 (16#FFFF bis 16#8001).

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	abs_value	INT	-123	result after a 0->1 leading edge from start: 123

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

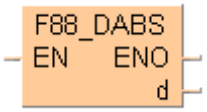
```
ST IF start THEN
       F87_ABS(abs_value);
   END_IF;
```

F88_DABS

32-Bit-Absolutbetrag

Erklärung

Vom 32-Bit-Wert des Speicherregisters **d** wird der absolute Betrag gebildet. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis der Bildung des absoluten Betrags überschreibt den Inhalt des Speicherregisters **d**. Diesen Befehl können Sie zum Beispiel einsetzen, um bei Prozesssignalen mit wechselndem Vorzeichen immer eine positive Zahl zu bekommen.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden ABS (s. S. 65). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen **Verfügbarkeit von F88_DABS (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
d	ANY32	32-Bit-Speicherbereich für die Originaldaten und den Absolutbetrag

Operanden

Für	Merker				T/C		Register			Konstante
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

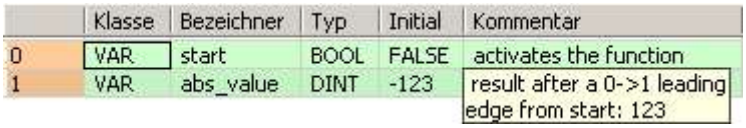
Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der negative minimale Wert der der 32-Bit-Daten ist -2147483648 (16#80000000).
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ der negative Wertebereich für 32-Bit-Daten liegt zwischen -1 und -2147483647 (16#FFFFFFFF bis 16#80000001).

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	abs_value	DINT	-123	result after a 0->1 leading edge from start: 123

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

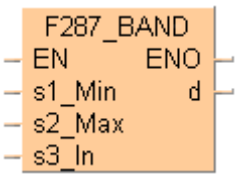
```
ST IF start THEN
       F88_DABS(abs_value);
   END_IF;
```

F287_BAND

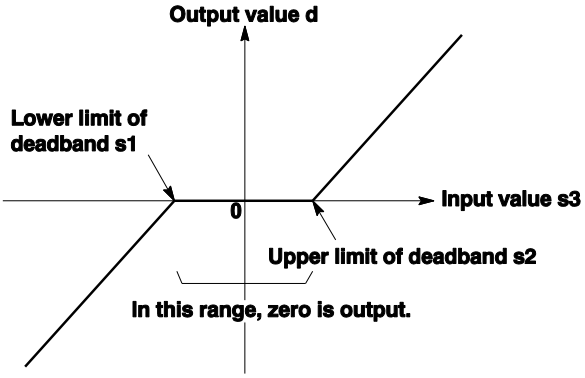
Totzonen-Ausgangssteuerung für 16-Bit-Daten

Erklärung

Die Funktion vergleicht den Eingangswert am Eingang **s3** mit dem unteren Grenzwert (Eingang **s1**) und dem oberen Grenzwert (Eingang **s2**) einer Totzone (Unempfindlichkeitsbereich). Das Funktionsergebnis wird am Ausgang **d** wie folgt zurückgegeben:



- Ist der Eingangswert am Eingang **s3** < **s1**, wird der untere Grenzwert des Eingangs **s1** vom Eingangswert an **s3** subtrahiert und das Ergebnis als Ausgangswert an **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3** > **s2**, wird der obere Grenzwert des Eingangs **s2** vom Eingangswert an **s3** subtrahiert und das Ergebnis als Ausgangswert an **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3** größer oder gleich dem unteren Grenzwert an **s1** und kleiner oder gleich dem oberen Grenzwert an **s2**, wird Null als Ausgangswert an **d** zurückgegeben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F287_BAND (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	der Bereich in dem der untere Grenzwert gespeichert ist oder die unteren Grenzwert-Daten
s2		der Bereich in dem der obere Grenzwert gespeichert ist oder die oberen Grenzwert-Daten
s3		der Bereich in dem der Eingangswert gespeichert ist oder die Eingangswert-Daten
d		der Bereich in dem die Ausgangswert-Daten gespeichert sind

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermer-ker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert an s1> s2 ist.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	TRUE	▪ der Eingangswert am Eingang s3 Null ist.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

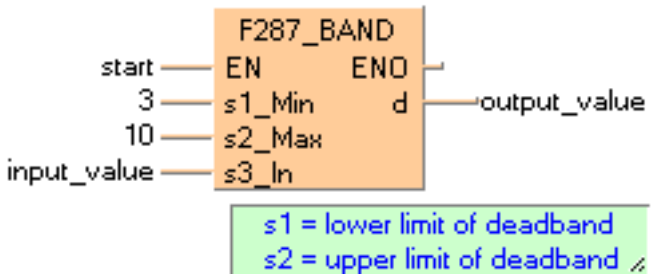
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	INT	12	
2	VAR	output_value	INT	0	result: here 2

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben (Freigabe-Eingang z.B. zum Testen).

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. An den Eingängen s1 und s2 stehen die Konstanten 3 (untere Totzone) und 10 (obere Totzone). Statt dessen können Sie auch Variablen im POE-Kopf deklarieren und im Rumpf an die Eingänge der Funktion schreiben.

KOP



ST

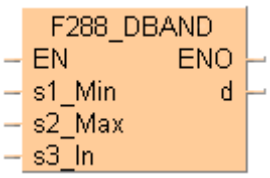
```
IF start THEN
    F287_BAND( 3, 10, input_value, output_value);
END_IF;     (* 3=lower limit of deadband, 10=upper limit of deadband *)
```

F288_DBAND

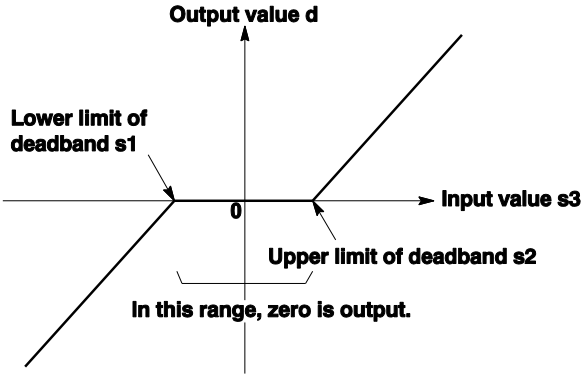
Totzonen-Ausgangssteuerung für 32-Bit-Daten

Erklärung

Die Funktion vergleicht den Eingangswert am Eingang **s3** mit dem unteren Grenzwert (Eingang **s1**) und dem oberen Grenzwert (Eingang **s2**) einer Totzone (Unempfindlichkeitsbereich). Das Funktionsergebnis wird am Ausgang **d** wie folgt zurückgegeben:



- Ist der Eingangswert am Eingang **s3** < **s1**, wird der untere Grenzwert des Eingangs **s1** vom Eingangswert an **s3** subtrahiert und das Ergebnis als Ausgangswert an **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3** > **s2**, wird der obere Grenzwert des Eingangs **s2** vom Eingangswert an **s3** subtrahiert und das Ergebnis als Ausgangswert an **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3** größer oder gleich dem unteren Grenzwert an **s1** und kleiner oder gleich dem oberen Grenzwert an **s2**, wird Null als Ausgangswert an **d** zurückgegeben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F288_DBAND (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	der Bereich in dem der untere Grenzwert gespeichert ist oder die unteren Grenzwert-Daten
s2		der Bereich in dem der obere Grenzwert gespeichert ist oder die oberen Grenzwert-Daten
s3		der Bereich in dem der Eingangswert gespeichert ist oder die Eingangswert-Daten
d		der Bereich in dem die Ausgangswert-Daten gespeichert sind

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert an s1> s2 ist.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ der Eingangswert am Eingang s3 Null ist.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

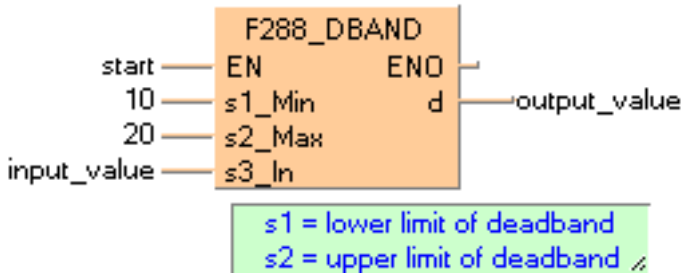
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	DINT	-22	
2	VAR	output_value	DINT	0	result: here -12

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben (Freigabe-Eingang z.B. zum Testen).

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. An den Eingängen s1 und s2 stehen die Konstanten -10 (untere Totzone) und 20 (obere Totzone). Statt dessen können Sie auch Variablen im POE-Kopf deklarieren und im Rumpf an die Eingänge der Funktion schreiben.

KOP

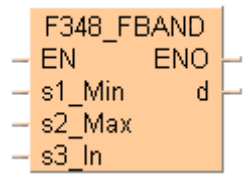


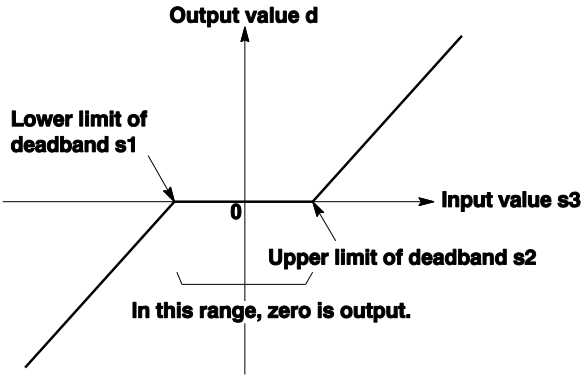
ST
```
IF start THEN
    F288_DBAND( -10, 20, input_value, output_value);
END_IF;     (* 10=lower limit of deadband, 20=upper limit of deadband *)
```

F348_FBAND

Totzonen-Ausgangsteuerung für Fließkommawerte

Erklärung Die Funktion vergleicht den Eingangswert am Eingang **s3** mit dem unteren Grenzwert (Eingang **s1**) und dem oberen Grenzwert (Eingang **s2**) einer Totzone (Unempfindlichkeitsbereich). Das Funktionsergebnis wird am Ausgang **d** wie folgt zurückgegeben:





Vergleich zwischen S1 und S2	Flag		
	R900A (> Flag)	R900B (= Flag)	R900C (< Flag)
s1 < s2	aus	aus	an
s1 ≤ s3 und s2 ≤ s1	aus	an	aus
s3 < s1	an	aus	aus

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F348_FBAND (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	REAL	der Bereich in dem der untere Grenzwert gespeichert ist oder die unteren Grenzwert-Daten
s2	REAL	der Bereich in dem der obere Grenzwert gespeichert ist oder die oberen Grenzwert-Daten
s3	REAL	der Bereich in dem der Eingangswert gespeichert ist oder die Eingangswert-Daten
d	REAL	der Bereich in dem die Ausgangswert-Daten gespeichert sind

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Werte an den Eingängen s1, s2 und s3 keine REALL-Zahlen sind oder der Wert an s1 > s2 ist.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

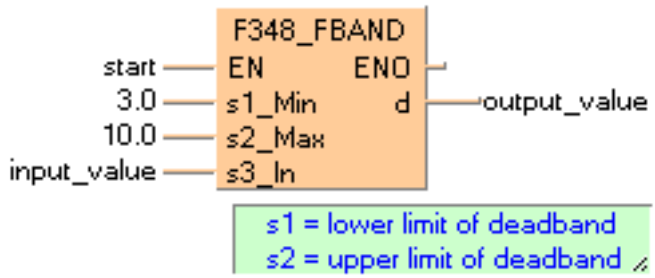
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	REAL	12.0	
2	VAR	output_value	REAL	0.0	result: here 2:0

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben (Freigabe-Eingang z.B. zum Testen).

Rumpf An den Eingängen s1 (untere Totzone) und s3 (obere Totzone) stehen die Konstanten 3.0 und 10.0. Statt dessen können Sie im POE-Kopf auch zwei Variablen deklarieren und am Rumpf an die Funktion schreiben. Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Da der **input_value** = 12.0 größer ist als der Wert der oberen Totzone an s2, ist der **output_value** = 12.0 -10.0 = 2.0.

KOP



ST

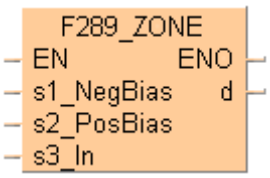
```
IF start THEN
    F348_FBAND( s1_Min:= 3.0 ,
        s2_Max:= 10.0 ,
        s3_In:= input_value ,
        d=> output_value )
END_IF;          (* 3.0=lower limit of deadband, 10.0=upper limit *)
```

F289_ZONE

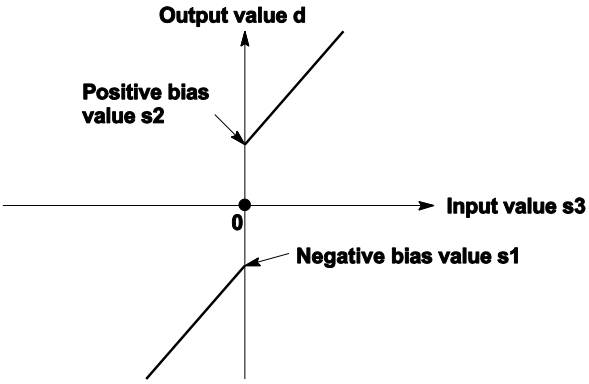
Offset-Ausgangssteuerung für 16-Bit-Daten

Erklärung

Die Funktion addiert zu einem Eingangswert am Eingang **s3** einen Offset-Wert. An den Eingängen **s1** und **s2** werden die Offset-Werte für den negativen und positiven Bereich eingegeben. Das Funktionsergebnis wird am Ausgang **d** wie folgt zurückgegeben:



- Ist der Eingangswert am Eingang **s3** < 0, wird der negative Offset-Wert an **s1** zum Eingangswert an **s3** addiert, und das Ergebnis wird als Ausgangswert am Ausgang **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3** = 0, wird Null als Ausgangswert am Ausgang **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3 >** 0, wird der positive Offset-Wert an **s2** zum Eingangswert an **s3** addiert, und das Ergebnis wird als Ausgangswert am Ausgang **d** zurückgegeben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F289_ZONE (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	der Bereich in dem der negative Vorspannungswert gespeichert ist oder negative Vorspannungswert-Daten
s2		der Bereich in dem der positive Vorspannungswert gespeichert ist oder positive Vorspannungswert-Daten
s3		der Bereich in dem der Eingangswert gespeichert ist oder Eingangswert-Daten
d		der Bereich in dem der Ausgangswert gespeichert ist

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das Funktionsergebnis zu einem Überlauf oder einem Unterlauf am Ausgang d führt.
R9009	%MX0.900.9	kurzzeitig	▪ der Eingangswert am Eingang s3 Null ist.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

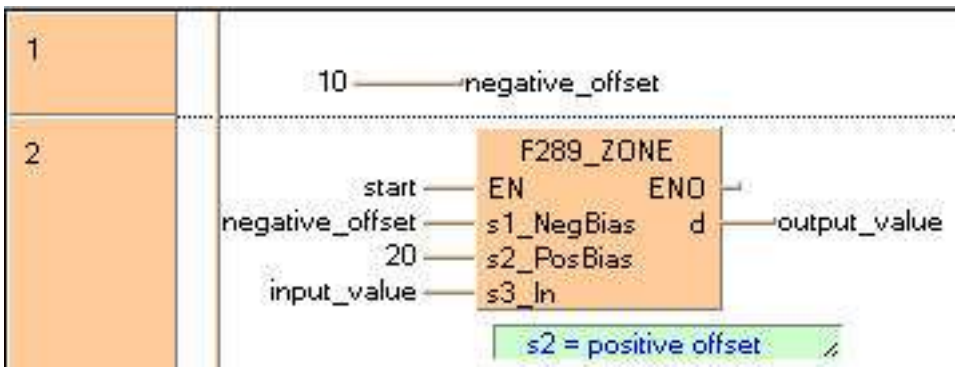
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	negative_offset	INT	0	
2	VAR	input_value	INT	-12	
3	VAR	output_value	INT	0	result: here -2

Hier wurden die Eingangsvariablen **input_value** und **negative_offset** deklariert. Statt dessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie addiert zum negativen **input_value** =-12 den entsprechenden negativen Offsetwert = 10. Statt dessen können Sie auch eine Variable im POE-Kopf deklarieren und im Rumpf an den Eingang der Funktion schreiben.

KOP



ST

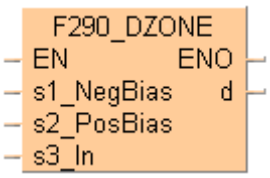
```
IF start THEN
    F289_ZONE( negative_offset, 20, input_value, output_value);
END_IF;       (*negative_offset=neg. offset, 20=pos. offset *)
```

F290_DZONE

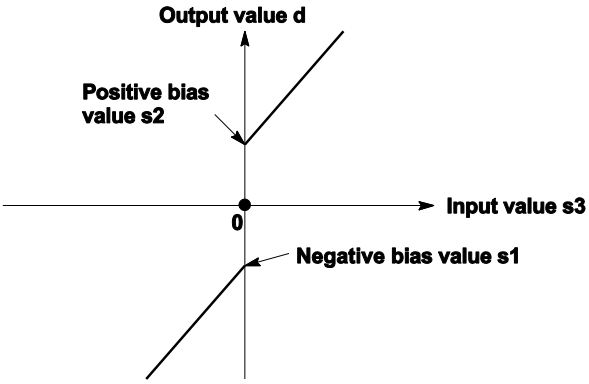
Offset-Ausgangssteuerung für 32-Bit-Daten

Erklärung

Die Funktion addiert zu einem Eingangswert am Eingang **s3** einen Offset-Wert. An den Eingängen **s1** und **s2** werden die Offset-Werte für den negativen und positiven Bereich eingegeben. Das Funktionsergebnis wird am Ausgang **d** wie folgt zurückgegeben:



- Ist der Eingangswert am Eingang **s3** < 0, wird der negative Offset-Wert an **s1** zum Eingangswert an **s3** addiert, und das Ergebnis wird als Ausgangswert am Ausgang **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3** = 0, wird Null als Ausgangswert am Ausgang **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3 >** 0, wird der positive Offset-Wert an **s2** zum Eingangswert an **s3** addiert, und das Ergebnis wird als Ausgangswert am Ausgang **d** zurückgegeben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F290_DZONE (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	der Bereich in dem der negative Vorspannungswert gespeichert ist oder negative Vorspannungswert-Daten
s2		der Bereich in dem der positive Vorspannungswert gespeichert ist oder positive Vorspannungswert-Daten
s3		der Bereich in dem der Eingangswert gespeichert ist oder Eingangswert-Daten
d		der Bereich in dem der Ausgangswert gespeichert ist

Operanden

Für	Merker				T/C		Register			Konst.
s1, s2, s3	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das Funktionsergebnis zu einem Überlauf oder einem Unterlauf am Ausgang d führt.
R9009	%MX0.900.9	kurzzeitig	▪ der Eingangswert am Eingang s3 Null ist.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

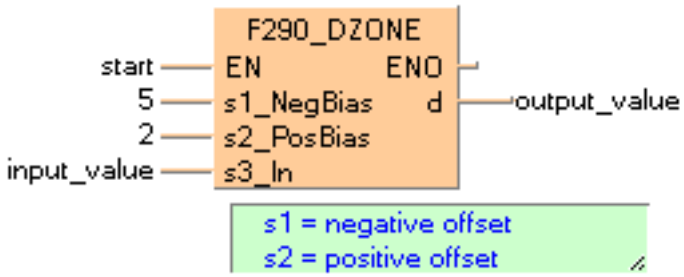
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	DINT	18	
2	VAR	output_value	DINT	0	result: here 20

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben (Freigabe-Eingang z.B. zum Testen).

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie addiert zum positiven Eingangswert = 18 den entsprechenden positiven Offsetwert = 2. An den Eingängen s1 und s2 stehen die Konstanten 5 (negativer Offset) und 2 (positiver Offset). Statt dessen können Sie auch Variablen im POE-Kopf deklarieren und im Rumpf an die Eingänge der Funktion schreiben.

KOP



ST

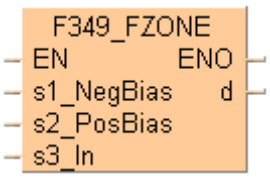
```
IF start THEN
    F290_DZONE( s1_NegBias := 5,
        s2_PosBias := 2,
        s3_In := input_value,
        d => output_value);
END_IF;        (*5=neg. offset, 2=pos. offset *)
```

F349_FZONE

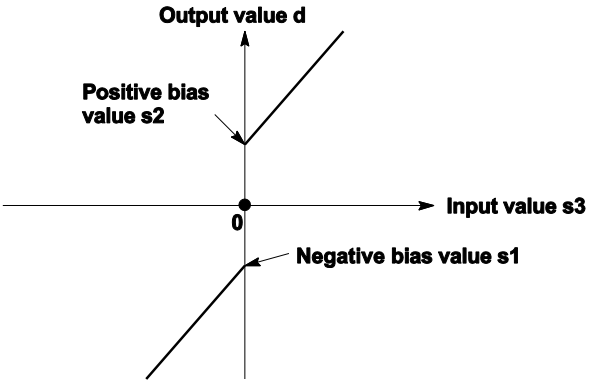
Offset-Ausgangssteuerung für Fließkommawerte

Erklärung

Die Funktion addiert zu einem Eingangswert am Eingang **s3** einen Offset-Wert. An den Eingängen **s1** und **s2** werden die Offset-Werte für den negativen und positiven Bereich eingegeben. Das Funktionsergebnis wird am Ausgang **d** wie folgt zurückgegeben:



- Ist der Eingangswert am Eingang **s3** < 0,0, wird der negative Offset-Wert an **s1** zum Eingangswert an **s3** addiert, und das Ergebnis wird als Ausgangswert am Ausgang **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3** = 0,0, wird Null als Ausgangswert am Ausgang **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3 >** 0,0, wird der positive Offset-Wert an **s2** zum Eingangswert an **s3** addiert, und das Ergebnis wird als Ausgangswert am Ausgang **d** zurückgegeben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F349_FZONE (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	REAL	der Bereich in dem der negative Vorspannungswert gespeichert ist oder negative Vorspannungswert-Daten
s2	REAL	der Bereich in dem der positive Vorspannungswert gespeichert ist oder positive Vorspannungswert-Daten
s3	REAL	der Bereich in dem der Eingangswert gespeichert ist oder Eingangswert-Daten
d	REAL	der Bereich in dem der Ausgangswert gespeichert ist

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ wenn die Werte an den Eingängen s1, s2 und s3 keine REAL-Zahlen sind.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

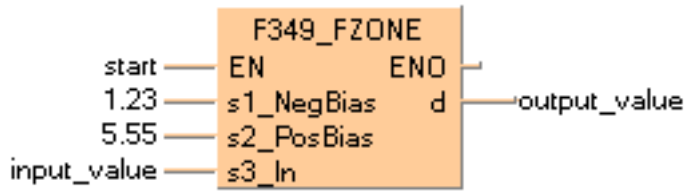
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	REAL	-10.0	
2	VAR	output_value	REAL	0.0	result: here -11.23

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben (Freigabe-Eingang z.B. zum Testen).

Rumpf An den Eingängen s1 (negativer Offset) und s2 (positiver Offset) stehen die Konstanten -1.23 und 5.55. Statt dessen können Sie im POE-Kopf auch zwei Variablen deklarieren und am Rumpf an die Funktion schreiben. Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Da der **input_value** negativ ist (-10.0), wird der negative Offset -1,23 addiert. Das Funktionsergebnis ist hier: **output_value** = -11.23.

KOP



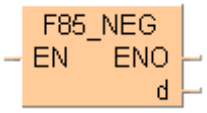
ST

```
IF start THEN
    F349_FZONE( s1_NegBias := -1.23 ,
        s2_PosBias := 5.55 ,
        s3_In := input_value ,
        d => output_value );
END_IF;          (*-1.23=neg. offset, 5.55=pos. offset *)
```

F85_NEG

16-Bit-Zweierkomplement

Erklärung Vom 16-Bit-Wert des Speicherregisters **d** wird das Zweierkomplement gebildet. Das Ergebnis der Zweierkomplementbildung überschreibt den Inhalt des Speicherregisters **d**.



Die Bildung des Zweierkomplements bedeutet, dass das Doppelwort zuerst bitweise invertiert und dann der Wert "1" addiert wird. Dieser Befehl wird eingesetzt, wenn binärcodierte Integerzahlen mit Vorzeichen und einer Breite von 16 Bits negiert werden sollen.

Dieser Befehl ist sinnvoll, um das Vorzeichen von 16-Bit-Daten von positiv in negativ oder von negativ in positiv umzuwandeln.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F85_NEG (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Speicherbereich für Originaldaten und seine Zweierkomplementbildung

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

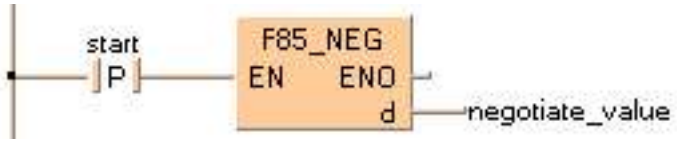
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	negotiate_value	WORD	2#1001001101110001	result after a 0->1 leading edge from start: 2#0110110010001111

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

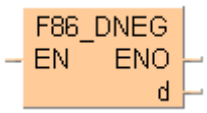


ST
```
IF DF(start) THEN
    F85_NEG(negotiate_value);
END_IF;
```

F86_DNEG

32-Bit-Zweierkomplement

Erklärung Vom 32-Bit-Wert des Speicherregisters **d** wird das Zweierkomplement gebildet. Das Ergebnis der Zweierkomplementbildung überschreibt den Inhalt des Speicherregisters **d**.



Die Bildung des Zweierkomplements bedeutet, dass das Doppelwort zuerst bitweise invertiert und dann der Wert "1" addiert wird. Dieser Befehl wird eingesetzt, wenn binärcodierte Integerzahlen mit Vorzeichen und einer Breite von 16 Bits negiert werden sollen.

Dieser Befehl ist sinnvoll, um das Vorzeichen von 16-Bit-Daten von positiv in negativ oder von negativ in positiv umzuwandeln.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F86_DNEG (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
d	ANY32	32-Bit-Speicherbereich für Originaldaten und seine Zweierkomplementbildung

Operanden

Für	Merker				T/C		Register			Konstante
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Beispiel In diesem Beispiel wird die Funktion F86_DNEG im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	negotiate_value	DWORD	2#11010001000011000110000011101111	result after a 0->1 leading edge from start: 2#00101110111110011 1001111100010001

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

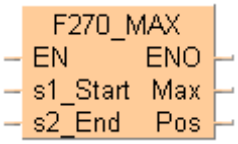
KOP

ST
```
IF DF(start) THEN
    F86_DNEG(negotiate_value);
END_IF;
```

F270_MAX

Maximalwert einer 16-Bit-Datentabelle

Erklärung Die Funktion sucht den Maximalwert und seine Position einer 16-Bit-Datentabelle und gibt beide Werte aus.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Der Maximalwert wird am Ausgang **max** und seine Position am Ausgang **pos** zurückgegeben.

Die Position **pos** entspricht der relativen Position vom Anfang der Datentabelle bis zum ersten Vorkommen des Maximalwertes.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F270_MAX (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Anfang der Datentabelle
s2		Ende der Datentabelle
max	INT	Spezifiziert den Maximalwert
pos	INT	Position, an der der Maximalwert gefunden wurde

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
max, pos	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereiche s1 und s2 verschieden sind.
R9008	%MX0.900.8	kurzzeitig	

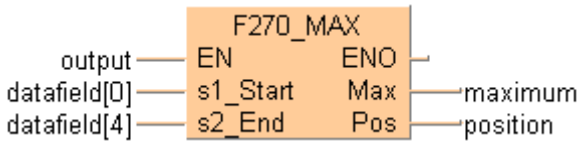
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF INT	[2,3,6,-3,1]	Arbitrarily large data field
2	VAR	maximum_value	INT	0	result: here 6
3	VAR	position	INT	0	result: here 2

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie sucht in **data_field** nach dem Maximalwert und seiner Position. Das Funktionsergebnis ist hier: **maximum_value** = 6 and **position** = 2.

KOP



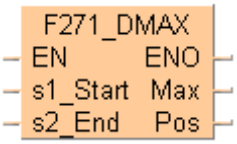
ST

```
IF start THEN
    F270_MAX( s1_Start := data_field[0],
        s2_End := data_field[4],
        Max => maximum_value,
        Pos => position);
END_IF;
```

F271_DMAX

Maximalwert einer 32-Bit-Datentabelle

Erklärung Die Funktion sucht den Maximalwert und seine Position einer 32-Bit-Datentabelle und gibt beide Werte aus.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Der Maximalwert wird am Ausgang **max** und seine Position am Ausgang **pos** zurückgegeben.

Die Position **pos** entspricht der relativen Position vom Anfang der Datentabelle bis zum ersten Vorkommen des Maximalwertes.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F271_DMAX (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Anfang der Datentabelle
s2		Ende der Datentabelle
max	DINT	Spezifiziert den Maximalwert
pos	WORD	Position, an der der Maximalwert gefunden wurde

Operanden

Für	Merker				T/C		Register			Konst.
s1, s2	DWX	DWY	DWF	DWL	DSV	DEV	DDT	DLD	DFL	-
max	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
pos	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereiche s1 und s2 verschieden sind.
R9008	%MX0.900.8	kurzzeitig	

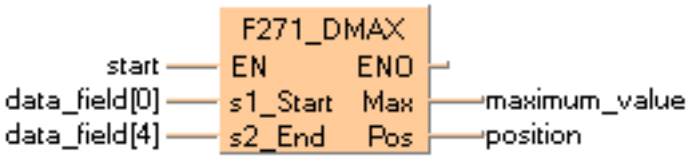
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF DINT	[2,3,222222,-333333,1]	Arbitrarily large data field
2	VAR	maximum_value	DINT	0	result: here 222222
3	VAR	position	INT	0	result: here 2

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie sucht in **data_field** nach dem Maximalwert und seiner Position. Das Funktionsergebnis ist hier: **maximum_value** = 222222 and **position** = 2.

KOP



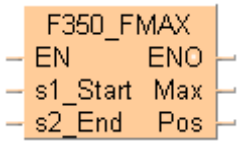
ST

```
IF start THEN
    F271_DMAX( s1_Start := data_field[0],
         s2_End := data_field[4],
         Max => maximum_value,
         Pos => position);
END_IF;
```

F350_FMAX

Maximalwert einer Tabelle mit Fließkommawerten

Erklärung Die Funktion sucht den Maximalwert und seine Position in einer Fließkomma-Datentabelle und gibt beide Werte aus.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Der Maximalwert wird am Ausgang **max** und seine Position am Ausgang **pos** zurückgegeben.

Die Adresse des Maximalwertes am Ausgang **pos** ist relativ zur Anfangsadresse der Datentabelle am Eingang **s1**.

Falls mehrere Maximalwerte gefunden werden, wird die relative Adresse des ersten, ausgehend von der Anfangsadresse am Eingang **s1** gefundenen Maximalwertes in **d** gespeichert.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F350_FMAX (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	REAL	Anfang der Datentabelle
s2	REAL	Ende der Datentabelle
max	REAL	Spezifiziert den Maximalwert
pos	INT	Position, an der der Maximalwert gefunden wurde

Operanden

Für	Merker				T/C		Register			Konst.
s1, s2	DWX	DWY	DWF	DWL	DSV	DEV	DDT	DLD	DFL	-
max	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
pos	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereich verschieden sind. ▪ die Fließkommawerte ihren möglichen Bereich überschreiten.
R9008	%MX0.900.8	kurzzeitig	

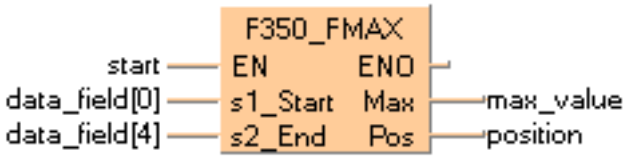
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0...4] OF REAL	[2.0,3.45,-6.91,5.44,1.3]	Arbitrarily large data field
2	VAR	max_value	REAL	0.0	result: here 5.44
3	VAR	position	INT	0	result: here 3

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie sucht in **data_field** nach dem Maximalwert und seiner Position. Das Funktionsergebnis ist hier: **max_value** = 5.44 and **position** = 3.

KOP



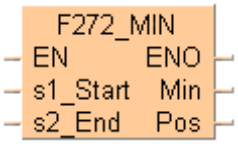
ST

```
IF start THEN
    F350_FMAX( s1_Start:= data_field[0],
        s2_End:= data_field[4],
        Max=> max_value,
        Pos=> position);
END_IF;
```

F272_MIN

Minimalwert einer 16-Bit-Datentabelle

Erklärung Die Funktion sucht den Minimalwert und seine Position einer 16-Bit-Datentabelle und gibt beide Werte aus.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Der Minimalwert wird am Ausgang **min** und seine Position am Ausgang **pos** zurückgegeben.

Die Position **pos** entspricht der relativen Position vom Anfang der Datentabelle bis zum ersten Vorkommen des Minimalwertes.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F272_MIN (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Anfang der Datentabelle
s2		Ende der Datentabelle
min	INT	Spezifiziert Minimalwert
pos	INT	Position, an der der Minimalwert gefunden wurde

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
min, pos	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereiche s1 und s2 verschieden sind.
R9008	%MX0.900.8	kurzzeitig	

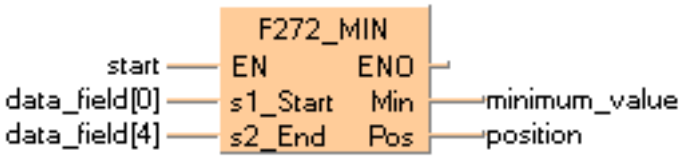
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF INT	[2,3,6,-3,1]	Arbitrarily large data field
2	VAR	minimum_value	INT	0	result: here -3
3	VAR	position	INT	0	result: here 3

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie sucht in **data_field** nach dem Minimalwert und seiner Position. Das Funktionsergebnis ist hier: **minimum_value** = -3 and **position** = 3.

KOP



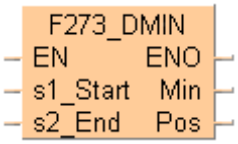
ST

```
IF start THEN
    F272_MIN( s1_Start:= data_field[0],
        s2_End:= data_field[4],
        Min=> minimum_value,
        Pos=> position);
END_IF;
```

F273_DMIN

Minimalwert einer 32-Bit-Datentabelle

Erklärung Die Funktion sucht den Minimalwert und seine Position einer 32-Bit-Datentabelle und gibt beide Werte aus.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Der Minimalwert wird am Ausgang **min** und seine Position am Ausgang **pos** zurückgegeben.

Die Position **pos** entspricht der relativen Position vom Anfang der Datentabelle bis zum ersten Vorkommen des Minimalwertes.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F273_DMIN (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Anfang der Datentabelle
s2		Ende der Datentabelle
min	DINT	Spezifiziert Minimalwert
pos	INT	Position, an der der Minimalwert gefunden wurde

Operanden

Für	Merker				T/C		Register			Konst.
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
min	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
pos	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereiche s1 und s2 verschieden sind.
R9008	%MX0.900.8	kurzzeitig	

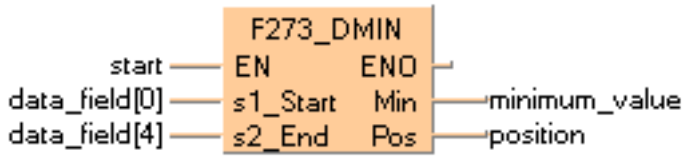
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF DINT	[2,3,222222,-333333,1]	Arbitrarily large data field
2	VAR	minimum_value	DINT	0	result: here -333333
3	VAR	position	INT	0	result: here 3

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie sucht in **data_field** nach dem Minimalwert und seiner Position. Das Funktionsergebnis ist hier: **minimum_value** = -333333 and **position** = 3.

KOP



ST

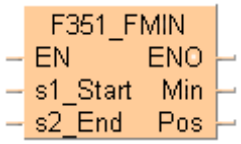
```
IF start THEN
     F273_DMIN( s1_Start:= data_field[0],
          s2_End:= data_field[4],
          Min=> minimum_value,
          Pos=> position);
END_IF;
```

F351_FMIN

Minimalwert einer Tabelle mit Fließkommawerten

Erklärung

Die Funktion sucht den Minimalwert und seine Position in einer Fließkomma-Datentabelle und gibt beide Werte aus.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Der Minimalwert wird am Ausgang **min** und seine Position am Ausgang **pos** zurückgegeben.

Die Adresse des Maximalwertes am Ausgang **pos** ist relativ zur Anfangsadresse der Datentabelle am Eingang **s1**.

Falls mehrere Minmalwerte gefunden werden, wird die relative Adresse des ersten, ausgehend von der Anfangsadresse am Eingang **s1** gefundenen Maximalwertes in **d** gespeichert.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F351_FMIN (s. S. 1191)

Datentypen

Variable	Datentyp	Funktion
s1	REAL	Anfang der Datentabelle
s2	REAL	Ende der Datentabelle
min	REAL	Spezifiziert Minimalwert
pos	INT	Position, an der der Minimalwert gefunden wurde

Operanden

Für	Merker				T/C		Register			Konst.
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
min	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
pos	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereich verschieden sind. ▪ die Fließkommawerte ihren möglichen Bereich überschreiten.
R9008	%MX0.900.8	kurzzeitig	

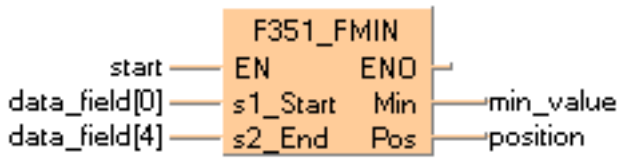
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF REAL	[2.0,3.45,-6.91,5.44,1.3]	Arbitrarily large data field
2	VAR	min_value	REAL	0.0	result: here -6.91
3	VAR	position	INT	0	result: here 2

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie sucht in **data_field** nach dem Minimalwert und seiner Position. Das Funktionsergebnis ist hier: **min_value** = 6.91 and **position** = 2.

KOP



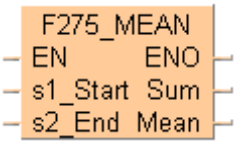
ST
```
IF start THEN
    F351_FMIN( s1_Start := data_field[0],
               s2_End := data_field[4],
               Min => min_value ,
               Pos => position );
END_IF;
```

F275_MEAN

Summe und arithmetischer Mittelwert einer 16-Bit-Datentabelle

Erklärung

Die Funktion berechnet die Summe und den arithmetischen Mittelwert (beides mit Vorzeichen) von Werten, die in einer 16-Bit-Datentabelle stehen.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Die Summe aller Elemente der Datentabelle wird am Ausgang **sum** und der arithmetische Mittelwert über alle Elemente der Datentabelle am Ausgang **mean** zurückgegeben. Wenn der arithmetische Mittelwert keine ganze Zahl ergibt, wird nur der Vorkommawert zurückgegeben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F275_MEAN (s. S. 1189)

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Anfang der Datentabelle
s2		Ende der Datentabelle
mean	INT	Mittelwert aller in der Datentabelle spezifizierten Elemente
Summe	DINT	Summe aller in der Datentabelle spezifizierten Elemente

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
mean	-	WY	WR	WL	SV	EV	DT	LD	FL	-
Summe	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereich verschieden sind.
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ wenn es zu einem Überlauf bzw. Unterlauf während der Funktionsausführung kommt.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

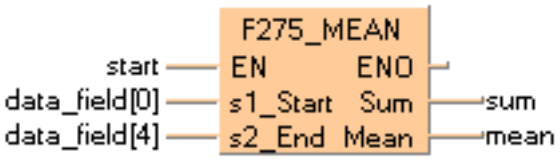
POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF INT	[2,3,6,-3,1]	Arbitrarily large data field
2	VAR	sum	DINT	0	result: here 9
3	VAR	mean	INT	0	result: here 1

Rumpf Wenn die Variable **output** auf den Wert TRUE gesetzt ist, wird die Funktion F275_MEAN ausgeführt. Die Funktion berechnet die Summe aller Elemente der Datentabelle (Summe = 4 + 3 + 8 + (-2) + 1 + (-6) = 8) und schreibt das Ergebnis (hier 8) in die Variable **sum.** Zusätzlich berechnet die Funktion den arithmetischen Mittelwert über alle Elemente der Datentabelle (Mittelwert = Summe/6 = (4 + 3 + 8 + (-2) + 1 + (-6)) / 6 = 1.333) und schreibt den Vorkommawert (hier 1) in die Variable **mean**.

KOP



ST

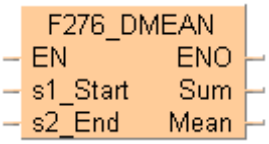
```
IF start THEN
    F275_MEAN( s1_Start:= data_field[0],
        s2_End:= data_field[4],
        Sum=> sum,
        Mean=> mean);
END_IF;
```

F276_DMEAN

Summe und arithmetischer Mittelwert einer 32-Bit-Datentabelle

Erklärung

Die Funktion berechnet die Summe und den arithmetischen Mittelwert (beides mit Vorzeichen) von Werten, die in einer 32-Bit-Datentabelle stehen.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Die Summe aller Elemente der Datentabelle wird am Ausgang **sum** und der arithmetische Mittelwert über alle Elemente der Datentabelle am Ausgang **mean** zurückgegeben. Wenn der arithmetische Mittelwert keine ganze Zahl ergibt, wird nur der Vorkommawert zurückgegeben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F276_DMEAN (s. S. 1189)

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Anfang der Datentabelle
s2		Ende der Datentabelle
mean	DINT	Mittelwert aller in der Datentabelle spezifizierten Elemente
Summe	ARRAY **[0..1] of** DINT	Summe aller in der Datentabelle spezifizierten Elemente

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
mean, sum	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereich verschieden sind.
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ wenn es zu einem Überlauf bzw. Unterlauf während der Funktionsausführung kommt.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

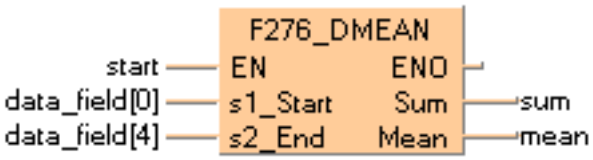
POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	output	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF DINT	[2,3,222222,-333333,1]	Arbitrarily large data field
2	VAR	sum	ARRAY [0..1] OF DINT	[2(0)]	result: here
3	VAR	mean	DINT	0	result: here -22221

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Funktion berechnet die Summe aller Elemente des ARRAY **data_field** (Summe = 2 + 3 + 222222 + (-333333) + 1 = -111105) und übergibt den Wert der Variablen **sum**. Zusätzlich berechnet die Funktion den Mittelwert (Mittelwert = Summe/5 = -111105/5 = -22221) und übergibt diesen der Variablen **mean**.

KOP



ST

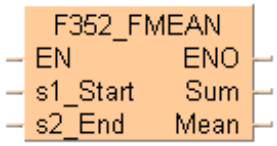
```
IF start THEN
    F276_DMEAN( s1_Start:= data_field[0],
        s2_End:= data_field[4],
        Sum=> sum,
        Mean=> mean);
END_IF;
```

F352_FMEAN

Summe und arithmetischer Mittelwert einer 16-Bit-Datentabelle

Erklärung

Die Funktion berechnet die Summe und den arithmetischen Mittelwert (beides mit Vorzeichen) von Fließkommawerten, die in einer 32-Bit-Datentabelle stehen.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Die Summe aller Elemente der Datentabelle wird am Ausgang **sum** und der arithmetische Mittelwert über alle Elemente der Datentabelle am Ausgang **mean** zurückgegeben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F352_FMEAN (s. S. 1191)

Datentypen

Variable	Datentyp	Funktion
s1	REAL	Anfang der Datentabelle
s2	REAL	Ende der Datentabelle
mean	REAL	Mittelwert aller in der Datentabelle spezifizierten Elemente
Summe	REAL	Summe aller in der Datentabelle spezifizierten Elemente

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
mean, sum	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereich verschieden sind. ▪ die Fließkommawerte ihren möglichen Bereich überschreiten.
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ wenn bei der Funktionsausführung ein Überlauf bzw. Unterlauf auftritt.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

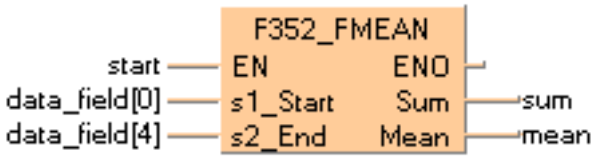
POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF REAL	[2.0,3.45,-6.91,5.44,1.3]	Arbitrarily large data field
2	VAR	sum	REAL	0.0	result: here 5.28
3	VAR	mean	REAL	0.0	result: here 1.056

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie berechnet **sum** = 2.0 + 3.45 + (-6.91) + 5.44 + 1.3 = 5.28 und **mean = Sum**/5 = 5.28/5 = 1.056 der Elemente von **data_field**.

KOP



ST

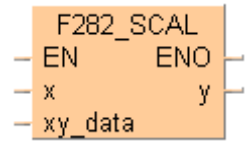
```
IF start THEN
     F352_FMEAN( s1_Start:= data_field[0] ,
          s2_End:= data_field[4] ,
          Sum=> sum ,
          Mean=> mean );
END_IF;
```

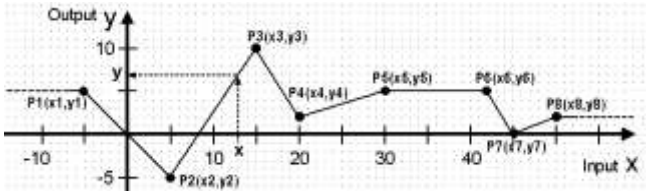
F282_SCAL

Interpolation einer integer-Messkurve

Erklärung

Die Funktion ermittelt an der Stelle **x** den Funktionswert **y** durch lineare Interpolation zwischen den benachbarten Stützstellen Pw(xw,yw) und Pw+1(xw+1,yw+1). Hier ist w die nächste Stützstelle, deren x-Wert kleiner als der Eingangswert x ist. D.h. die Funktion verbindet nacheinander die einzelnen Stützstellen und ermittelt aus dem Eingangswert **x** den zugehörigen Ausgangswert **y**.





Die Funktion kann verwendet wenden für:

- die Linearisierung von Messwerten z. B. bei nichtlinearen Sensoren
- die Ermittlung der Vorlauftemperatur y einer Heizung in Abhängigkeit von der Außentemperatur x
- usw.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F282_SCAL (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
x	INT	Eingangswert **x**
xy_daten	SDT	Erstes Element einer Variablen von Typ SDT die die xy-Wertepaare enthält
y	INT	Ausgangswert **y**
EN	BOOL	Aktivierung der Funktion (bei EN = TRUE wird die Funktion bei jedem SPS-Zyklus abgearbeitet)
ENO	BOOL	ENO wird auf den Wert TRUE gesetzt, sobald die Funktion abgearbeitet ist. Hilfreich bei der Kaskadierung von Bausteinen mit EN-Funktion.

Operanden

Für	Merker				T/C		Register			Konstante
x	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
y	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ an, wenn die Anzahl an Stützstellen nicht zwischen 2 .. 100 liegt oder die x-Werte keine homogen steigende Reihenfolge (x1 < x2 < x3 < ...) haben. ▪
R9008	%MX0.900.8	kurzzeitig	

■ **Begrenzung des Ausgangswertes y:**

Ist der Eingangswert **x** kleiner als die x-Koordinate der ersten Stützstelle (P1: x < x1), wird der Ausgang **y** auf die y-Koordinate der ersten Stützstelle gesetzt (Ausgang **y** = y1, waagrechte gestrichelte Linie links im Graph oben).

Ist der Eingangswert **x** größer als die x-Koordinate der letzten Stützstelle (P8: x > x8), wird der Ausgang **y** auf die y-Koordinate der ersten Stützstelle gesetzt (Ausgang **y** = y8, waagrechte gestrichelte Linie rechts oben im Graph).

■ **SDT für die xy-Wertepaare (Stützstellen P1, P2, ...):**

Die Stützstellen (P1, P2, ...) werden der Funktion durch eine Variable vom Type SDT die die Anzahl der Stützstellen und die xy-Wertepaare (Anzahl; x1, x2, ...; y1, y2; ...) enthält übergeben.

Aufbau des SDT

- Eintrag: Variable die die Anzahl an Stützstellen enthält von Datentyp INT.
  Die Anzahl an Stützstellen (xy-Wertepaare) kann beliebig zwischen 2 .. 100 eingestellt werden. In der Grafik wurden acht Stützstellen (P1 .. P8) verwendet.
- Eintrag: Variable die die x-Werte enthält von Datentyp ARRAY [0..z] OF INT.
  Hier steht z als Platzhalter für die Anzahl an Stützstellen (siehe 1. Eintrag).
- Eintrag: Variable die die y-Werte enthält von Datentyp ARRAY [0..z] OF INT.
  Hier steht z als Platzhalter für die Anzahl an Stützstellen (siehe 1. Eintrag).

■ **Wichtige Informationen:**

x-Werte

Die x-Werte müssen in homogen steigender Reihenfolge (x1 < x2 < x3 < ...) eingetragen sein. Sind gleiche x-Werte vorhanden (z.B. x2 = x3 = x4) werden die Stützstellen P2(x2,y2) und P3(x3,y3) ignoriert.

Überlauf in der Funktion:

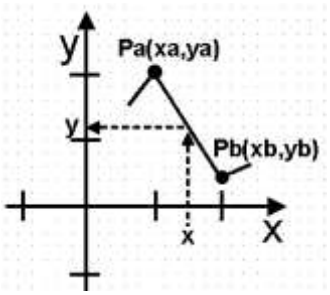
Um einen Berechnungsüberlauf in der Funktion zu vermeiden, müssen benachbarte Stützstellen folgende Bedingungen erfüllen:

|ya - yb| < 32767

|x - xb| < 32767

|(ya - yb)*(x - xb)| < 32767

|xa - xb| < 32767



Berechnungsgenauigkeit:

Diese Funktion kann ausschließlich ganze Zahlen verarbeiten. Nachkommastellen bei der Berechnung des Funktionswertes y werden abgeschnitten. Zum Beispiel ein intern berechneter Funktionswert **y** an einer Stelle **x** z.B. **y** = 511,13 ergibt Rückgabewert der Funktion = 511.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

SDT Im SDT-Pool wird ein strukturierter Datentyp angelegt in dem die Anzahl der Stützstellen und die xy-Wertepaare deklariert werden.

Interpolation_8 [SDT]

	Bezeichner	Typ	Initial	Kommentar
0	referencepoints	INT	8	eight reference points were
1	X_values	ARRAY [1..8] OF INT	[8(0)]	Field 1..8 -> contains 8 x-values
2	Y_values	ARRAY [1..8] OF INT	[8(0)]	Field 1..8 -> contains 8 y-values

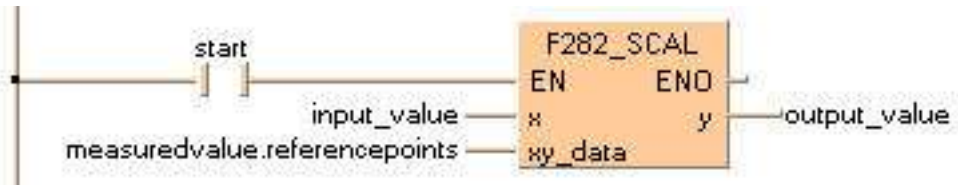
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	avtivates the function
1	VAR	input_value	INT	0	input_value x
2	VAR	measured_value	Interpolation_8	X_values := [-5,5,15,20,30,42,45,50],Y_values := [5,-5,10,2,2(5),0,2]	number of reference points
3	VAR	output_value	INT	0	output_value y

Hier wurde die Eingangsvariable **measured_value** von Typ des oben angelegten SDT deklariert. Die Belegung der x-Werte und y-Werte wurde bereits im POE-Kopf vorgenommen. Statt dessen können Sie im Rumpf auch die x-Werte und y-Werte verändern indem Sie einen Wert auf die Variable, z.B. **Measuredvalues.X_Values[1]** für x **schreiben.**

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie berechnet für den Eingangswert an der Stelle x den Ausgangswert y durch lineare Interpolation zwischen den benachbarten Stützstellen die in der Variablen **measured value** gespeichert sind.

KOP



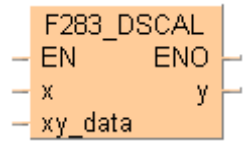
ST

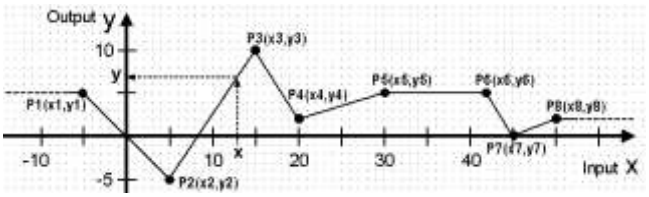
```
IF start THEN
    F282_SCAL(input_value, measured_value.referencepoints, output_value);
END_IF;
```

F283_DSCAL

Interpolation einer Doubleinteger-Meßkurve

Erklärung Die Funktion ermittelt an der Stelle **x** den Funktionswert **y** durch lineare Interpolation zwischen den benachbarten Stützstellen Pw(xw,yw) und Pw+1(xw+1,yw+1). Hier ist w die nächste Stützstelle, deren x-Wert kleiner als der Eingangswert x ist. D.h. die Funktion verbindet nacheinander die einzelnen Stützstellen und ermittelt aus dem Eingangswert **x** den zugehörigen Ausgangswert **y**.





Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Die Funktion kann verwendet wenden für:

- die Linearisierung von Messwerten z. B. bei nichtlinearen Sensoren
- die Ermittlung der Vorlauftemperatur y einer Heizung in Abhängigkeit von der Außentemperatur x
- usw.

SPS-Typen **Verfügbarkeit von F283_DSCAL (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
x	DINT	Eingangswert **x**
xy_daten	SDT	Erstes Element einer Variablen von Typ SDT, die die xy-Wertepaare enthält
y	DINT	Ausgangswert **y**
EN	BOOL	Aktivierung der Funktion (bei EN = TRUE wird die Funktion bei jedem SPS-Zyklus abgearbeitet)
ENO	BOOL	ENO wird auf den Wert TRUE gesetzt, sobald die Funktion abgearbeitet ist. Hilfreich bei der Kaskadierung von Bausteinen mit EN-Funktion.

Operanden

Für	Merker				T/C		Register			Konstante
x	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
y	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ an, wenn die Anzahl an Stützstellen nicht zwischen 2 .. 100 liegt oder die x-Werte keine homogen steigende Reihenfolge (x1 < x2 < x3 < ...) haben.
R9008	%MX0.900.8	kurzzeitig	

■ **Begrenzung des Ausgangswertes y:**

Ist der Eingangswert **x** kleiner als die x-Koordinate der ersten Stützstelle (P1: x < x1), wird der Ausgang **y** auf die y-Koordinate der ersten Stützstelle gesetzt (Ausgang **y** = y1, waagrechte gestrichelte Linie links im Graph oben).

Ist der Eingangswert **x** größer als die x-Koordinate der letzten Stützstelle (P8: x > x8), wird der Ausgang **y** auf die y-Koordinate der ersten Stützstelle gesetzt (Ausgang **y** = y8, waagrechte gestrichelte Linie rechts oben im Graph).

■ **SDT für die xy-Wertepaare (Stützstellen P1, P2, ...):**

Die Stützstellen (P1, P2, ...) werden der Funktion durch eine Variable vom Type SDT die die Anzahl der Stützstellen und die xy-Wertepaare (Anzahl; x1, x2, ...; y1, y2; ...) enthält übergeben.

Aufbau des SDT

- Eintrag: Variable die die Anzahl an Stützstellen enthält von Datentyp INT.
  Die Anzahl an Stützstellen (xy-Wertepaare) kann beliebig zwischen 2 .. 100 eingestellt werden. 100. In der Grafik wurden acht Stützstellen (P1 .. P8) verwendet.
- Eintrag: Variable die die x-Werte enthält von Datentyp ARRAY [0..z] OF DINT.
  Hier steht **z** als Platzhalter für die Anzahl an Stützstellen (siehe 1. Eintrag).
- Eintrag: Variable die die x-Werte enthält von Datentyp ARRAY [0..z] OF DINT.
  Hier steht **z** als Platzhalter für die Anzahl an Stützstellen (siehe 1. Eintrag).

■ **Wichtige Informationen:**

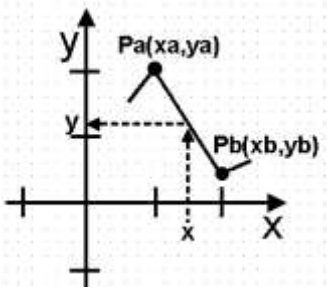
x-Werte

Die x-Werte müssen in homogen steigender Reihenfolge (x1 < x2 < x3 < ...) eingetragen sein. Sind gleiche x-Werte vorhanden (z.B. x2 = x3 = x4) werden die Stützstellen P2(x2,y2) und P3(x3,y3) ignoriert.

Überlauf in der Funktion:

Um einen Berechnungsüberlauf in der Funktion zu vermeiden, müssen benachbarte Stützstellen folgende Bedingungen erfüllen:

|ya - yb| < 2147483647
|x - xb| < 2147483647
|(ya - yb)*(x - xb)| < 2147483647
|xa - xb| < 2147483647



Berechnungsgenauigkeit:

Diese Funktion kann ausschließlich ganze Zahlen verarbeiten. Nachkommastellen bei der Berechnung des Funktionswertes y werden abgeschnitten. Zum Beispiel ein intern berechneter Funktionswert **y** an einer Stelle **x** z.B. **y** = 511,13 ergibt Rückgabewert der Funktion = 511.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

SDT

Im SDT-Pool wird ein strukturierter Datentyp angelegt in dem die Anzahl der Stützstellen und die xy-Wertepaare deklariert werden.

Interpolation_8_F283_DSCAL [SDT]

	Bezeichner	Typ	Initial	Kommentar
0	referencepoints	INT	8	eight reference points were
1	X_values	ARRAY [1..8] OF DINT	[8(0)]	Field 1..8 -> contains 8 x-values
2	Y_values	ARRAY [1..8] OF DINT	[8(0)]	Field 1..8 -> contains 8 y-values

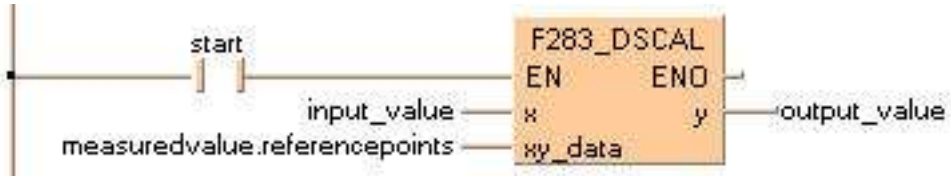
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	avtivates the function
1	VAR	input_value	DINT	0	input_value x
2	VAR	measured_value	Interpolation_8	X_values := [-5,5,15,20,30,42,45,50],Y_values := [5,-5,10,2,2(5),0,2]	number of reference points
3	VAR	output_value	DINT	0	output_value y

Hier wurde die Eingangsvariable **measured_value** von Typ des oben angelegten SDT deklariert. Die Belegung der x-Werte und y-Werte wurde bereits im POE-Kopf vorgenommen. Statt dessen können Sie im Rumpf auch die x-Werte und y-Werte verändern indem Sie einen Wert auf die Variable, z.B. **Measuredvalues.Y_Values[3]** für y3**, schreiben.**

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie berechnet für **input value** an der Stelle x den Ausgangswert y durch lineare Interpolation zwischen den benachbarten Stützstellen die in der Variablen **measured value** gespeichert sind.

KOP

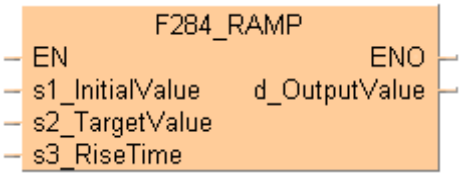


ST
```
IF start THEN
    F283_DSCAL(input_value, measured_value.referencepoints, output_value);
END_IF;
```

F284_RAMP

Ausgabe der Rampenfunktion im 16-Bit-Format

Erklärung Dieser Befehl gibt die durch die lineare Rampenfunktion ermittelten Werte auf der Basis der eingestellten Parameter aus.



SPS-Typen **Verfügbarkeit von F284_RAMP (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1_InitialValue	INT	Anfangswert, von dem der Ausgabewert an- oder absteigt, sobald die steigende Flanke erkannt worden ist.
s2_TargetValue	INT	Sollwert, bis zu dem der Ausgabewert an- oder absteigt.
s3_RiseTime	INT	Zeitspanne in ms, die der Ausgabewert benötigt, um vom Anfangs- zum Sollwert an- oder abzusteigen.
d_OutputValue	INT	Der Ausgabewert

Operanden

Für	Bitmerker				T/C		Register			Konstante
s1, s2, s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ mit Indexmodifizierern definierter Bereich größer als zulässiger Bereich.der Zeitbereich für die Ausgabe **s3_RiseTime** ist kleiner 1 oder größer 30000.
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet.

POE Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

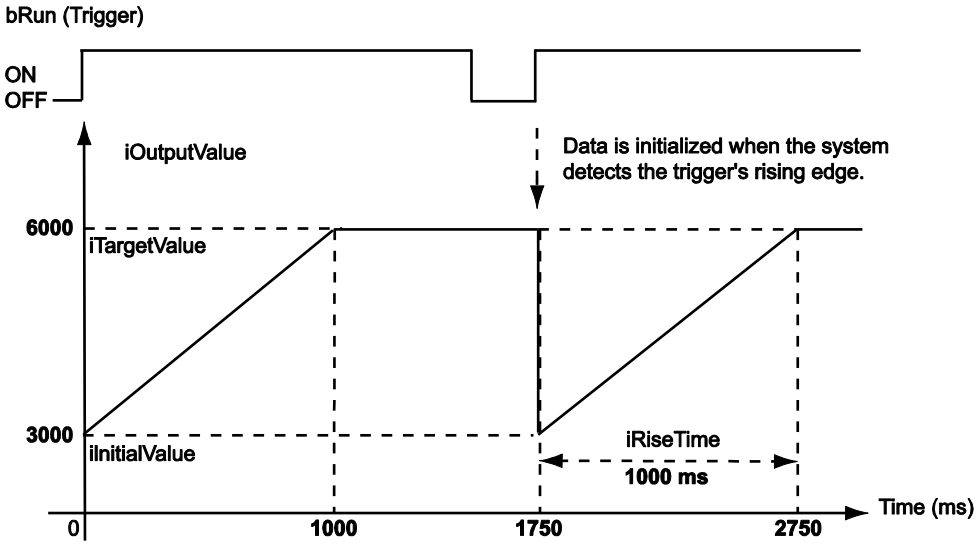
	Klasse	Bezeichner	Typ	Initial
0	VAR	iInitialValue	INT	3000
1	VAR	iTargetValue	INT	6000
2	VAR	iRiseTime	INT	1000
3	VAR	iOutputValue	INT	0
4	VAR	bRun	BOOL	FALSE

In diesem Beispiel werden die Eingangsvariablen **iInitialValue**, **iTargetValue** und **iRiseTime** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben (Freigabe-Eingang z. B. zum Testen). Ferner wurde die Variable **bRun** deklariert, um die Rampenfunktion zu starten; die Variable **iOutputValue** wurde zum Speichern der Ergebnisse deklariert.

Rumpf Wenn die Variable **bRun** auf TRUE gesetzt ist, wird die Funktion ausgeführt und **iOutputValue** steigt von 3000 (Anfangswert von **iInitialValue**) auf 6000 (Anfangswert von **iTargetValue**). Die Zeitdauer beträgt 1000ms (entsprechend dem Anfangswert von **iRiseTime**).

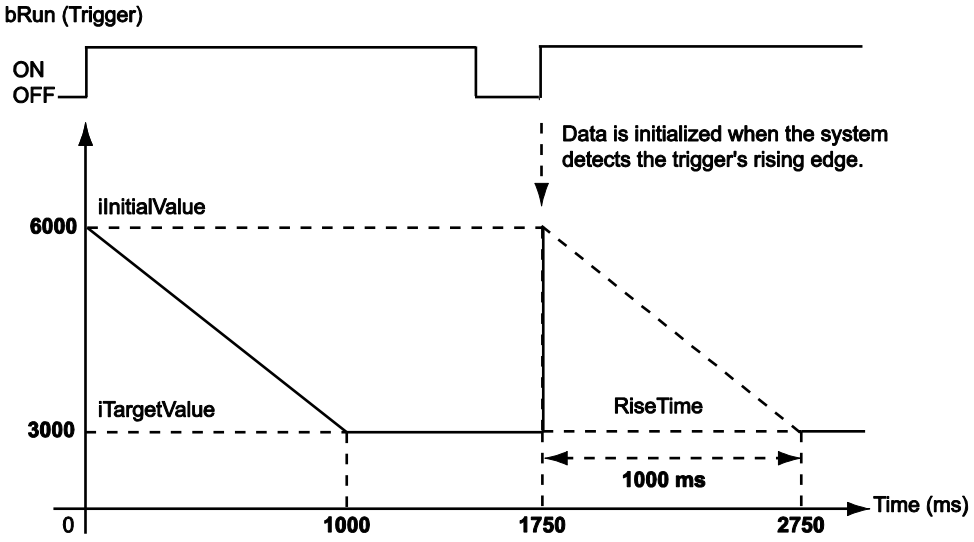
Zeitdiagramm für ansteigenden Ausgabewert:

Beispielwerte: **iInitialValue = 3000, iTargetValue = 6000, iRiseTime = 1000**

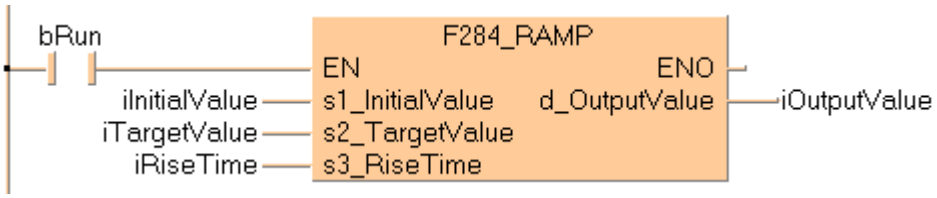


Zeitdiagramm für absteigenden Ausgabewert:

Beispielwerte: **iInitialValue = 6000, iTargetValue = 3000, iRiseTime = 1000**



LD



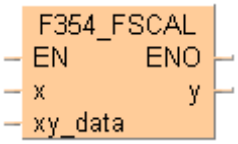
ST

```
IF bRun THEN
    F284_RAMP(iInitialValue, iTargetValue, iRiseTime, iOutputValue);
END_IF;
```

F354_FSCAL

Interpolation einer Messkurve mit REAL-Zahlen

Erklärung Die Funktion ermittelt an der Stelle x den Funktionswert y durch lineare Interpolation zwischen den benachbarten Stützstellen. Die Funktion kann REAL-Zahlen verarbeiten.



Die Stützstellen werden der Funktion durch eine Variable vom Type SDT, die die Anzahl der Stützstellen und die xy-Wertepaare enthält, übergeben.

XY_DUT [SDT]

	Bezeichner	Typ	Initial
0	Number	INT	6
1	X_values	ARRAY [1..6] OF REAL	[1.0,1.5,2.0,5.0,5.5,6.0]
2	Y_values	ARRAY [1..6] OF REAL	[1.0,1.3,2.5,10.0,11.0,9.0]

Beispiel:

	Klasse	Bezeichner	Typ	Initial
0	VAR	CalculateY	BOOL	FALSE
1	VAR	X_value	REAL	4.0
2	VAR	Y_value	REAL	0.0
3	VAR	XY_values	XY_DUT	

Weitere Informationen zu dieser Funktion enthalten die Hilfetexte zu: F282_SCAL (s. S. 458) und F283_DSCAL (s. S. 461).

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F354_FSCAL (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
x	REAL	Eingangswert (X)
xy_daten	INT	Erstes Element einer Variablen vom Typ SDT, der die xy-Wertepaare enthält
y	REAL	Ausgangswert (Y)

Operanden

Für	Merker				T/C		Register			Konstante
x	WX	WY	WR	WL	SV	EV	DT	LD	FL	real
xy_daten	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
y	-	WY	WR	WL	SV	EV	DT	LD	FL	-

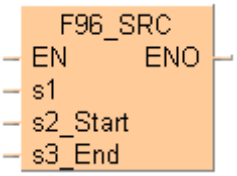
Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die mit Indexmodifizierern definierte Adresse den zulässigen Bereich überschreitet ▪ für 'x' eine nicht-reelle Zahl eingegeben wird ▪ die Zahl der Wertepaare (erstes SDT-Element) <2 oder >99 ist. ▪ für 'xy_data' ein nicht-reeller Wert (xt, yt) angegeben wird ▪ die Werte für 'xy_data' nicht in aufsteigender Reihenfolge angegeben wurden ▪ die Werte für 'xy_data' den zulässigen Bereich überschreiten ▪ es bei der Interpolation zu einem Berechnungsüberlauf kommt.
R9008	%MX0.900.8	kurzzeitig	

F96_SRC

16-Bit-Suchen

Erklärung Der 16-Bit-Wert des Speicherregisters **s1** oder eine Konstante **s1** wird im Speicherbereich mit der Anfangsadresse **s2_Start** und der Endadresse **s3_End** gesucht.



Wenn die Suche ausgeführt ist, werden die Suchergebnisse wie folgt gespeichert:

- Die Anzahl der Sucherfolge, die identisch mit **s1** ist, wird im Sonder-Datenregister DT9037 (DT90037 bei FP2/2SH, FP10/10S/10SH) abgelegt.
- Der Abstand von der Anfangsadresse des Suchbereichs s2 bis zur gefundenen Übereinstimmung mit dem Vergleichswert wird im Sonder-Datenregistern DT9038 (DT90038 bei FP2/2SH, FP10/10S/10SH) übergeben.

Es gilt: Anfangsadresse **s2_Start** ≤ Endadresse **s3_End**.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F96_SRC (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	16-Bit-Register oder Konstante zum Speichern des gesuchten Wertes
s2_Start		Anfangsadresse
s3_End		Endadresse

Die Variablen **s1, s2** und **s3** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
s2, s3	-	WY	WR	WL	SV	EV	DT	LD	FL	-

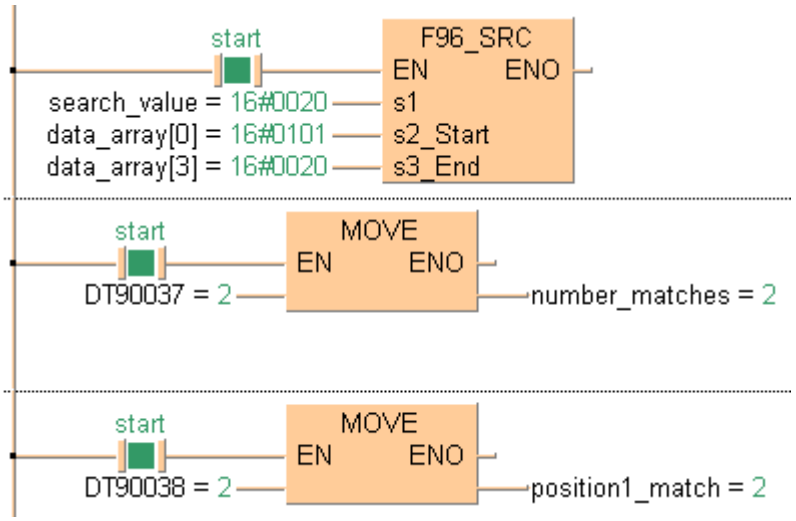
Beispiel In diesem Beispiel wird die Funktion F96_SRC im Kontaktplan (KOP) und im Strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the fuction
1	VAR	search_value	WORD	16#20	specifies the value to
2	VAR	data_array	ARRAY [0..3] OF WORD	[16#101,16#2A04,16#20,16#20]	2 matches for 16#20 data_array[2] = 1st match
3	VAR	number_matches	INT	0	
4	VAR	position1_match	INT	0	

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

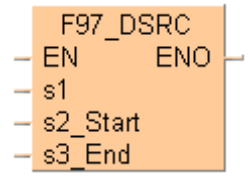


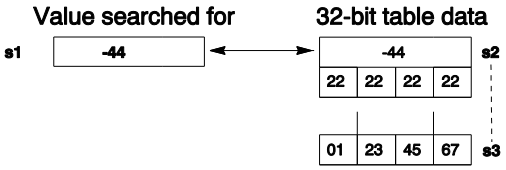
ST

```
IF start THEN
    F96_SRC( s1:= search_value ,
        s2_Start:= data_array[0] ,
        s3_End:= data_array[3] );
    number_matches:=DT90037;
    position_1match:=DT90038;
END_IF;
```

F97_DSRC 32-Bit-Suchen

Erklärung Die Funktion sucht in einem Speicherbereich, dessen Anfang und Ende dem Eingang **s2** und dem Eingang **s3** übergeben werden nach einem Wert, der am Eingang **s1** anliegt.





Die Anzahl der gefundenen Übereinstimmungen mit **s1** wird in das Sonderdatenregister DT90037 geschrieben.

Der relative Abstand vom Anfang des Suchbereichs am Eingang **s2** bis zur ersten gefundenen Übereinstimmung mit dem gesuchten Wert wird in das Sonderdatenregister DT90038 geschrieben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F97_DSRC (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	32-Bit-Register oder Konstante zum Speichern des gesuchten Wertes
s2		Anfangsadresse
s3		Endadresse

Die Adressen der Variablen an den Eingängen **s2** und **s3** müssen vom gleichen Adresstyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
s2, s3:	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Ausgängen von s2 > s3 ist.
R9008	%MX0.900.8	kurzzeitig	

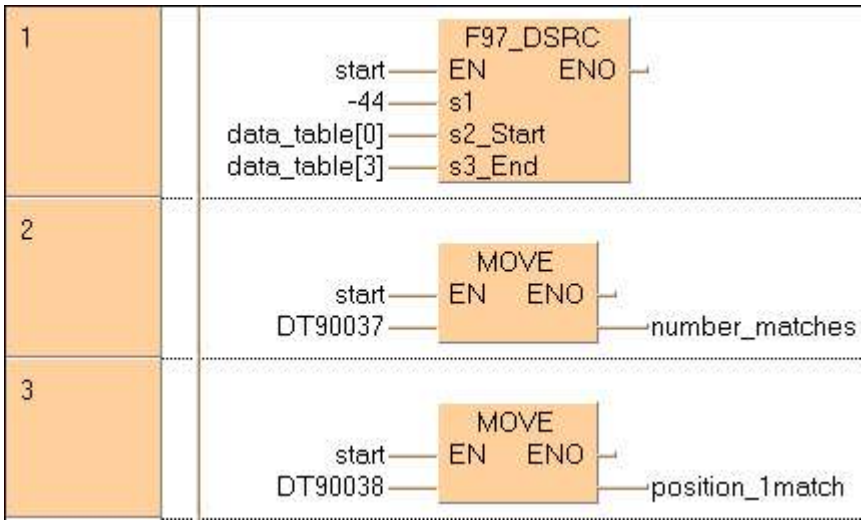
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the funktion
1	VAR	data_table	ARRAY [0..3] OF DINT	[-44,222222,-44,12345]	Arbitrarily large data field
2	VAR	number_matches	INT	0	result: here 2
3	VAR	position_1match	INT	0	result: here 0

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Statt einer Eingangsvariablen wurde in diesem Beispiel eine Konstante (-44) an den Eingang s1 der Funktion geschrieben. Das Funktionsergebnis wird in die Sonderdatenregister DT90037 und DT90038 geschrieben. Die beiden Funktionen E_MOVE kopieren das Ergebnis in zwei Variablen **number_matches** und **position_1match**.

KOP



ST

```
IF start THEN
    F97_DSRC( s1:= -44 ,
        s2_Start:= data_table[0] ,
        s3_End:= data_table[3] );
    number_matches:=DT90037;
    position1_match:=DT90038;
END_IF;
```

15.1 Einführung in die Verwendung des FIFO-Puffers

Der FIFO-Puffer (first in/first out) ist ein Ringpuffer, in dem Daten in der Reihenfolge gespeichert werden, in der sie in den Puffer geschrieben werden. Die Daten werden in der gespeicherten Reihenfolge aus dem Puffer gelesen, und zwar beginnend mit dem zuerst gespeicherten Datenelement. Der Puffer eignet sich zum Zwischenspeichern von Daten in der Reihenfolge, in der sie anfallen.

Anwendung:

- Mit dem Befehl F115_FIFT (s. S. 473) wird der Adressbereich des FIFO-Puffers definiert. (Dies sollte nur einmal erfolgen, und zwar bevor Daten aus ihm gelesen oder in ihn geschrieben werden.)
- Die Daten müssen mit dem Befehl F117_FIFW (s. S. 479) in den Puffer geschrieben und mit dem Befehl F116_FIFR (s. S. 476) aus dem Puffer gelesen werden.

Daten schreiben:

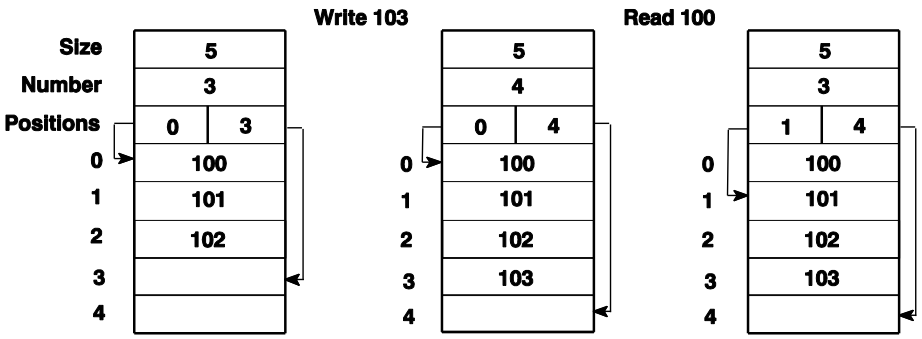
- Wenn Daten in den Puffer geschrieben werden, werden nacheinander die Datenspeicher 0 bis n-1 (n = Speichergröße des Puffers) aufgefüllt. Ein Zeiger gibt an, in welchen Datenspeicher als nächstes geschrieben wird. Die Anzahl der gespeicherten Datenelemente (Worte) wird bei jedem Schreibvorgang um 1 erhöht.
- Wenn die Datenspeicher voll sind, d. h., wenn die Anzahl der gespeicherten Worte gleich n-1 ist, können keine weiteren Daten mehr in den Puffer geschrieben werden.

Daten lesen:

- Beim Lesen von Daten, werden die zuerst gespeicherten Datenelemente als erstes aus dem Puffer gelesen. Ein Zeiger gibt an, aus welchem Datenspeicher als nächstes gelesen werden soll. Die Anzahl der gespeicherten Datenelemente (Worte) verringert sich bei jedem Lesevorgang um 1.
- Wenn versucht wird, Daten aus einem leeren Puffer zu lesen, d. h., wenn die Anzahl der gespeicherten Datenelemente gleich 0 ist, geht die Steuerung in den Fehlerzustand.

Datenspeicherbereich:

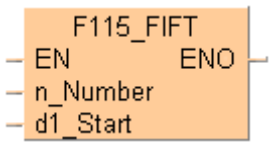
Im unten dargestellten Beispiel befindet sich der Lesezeiger an Position 0 und der Schreibzeiger an Position 3. Daten, die als nächstes in den FIFO-Puffer geschrieben werden, werden in Datenspeicher 3 gespeichert. Danach zeigt der Schreibzeiger auf Datenspeicher 4. Wenn Daten gelesen werden, werden Sie aus Datenspeicher 0 gelesen. Danach zeigt der Lesezeiger auf Datenspeicher 1. (Weitere Informationen zu den Zeigern finden Sie unter F115_FIFT (s. S. 473)).



F115_FIFT

FIFO-Pufferbereich-Definition

Erklärung Mit F/P115 definieren Sie die Anfangsadresse d1 des FIFO-Puffers (first in/first out) und die Speichergröße n des FIFO-Puffers.

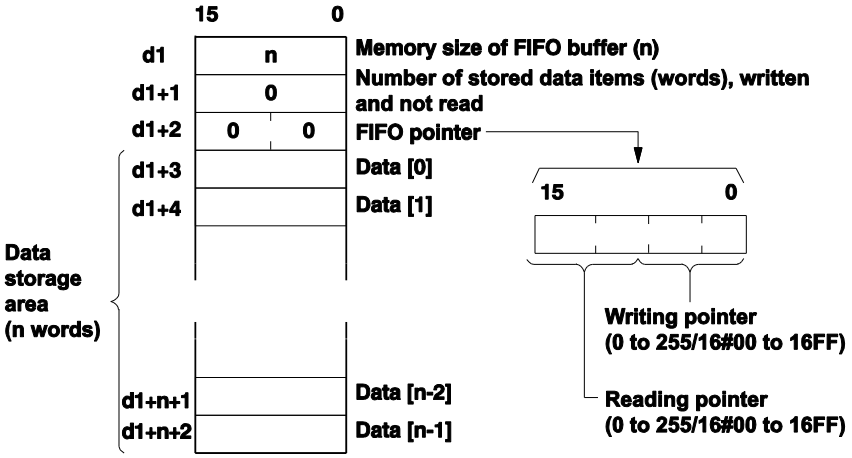


n: Speichergröße (Anzahl der Worte) des FIFO-Puffers,
n = 1 bis 256.

d1: Anfangsadresse des FIFO-Puffers

Einführung in die Verwendung des FIFO-Puffers (siehe S. 473)

Die Definition des Speicherbereichs mit dem FIFT-Befehl sollte nur einmal durchgeführt werden, und zwar bevor Daten in den FIFO-Puffer geschrieben oder aus ihm gelesen werden. Der FIFO-Puffer wird mit dem FIFT-Befehl wie folgt definiert:



Mit dem FIFT-Befehl werden folgende Standardwerte eingestellt: d1 = n (mit FIFT-Befehl definierter Wert), d1+1 = 0 und d1+2 = 16#0000.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F115_FIFT (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
n	INT	spezifiziert die Speichergröße des FIFO-Puffers
d1	ANY16	Anfangsadresse des FIFO-Puffers

Operanden

Für	Merker				T/C		Register			Konstante
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d1	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ n = 0 ▪ n > 256 ▪ mit n definierter Bereich größer als zulässiger Bereich
R9008	%MX0.900.8	kurzzeitig	

Beispiel

Hier wird der FIFO-Puffer veranschaulicht, indem die Funktionen F115_FIFT, F116_FIFR und F117_FIFW verwendet werden.

SDT

Der SDT wird im SDT-Pool definiert. Sie können ihn im POE-Kopf verwenden.

FIFO_n_WORD [SDT]

	Bezeichner	Typ	Initial
0	Size	INT	0
1	Number	INT	0
2	Positions	WORD	0
3	Data	ARRAY [0..12] OF WORD	[13(0)]

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	FIFO	FIFO_n_WORD	
1	VAR	Read_Data	INT	0
2	VAR	Write_Data	INT	1
3	VAR	FIFO_Initialize	BOOL	FALSE
4	VAR	FIFO_Write	BOOL	FALSE
5	VAR	FIFO_Read	BOOL	FALSE
6	VAR	Change_Value	BOOL	FALSE

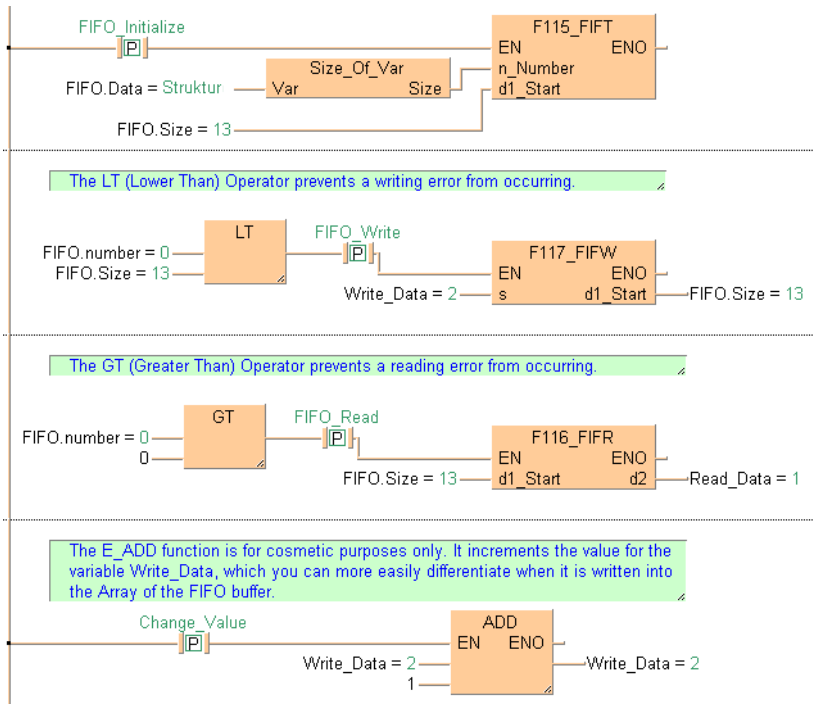
Rumpf

Das folgende Beispiel zeigt den Zustand des Puffers, nachdem **FIFO_Write** zweimal ausgeführt wurde und **FIFO_Read** einmal. Wird **FIFO_Write** das erste Mal ausgeführt, dann wird der Wert 1 in **FIFO.Data[0]** geschrieben. Wird **FIFO_Read** ausgeführt, dann liest **Read_Data** diesen Wert. Wird **FIFO_Write** zum zweiten Mal ausgeführt, dann erhöht sich der Schreibzeiger um den Wert 1 und der Wert 2 wird in **FIFO.Data[1]** geschrieben. siehe Entry Data Monitor 1

FIFO_DUT

-FIFO_DUT	Structure
-FIFO	Structure
Size	13 at DT1200
Number	0 at DT1201
Positions	16#0101 at DT1202
-Data	Structure
[0]	16#0001 at DT1203
[1]	16#0000 at DT1204
[2]	16#0000 at DT1205
[3]	16#0000 at DT1206
[4]	16#0000 at DT1207
[5]	16#0000 at DT1208
[6]	16#0000 at DT1209
[7]	16#0000 at DT1210
[8]	16#0000 at DT1211
[9]	16#0000 at DT1212
Read_Data	1 at DT1216
Write_Data	2 at DT1217
FIFO_Initialize	2#1 at R250
FIFO_Write	2#1 at R251
FIFO_Read	2#1 at R252
Change_Value	2#1 at R253

KOP

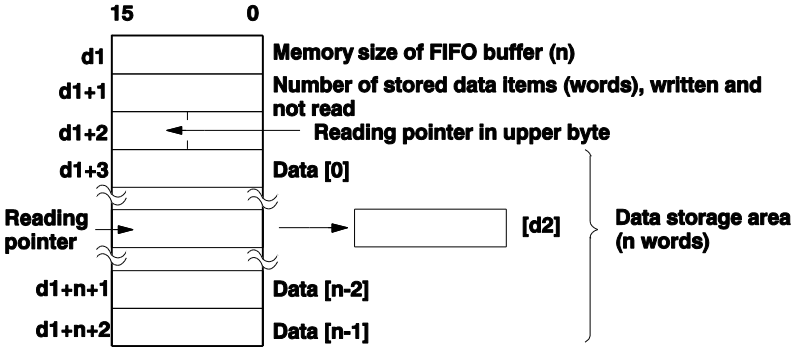


F116_FIFR Lesen aus FIFO-Puffer

Erklärung Mit F/P116 werden die Daten **d1** aus dem FIFO-Puffer (First-In-First-Out) gelesen und in dem mit **d2** definierten Bereich gespeichert.

Einführung in die Verwendung des FIFO-Puffers (siehe S. 473)

Wenn der Befehl F/P116 ausgeführt wird, werden die Daten aus der vom Lesezeiger bezeichneten Adresse gelesen.



- (0), (n–2) und (n–1) sind dem Datenspeicherbereich zugewiesene Adressen.
- n ist der mit dem Befehl F115_FIFT (s. S. 473) zugewiesene Wert.

Der Lesezeiger wird in den oberen acht Bits des dritten Wortes im FIFO-Pufferbereich gespeichert. Die tatsächliche Adresse, aus der gelesen wird, ergibt sich aus d1 plus dem Offset 3 plus dem Wert des Lesezeigers.

Beim Lesen wird die Anzahl der gespeicherten Datenelemente um 1 herabgesetzt, und der Lesezeiger wird um eine Position weitergeschoben (der Wert wird um 1 erhöht) oder auf Null gesetzt, wenn der Zeiger auf dem letzten Element stand.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F116_FIFR (s. S. 1186)**

☞

- **Wenn der Befehl F/P116_FIFR ausgeführt wird, obwohl die Anzahl der gespeicherten Datenelemente 0 ist bzw. die Werte des Lese- und des Schreibzeigers gleich sind, geht die Steuerung in den Fehlerzustand.**
- **Wenn die Werte des Lese- und des Schreibzeigers gleich sind, kann nicht gelesen werden.**
- **Wenn der Lesezeiger auf der letzten Adresse des FIFO-Puffers steht (entspricht der mit dem FIFO-Befehl definierten Speichergröße n) und der Befehl F/P116_FIFR wird ausgeführt, wird der Lesezeiger auf 0 gesetzt.**

Datentypen

Variable	Datentyp	Funktion
d1	ANY16	Anfangsadresse des FIFO-Puffers

d2		16-Bit-Bereich zum Speichern der vom FIFO-Puffer eingelesenen Daten

Die Variablen **d1** und **d2** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ mit d1 definierte Größe (n) des FIFO-Puffers ist n = 0 oder n > 256 ▪ Anzahl der gespeicherten Datenelemente im FIFO-Puffer = 0 ▪ Anzahl der gespeicherten Datenelemente im FIFO>-Puffer > n ▪ auf der Speichergröße n basierende Endadresse des FIFO-Puffers größer als zulässiger Bereich ▪ Wert des Schreibzeigers > n ▪ Wert des Lesezeigers nach dem Lesen = 256 (16#100) oder > 256
R9008	%MX0.900.8	kurzzeitig	

Beispiel

Hier wird der FIFO-Puffer veranschaulicht, indem die Funktionen F115_FIFT, F116_FIFR und F117_FIFW verwendet werden.

SDT

Der SDT wird im SDT-Pool definiert. Sie können ihn im POE-Kopf verwenden.

FIFO_n_WORD [SDT]

	Bezeichner	Typ	Initial
0	Size	INT	0
1	Number	INT	0
2	Positions	WORD	0
3	Data	ARRAY [0..12] OF WORD	[13(0)]

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

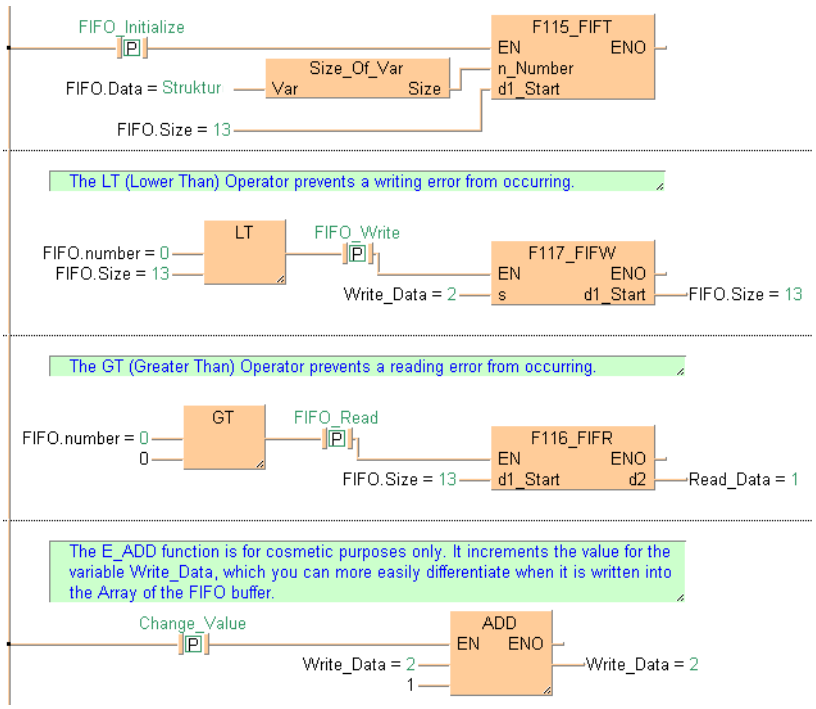
	Klasse	Bezeichner	Typ	Initial
0	VAR	FIFO	FIFO_n_WORD	
1	VAR	Read_Data	INT	0
2	VAR	Write_Data	INT	1
3	VAR	FIFO_Initialize	BOOL	FALSE
4	VAR	FIFO_Write	BOOL	FALSE
5	VAR	FIFO_Read	BOOL	FALSE
6	VAR	Change_Value	BOOL	FALSE

Rumpf Das folgende Beispiel zeigt den Zustand des Puffers, nachdem **FIFO_Write** zweimal ausgeführt wurde und **FIFO_Read** einmal. Wird **FIFO_Write** das erste Mal ausgeführt, dann wird der Wert 1 in **FIFO.Data[0]** geschrieben. Wird **FIFO_Read** ausgeführt, dann liest **Read_Data** diesen Wert. Wird **FIFO_Write** zum zweiten Mal ausgeführt, dann erhöht sich der Schreibzeiger um den Wert 1 und der Wert 2 wird in **FIFO.Data[1]** geschrieben. siehe Entry Data Monitor 1

FIFO_DUT

-FIFO_DUT	Structure
-FIFO	Structure
Size	13 at DT1200
Number	0 at DT1201
Positions	16#0101 at DT1202
-Data	Structure
[0]	16#0001 at DT1203
[1]	16#0000 at DT1204
[2]	16#0000 at DT1205
[3]	16#0000 at DT1206
[4]	16#0000 at DT1207
[5]	16#0000 at DT1208
[6]	16#0000 at DT1209
[7]	16#0000 at DT1210
[8]	16#0000 at DT1211
[9]	16#0000 at DT1212
Read_Data	1 at DT1216
Write_Data	2 at DT1217
FIFO_Initialize	2#1 at R250
FIFO_Write	2#1 at R251
FIFO_Read	2#1 at R252
Change_Value	2#1 at R253

KOP



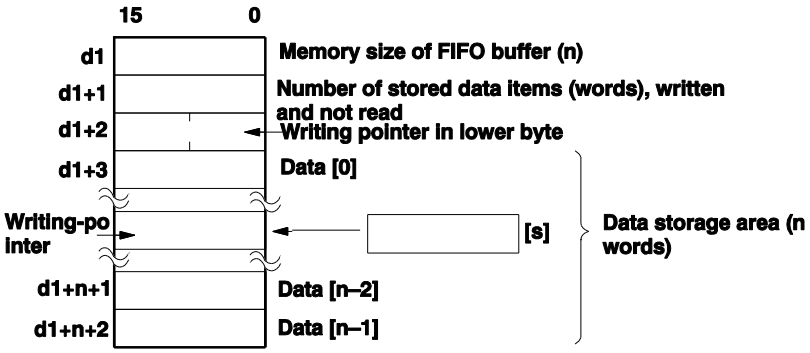
F117_FIFW

Schreiben in FIFO-Puffer

Erklärung Mit dem Befehl F/P117 werden die mit **s** festgelegten Daten in den mit **d1** definierten FIFO-Puffer geschrieben.

Einführung in die Verwendung des FIFO-Puffers (siehe S. 473)

Wenn der Befehl F/P117 ausgeführt wird, werden die Daten in den vom Schreibzeiger bezeichneten Datenspeicher geschrieben.



- (0), (n-2) und (n-1) sind dem Datenspeicherbereich zugewiesene Adressen.
- **n** ist der mit dem Befehl F115_FIFT (s. S. 473) zugewiesene Wert.

Der Schreibzeiger wird in den niederwertigen acht Bits des dritten Wortes im FIFO-Puffer gespeichert. Die tatsächliche Adresse, aus der gelesen wird, ergibt sich aus d1 plus dem Offset 3 plus dem Wert des Schreibzeigers.

Beim Schreiben wird die Anzahl der gespeicherten Datenelemente um 1 heraufgesetzt, und der Schreibzeiger wird um eine Position weitergeschoben (der Wert wird um 1 erhöht) oder auf Null gesetzt, wenn der Zeiger auf dem letzten Element stand.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F117_FIFW (s. S. 1186)**

☞

- **Wenn der Befehl F/P117_FIFW ausgeführt und der FIFO-Puffer voll ist (Anzahl der gespeicherten Datenelemente = Speichergröße n des Puffers), geht die Steuerung in den Fehlerzustand. Schreiben ist dann nicht möglich.**
- **Wenn der Schreibzeiger auf der Endadresse des FIFO-Puffers steht und der Befehl F/P117_FIFW ausgeführt wird, wird der Schreibzeiger auf 0 gesetzt.**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	16-Bit-Speicherbereich oder äquivalente Kostante um im FIFO-Puffer zu schreiben
d1		Anfangsadresse des FIFO-Puffers

Die Variablen **s** und **d1** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d1	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermer-ker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ mit **d1** definierte Größe (n) des FIFO-Puffers ist **n** = 0 oder **n** > 256 ▪ Anzahl der gespeicherten Datenelemente im FIFO-Puffer = 0 ▪ Anzahl der gespeicherten Datenelemente im FIFO > -Puffer > **n** ▪ auf der Speichergröße **n** basierende Endadresse des FIFO-Puffers größer als zulässiger Bereich ▪ Wert des Schreibzeigers > **n** ▪ Wert des Schreibzeigers nach dem Schreiben = 256 (16#100) oder > 256
R9008	%MX0.900.8	kurzzeitig	

Beispiel

Hier wird der FIFO-Puffer veranschaulicht, indem die Funktionen F115_FIFT, F116_FIFR und F117_FIFW verwendet werden.

SDT

Der SDT wird im SDT-Pool definiert. Sie können ihn im POE-Kopf verwenden.

FIFO_n_WORD [SDT]

	Bezeichner	Typ	Initial
0	Size	INT	0
1	Number	INT	0
2	Positions	WORD	0
3	Data	ARRAY [0..12] OF WORD	[13(0)]

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

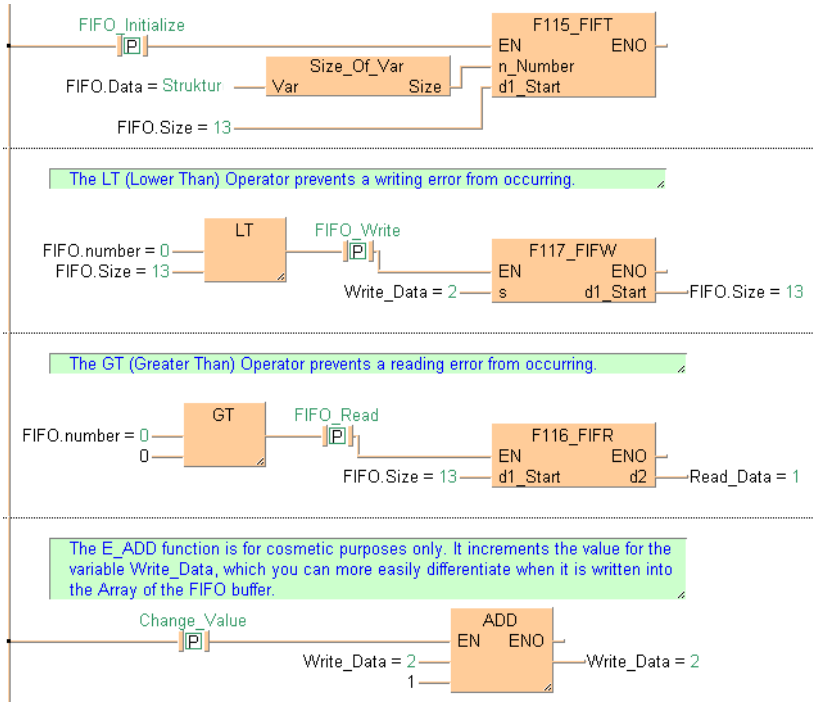
	Klasse	Bezeichner	Typ	Initial
0	VAR	FIFO	FIFO_n_WORD	
1	VAR	Read_Data	INT	0
2	VAR	Write_Data	INT	1
3	VAR	FIFO_Initialize	BOOL	FALSE
4	VAR	FIFO_Write	BOOL	FALSE
5	VAR	FIFO_Read	BOOL	FALSE
6	VAR	Change_Value	BOOL	FALSE

Rumpf Das folgende Beispiel zeigt den Zustand des Puffers, nachdem **FIFO_Write** zweimal ausgeführt wurde und **FIFO_Read** einmal. Wird **FIFO_Write** das erste Mal ausgeführt, dann wird der Wert 1 in **FIFO.Data[0]** geschrieben. Wird **FIFO_Read** ausgeführt, dann liest **Read_Data** diesen Wert. Wird **FIFO_Write** zum zweiten Mal ausgeführt, dann erhöht sich der Schreibzeiger um den Wert 1 und der Wert 2 wird in **FIFO.Data[1]** geschrieben. siehe Entry Data Monitor 1

FIFO_DUT

-FIFO_DUT	Structure
-FIFO	Structure
Size	13 at DT1200
Number	0 at DT1201
Positions	16#0101 at DT1202
-Data	Structure
[0]	16#0001 at DT1203
[1]	16#0000 at DT1204
[2]	16#0000 at DT1205
[3]	16#0000 at DT1206
[4]	16#0000 at DT1207
[5]	16#0000 at DT1208
[6]	16#0000 at DT1209
[7]	16#0000 at DT1210
[8]	16#0000 at DT1211
[9]	16#0000 at DT1212
Read_Data	1 at DT1216
Write_Data	2 at DT1217
FIFO_Initialize	2#1 at R250
FIFO_Write	2#1 at R251
FIFO_Read	2#1 at R252
Change_Value	2#1 at R253

KOP

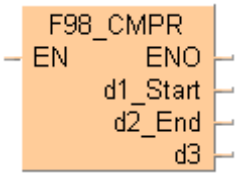


F98_CMPR

Arraydaten-Herausschieben und -Komprimieren

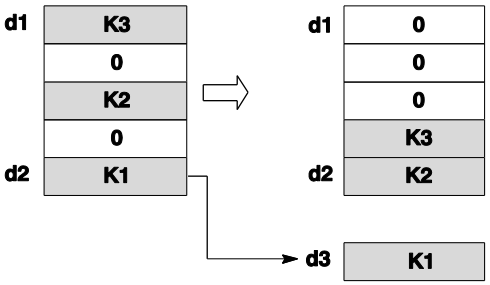
Erklärung

Der Wert ungleich 0 mit der höchsten Adresse innerhalb eines Arrays wird in die angegebene Variable geschoben, und die Werte im Array werden zur höheren Adresse hin komprimiert. Dabei werden alle Werte, die 0 sind, in den unteren Adreßbereich des Puffers gestellt und alle Werte ungleich 0 rücken auf in den oberen Bereich. Die Werte in dem durch **d1_Start** und **d2_End** definierten Array werden wie folgt verschoben:



- Der Inhalt von **d2_End** (höchste Adresse) wird in die mit **d3** definierte Variable geschoben.

Werte ungleich 0 werden innerhalb des definierten Bereichs in Richtung der höheren Adresse verschoben (komprimiert).



- Die Startadresse **d1_Start** und die Endadresse **d2_End** müssen im gleichen Speicherbereich liegen.
- **d1_Start** und **d2_End** müssen mit "**d1** ≤ **d2**" definiert werden.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F98_CMPR (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
d1_Start	ANY16	Startadresse (niedrigste) der zu komprimierenden Daten
d2_End		Endadresse (höchste) der zu komprimierenden Daten. Daten werden aus **d2_End** herausgeschoben.
d3		Empfängt Daten, die aus **d2_End** herausgeschoben werden.

Operanden

Für	Merker				T/C		Register			Konstante
d1, d2, d3	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermer-ker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ d1 > d2
R9008	%MX0.900.8	kurzzeitig	▪ d1 und d2 nicht im gleichen Speicherbereich

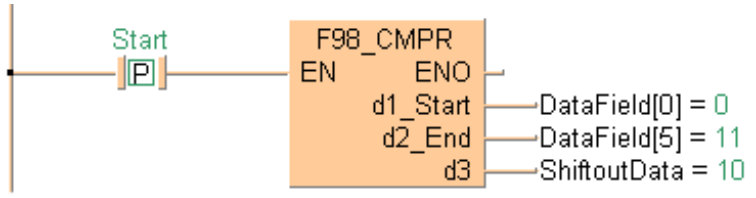
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Start	BOOL	FALSE
1	VAR	DataField	ARRAY [0..5] OF INT	[555,444,0,11,0,10]
2	VAR	ShiftoutData	INT	0

Rumpf Wenn die Variable **Start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Werte in den niedrigeren Adressen werden zur höheren Adresse hin komprimiert, und der Wert in der höchsten Adresse, hier 10, wird heraus geschoben.

KOP



-F98_CMPR_LD	Struktur
Start	2#1 at R262
-DataField	Struktur
[0]	0 at DT1211
[1]	0 at DT1212
[2]	0 at DT1213
[3]	555 at DT1214
[4]	444 at DT1215
[5]	11 at DT1216
ShiftoutData	10 at DT1217

Beispiel 2 Die Funktion F98_CMPR kann zusammen mit F99_CMPW/P99_CMPW verwendet werden, um einen FIFO-Puffer für Werte ungleich 0 einzurichten. (Verwenden Sie für Werte ungleich Null einen FIFO-Puffer).

1. Ausführung des Befehls F99_CMPW/P99_CMPW

Datenelemente, die in den Puffer geschrieben werden, werden in der Reihenfolge, in der sie in den Puffer geschrieben werden, im Puffer abgelegt. Die zuerst eingelesenen Daten befinden sich dabei in den höheren Adressen, die zuletzt eingelesenen in den niedrigeren Adressen des Puffers.

2. Ausführung des Befehls F98_CMPR/P98_CMPR

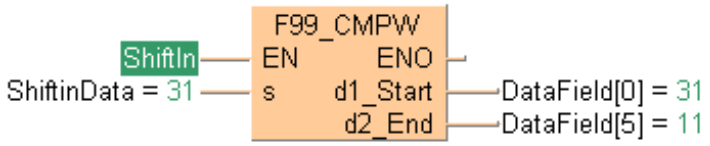
Nach dem Lesen der Daten in der höchsten Adresse des Puffers, werden die restlichen Daten im Puffer in Richtung der höheren Adresse verschoben, so dass sich die zuerst eingelesenen Daten dann in den höheren, die zuletzt eingelesenen in den niedrigeren Adressen befinden.

Die restlichen Daten im Puffer werden in Richtung der Anfangsadresse verschoben, so dass sich die ältesten Daten dann in der Regel in der Endadresse des Puffers befinden.

POU header

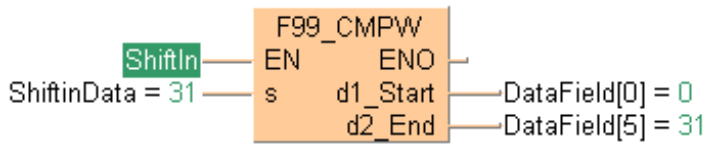
	Klasse	Bezeichner	Typ	Initial
0	VAR	DataField	ARRAY [0..5] OF INT	[0,44,0,555,0,11]
1	VAR	ShiftinData	INT	31
2	VAR	ShiftoutData	INT	0
3	VAR	ShiftIn	BOOL	FALSE
4	VAR	ShiftOut	BOOL	FALSE

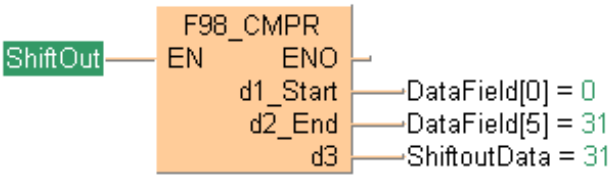
Body In Schritt 1 wird die Funktion F99 aktiviert. Der Wert der Variablen **s ShiftinData** , hier 31, wird in den Puffer hinein geschoben. Die Daten im Puffer werden komprimiert.



-F98_CMPR_Ex2_LD	Struktur
-DataField	Struktur
[0]	31 at DT1218
[1]	31 at DT1219
[2]	31 at DT1220
[3]	44 at DT1221
[4]	555 at DT1222
[5]	11 at DT1223
ShiftinData	31 at DT1224
ShiftoutData	0 at DT1225
ShiftIn	2#1 at R263
ShiftOut	2#0 at R264

In Schritt 2 wird die Funktion 98 aktiviert, und der in Variable **d3** definierte Wert, hier 11, wird herausgeschoben.



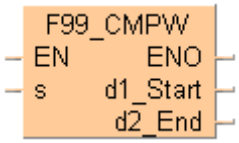


-F98_CMPR_Ex2_LD	Struktur
-DataField	Struktur
[0]	0 at DT1218
[1]	31 at DT1219
[2]	31 at DT1220
[3]	31 at DT1221
[4]	31 at DT1222
[5]	31 at DT1223
ShiftinData	31 at DT1224
ShiftoutData	31 at DT1225
ShiftIn	2#1 at R263
ShiftOut	2#1 at R264

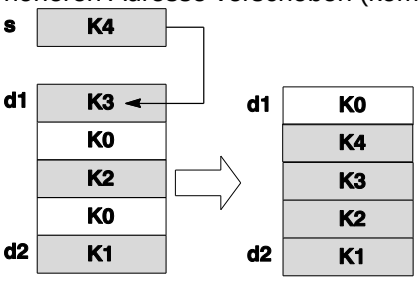
F99_CMPW

Arraydaten-Herausschieben und -Komprimieren

Erklärung Mit dieser Funktion werden neue Werte in die niedrigste Adresse des angegebenen Arrays geschoben. Die Werte im Array werden zur höheren Adresse hin komprimiert. Dabei werden alle Werte, die 0 sind, in den unteren Adreßbereich des Puffers gestellt und alle Werte ungleich 0 rücken auf in den oberen Bereich. Die Werte in dem durch **d1_Start** und **d2_End** definierten Array werden wie folgt verschoben:



- Die durch **s** definierten Werte werden in die mit **d1_Start** definierte Anfangsadresse geschoben.
- Werte ungleich 0 werden innerhalb des definierten Bereichs in Richtung der höheren Adresse verschoben (komprimiert).



- Die Startadresse **d1_Start** und die Endadresse **d2_End** müssen im gleichen Speicherbereich liegen.
- **d1_Start** und **d2_End** müssen mit "**d1** ≤ **d2**" definiert werden.
- Ist **s** = 0, findet nur eine Datenkomprimierung statt.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

☞ **Beispiel 2 zu F/P98 zeigt Ihnen, wie Sie mit den Funktionen F/P99 und F/P98 einen FIFO-Puffer einrichten können.**

SPS-Typen **Verfügbarkeit von F99_CMPW (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	Hereinzuschiebende Daten
d1_Start		Anfangsadresse des komprimierten Bereichs in den Daten aus **s** geschoben werden
d2_End		Endadresse des komprimierten Bereichs

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ d1 > d2
R9008	%MX0.900.8	kurzzeitig	▪ d1 und d2 nicht im gleichen Speicherbereich

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

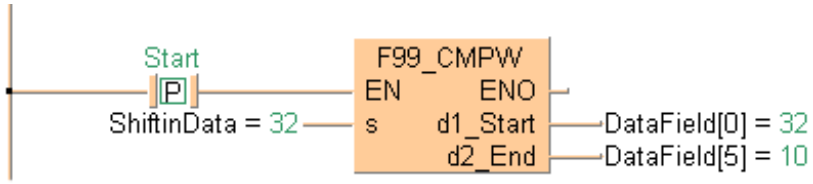
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Start	BOOL	FALSE
1	VAR	DataField	ARRAY [0..5] OF INT	[555,444,0,11,0,10]
2	VAR	ShiftinData	INT	32

Rumpf

Nachdem die Variable **Start** auf TRUE gesetzt wurde, wird der Wert der Variablen **ShiftinData**, hier 32, am Eingang s in die Anfangsadresse des Arrays geschoben, und die Daten im Array werden komprimiert.

KOP

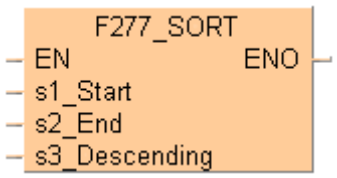


-F99_CMPW_LD	Struktur
Start	2#1 at R261
-DataField	Struktur
[0]	32 at DT1205
[1]	32 at DT1206
[2]	32 at DT1207
[3]	444 at DT1208
[4]	11 at DT1209
[5]	10 at DT1210
ShiftinData	32 at DT1211

F277_SORT

Sortieren von Werten einer 16-Bit-Datentabelle

Erklärung Die Funktion sortiert Werte (mit Vorzeichen) einer Datentabelle in auf- oder absteigender Reihenfolge.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Die Sortierreihenfolge legen Sie mit dem Wert am Eingang **s3** fest.

Am Eingang **s3** können Sie folgende Werte eingeben:

0: aufsteigende Sortierreihenfolge, d.h. mit dem kleinsten Wert beginnend

1: absteigende Sortierreihenfolge, d.h. mit dem größten Wert beginnend

Die Daten werden mit dem am Eingang **s1** liegenden Wert beginnend in der festgelegten Reihenfolge durch Vertauschen (bubble sort) sortiert. Da die Zahl der erforderlichen Wortvergleiche der quadratischen Wortanzahl entspricht, kann das Sortieren bei vielen Wörtern einige Zeit dauern. Wenn die Adresse der Variablen an den Eingängen von **s1 = s2** ist, wird nicht sortiert.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F277_SORT (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	INT	Anfang der zu sotierrenden Datentabelle
s2	INT	Ende der zu sotierenden Datentabelle
s3	INT	Spezifiziert Sortierreihenfolge: 0 = steigend, 1 = fallend

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	-	WY	WR	WL	SV	EV	DT	LD	FL	-
s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereiche s1 und s2 verschieden sind. ▪
R9008	%MX0.900.8	kurzzeitig	

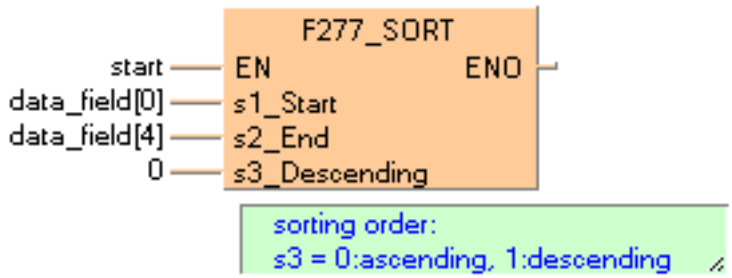
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF INT	[2,3,6,-3,1]	Arbitrarily large data field result: here [-3,1,2,3,6]
2	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Am Eingang s3 steht die Konstante 0, die die Sortierreihenfolge in aufsteigender Richtung festlegt. Statt dessen können Sie auch eine Variable im POE-Kopf deklarieren und im Rumpf an den Eingang s3 der Funktion schreiben.

KOP



ST
```
IF start THEN
    F277_SORT( s1_Start:= data_field[0],
               s2_End:= data_field[4],
               s3_Descending:= 0);
END_IF;
```

F278_DSORT

Sortieren von Werten einer 32-Bit-Datentabelle

Erklärung Die Funktion sortiert Werte (mit Vorzeichen) einer Datentabelle in auf- oder absteigender Reihenfolge.

Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Die Sortierreihenfolge legen Sie mit dem Wert am Eingang **s3** fest.

Am Eingang **s3** können Sie folgende Werte eingeben:

0: aufsteigende Sortierreihenfolge, d.h. mit dem kleinsten Wert beginnend

1: absteigende Sortierreihenfolge, d.h. mit dem größten Wert beginnend

Die Daten werden mit dem am Eingang **s1** liegenden Wert beginnend in der festgelegten Reihenfolge durch Vertauschen (bubble sort) sortiert. Da die Zahl der erforderlichen Wortvergleiche der quadratischen Wortanzahl entspricht, kann das Sortieren bei vielen Wörtern einige Zeit dauern. Wenn die Adresse der Variablen an den Eingängen von **s1 = s2** ist, wird nicht sortiert.

☞ **Die Schrittanzahl dieses 32-Bit-Befehls entspricht der des 16-Bit-Befehls.**

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F278_DSORT (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	DINT	Anfang der zu sotierrenden Datentabelle
s2	DINT	Ende der zu sotierenden Datentabelle
s3	INT	Spezifiziert Sortierreihenfolge: 0 = steigend, 1 = fallend

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereiche s1 und s2 verschieden sind. ▪
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

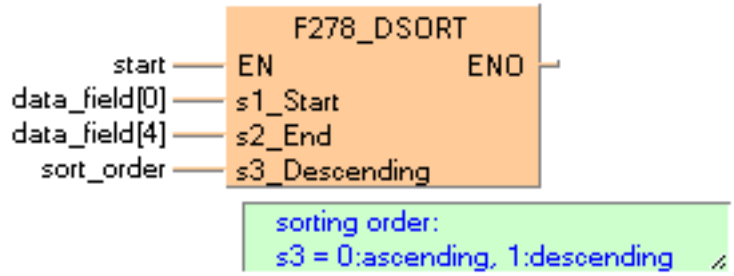
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF DINT	[2,3,222222,-333333,1]	Arbitrarily large data field
2	VAR	sort_order	INT	1	0:ascending, 1:descending

Hier wurde die Eingangsvariable **sort_order** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben (Freigabe-Eingang z.B. zum Testen).

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Da die Variable **sort_order** auf den Wert 1 gesetzt ist, wird das anliegende Datenfeld absteigend sortiert.

KOP

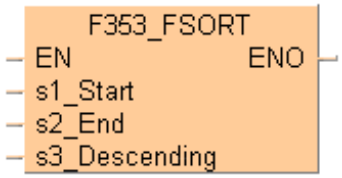


ST
```
IF start THEN
    F278_DSORT( s1_Start := data_field[0],
        s2_End := data_field[4],
        s3_Descending := sort_order);
END_IF;
```

F353_FSORT

Sortieren von Werten einer Fließkomma-Datentabelle

Erklärung Die Funktion sortiert Werte (mit Vorzeichen) einer Datentabelle in auf- oder absteigender Reihenfolge.



Der Anfang der Datentabelle wird dem Eingang **s1** und das Ende dem Eingang **s2** übergeben. Die Sortierreihenfolge legen Sie mit dem Wert am Eingang **s3** fest.

Am Eingang **s3** können Sie folgende Werte eingeben:

0: aufsteigende Sortierreihenfolge, d.h. mit dem kleinsten Wert beginnend

1: absteigende Sortierreihenfolge, d.h. mit dem größten Wert beginnend

Die Daten werden mit dem am Eingang **s1** liegenden Wert beginnend in der festgelegten Reihenfolge durch Vertauschen (bubble sort) sortiert. Da die Zahl der erforderlichen Wortvergleiche der quadratischen Wortanzahl entspricht, kann das Sortieren bei vielen Wörtern einige Zeit dauern. Wenn der Wert am Eingang **s1 = s2** ist, wird nicht sortiert.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F353_FSORT (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	REAL	Anfang der zu sotierrenden Datentabelle
s2	REAL	Ende der zu sotierenden Datentabelle
s3	INT	Spezifiziert Sortierreihenfolge: 0 = steigend, 1 = fallend

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Eingängen von s1 > s2 ist. ▪ die Adressbereich verschieden sind. ▪ die Fließkommawerte ihren möglichen Bereich überschreiten.
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

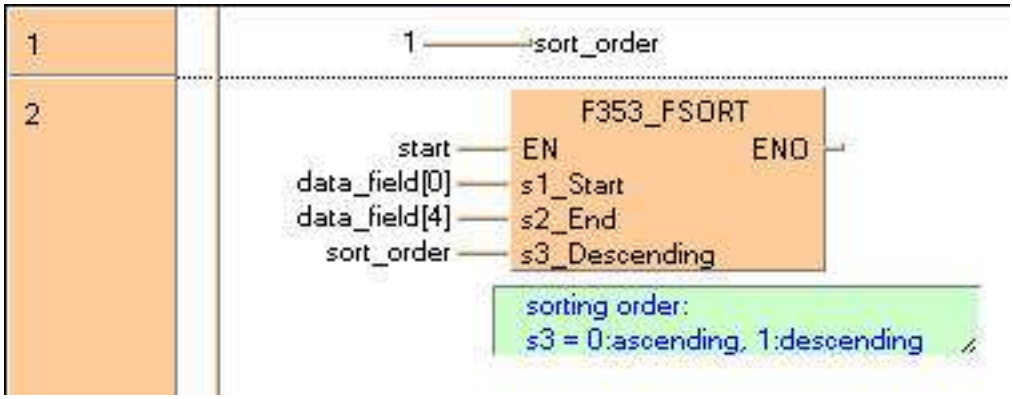
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..4] OF REAL	[2.0,3.45,-6.91,5.44,1.3]	Arbitrarily large data field
2	VAR	sort_order	INT	0	0:ascending, 1:descending

Hier wurde die Eingangsvariable **sort_order** deklariert. Statt dessen können Sie im Rumpf auch eine Konstante (z.B.1 für absteigende Sortierreihenfolge) direkt an den Eingang der Funktion schreiben.

Rumpf Der Wert 1 wird der Variablen **sort_order** zugeordnet. Wenn die Variable **start** auf den Wert TRUE gesetzt ist, wird die Funktion ausgeführt. Sie sortiert die Elemente von dem ARRAY **data_field** in absteigender Reihenfolge.

KOP



ST

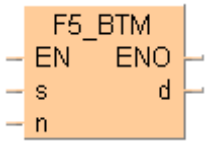
```
sort_order:=1;
IF start THEN
     F353_FSORT( s1_Start:= data_field[0],
          s2_End:= data_field[4],
          s3_Descending:= sort_order);
END_IF;
```

Kapitel 16

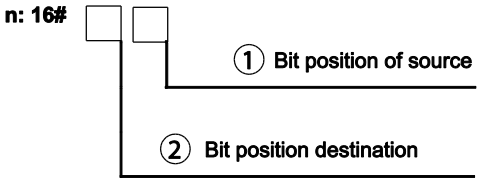
Bitweise Boolsche Befehle

F5_BTM Bit-Transfer

Erklärung Ein Bit des Speicherregisters **s** oder ein Bit der Konstanten **s** wird gemäß dem in Speicherregister **n** festgelegten Inhalt in das Speicherregister **d** transferiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Wenn eine äquivalente 16-Bit-Konstante in **s** angegeben ist, wird der Bit-Transfer intern ausgeführt, der die Konstante in einen 16-Bit-Ausdruck konvertiert.



Die Bits des Speicherregisters **n** oder Konstanten **n** geben binär codiert die Position des Quellbits an, das transferiert werden soll:



- Bit Nr. 0 bis 3: Quell-Bit Nr. (16#0 bis 16#F)
- Bit 4 bis 7: **FP2/2SH und 10SH:** Anzahl der Bits, die transferiert werden (16#0 bis 16#F)
  FP3: ungültig
- Bit Nr. 8 bis 11: Ziel-Bit Nr. (16#0 bis 16#F)
- Bit Nr. 12 bis 15: ungültig

16#C01 bewirkt zum Beispiel von rechts nach links gelesen: Ein Bit wird von Position 1 nach Position 12 (16#C) transferiert.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F5_BTM (s. S. 1191)**

Datentypen

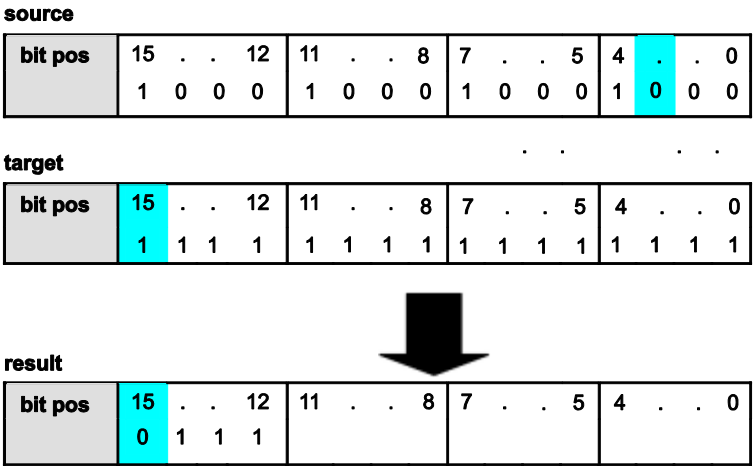
Variable	Datentyp	Funktion
s	ANY16	16-Bit-Quellregister
n		legt Anfangs- und Zielposition des Bits fest
d		16-Bit-Zielregister

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Erklärung mit dem Beispielwert 16#8888 und einem Bit an Position 2, die zum Zielwert an Bitposition 15 transferiert werden.



Das Bit an Position 15 wird ausgetauscht, der Zielwert für dieses Beispiel lautet: 16#7FFF

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

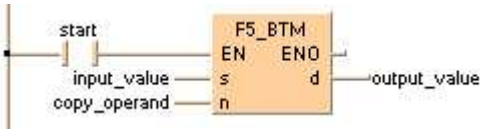
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	WORD	2#1000100010001000	
2	VAR	copy_operand	WORD	16#0F02	digit no.1 and no.3 are invalid, digit no.0 locates
3	VAR	output_value	WORD	2#1111111111111111	result after a 0->1 leading edge from start: 2#0111111111111111
4	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



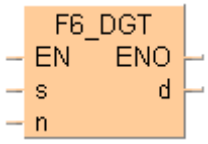
ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

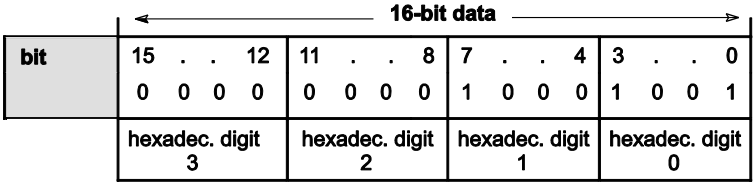
```
IF start THEN
    F5_BTM( s:= input_value,
            n:= copy_operand,
            d=> output_value);
END_IF;
```

F6_DGT Digit-Transfer

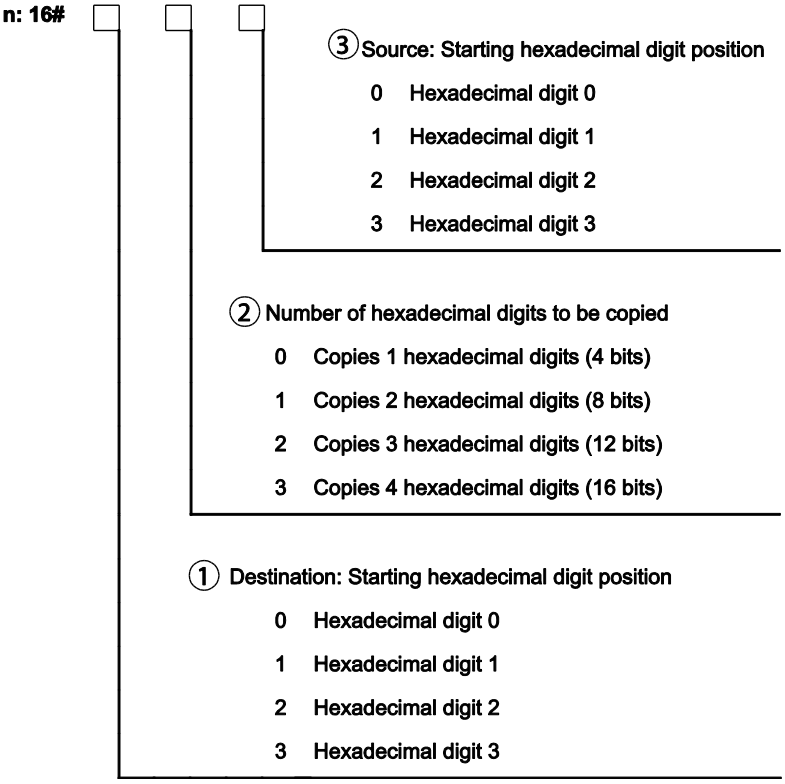
Erklärung Die Nibbles des Speicherregisters oder der Speicherkonstante **s** werden, wie im Speicherregister **n** festgelegt, in das Speicherregister **d** transferiert.



Nibbles sind Einheiten von 4 Bits, die zum Datentransfer verwendet werden. Mit diesem Befehl werden 16-Bit-Daten in 4 Nibbles aufgeteilt. Die Nibbles werden der Reihe nach Hexadezimal-Nibble 0, Nibble 1, Nibble 2 und Nibble 3 benannt, angefangen von den vier niedrigsten signifikanten Bits:

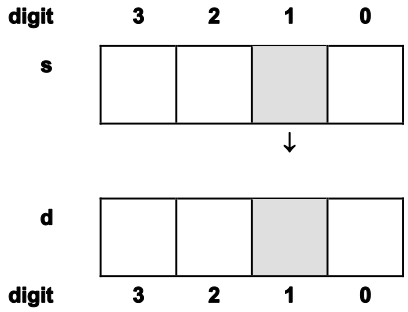


n legt die ③ Ausgangsposition der Hexadezimal-Nibbles, die ② Anzahl der Nibbles und die ① Zielposition der Hexadezimal-Nibbles, die kopiert werden sollen, wie folgt fest:

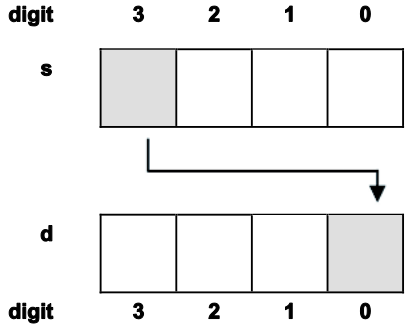


Nachstehend sind einige Muster des Digit-Transfers auf der Basis der Spezifikation von n dargestellt.

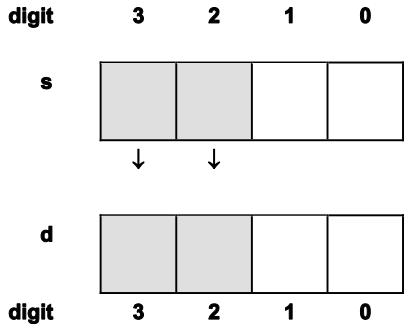
- Das Hexadezimal-Nibble 1 der Quelle wird in das Hexadezimal-Nibble des Ziels kopiert:



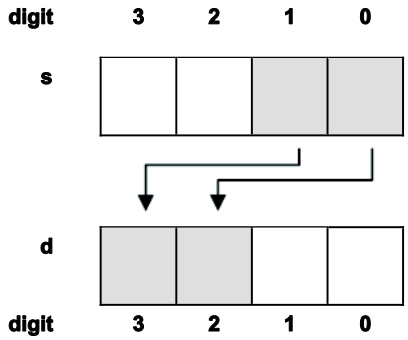
- Das Hexadezimal-Nibble 3 der Quelle wird in das Hexadezimal-Nibble des Ziels kopiert:



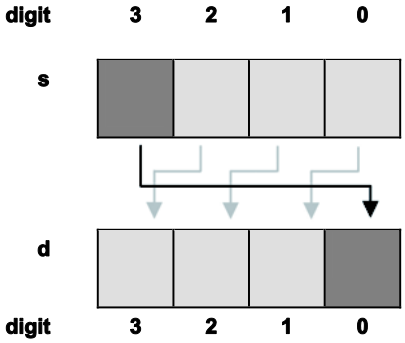
- Wenn mehrere Hexadezimal-Nibbles (2 und 3) der Quelle in mehrere Hexadezimal-Nibbles (2 und 3) des Ziels kopiert werden:



- Wenn mehrere Hexadezimal-Nibbles (0 und 1) der Quelle in mehrere Hexadezimal-Nibbles (2 und 3) des Ziels kopiert werden:



- Wenn 4 Hexadezimal-Nibbles (0 bis 3) der Quelle in 4 Hexadezimal-Nibbles (0 bis 3) des Ziels kopiert werden:



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F6_DGT (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	16-Bit-Quellregister
n		Spezifiziert die Nibble Quell- und die Zielposition sowie die Anzahl der Nibbles.
d		16-Bit-Zielregister

Operanden

Für	Merker				T/C		Register			Konstante
s, n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

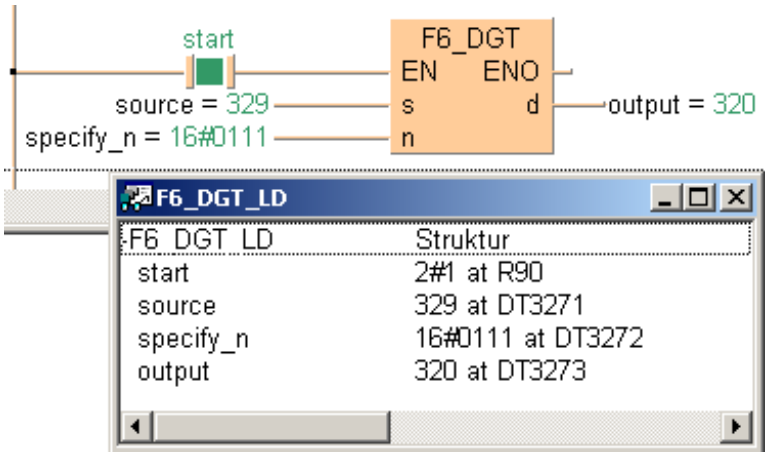
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	
1	VAR	source	INT	329	decimal 329 = 16#149
2	VAR	specify_n	WORD	16#111	Beginning from the end:
3	VAR	output	INT	0	1: first hex. digit is digit 1, i.e. 4
4	VAR				1: copies 2 hex. digits, i.e. 14
					1: destination is hex. digit 1

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Werte für **source** und **output** in der Kopfzeile des Kontaktplanrumpfes müssen eingestellt werden, damit der Hexadezimalwert durch Aktivieren des Hex-Symbols in der Werkzeugleiste angezeigt werden kann.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

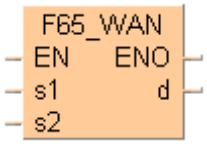
```
IF start THEN
    F6_DGT( s:= source,
            n:= specify_n,
            d=> output);
END_IF;
```

F65_WAN

16-Bit-UND-Verknüpfung

Erklärung

Der 16-Bit-Wert des Speicherregisters oder eine Konstante **s1** und der 16-Bit-Wert des Speicherregisters oder eine Konstante **s2** werden bitweise UND-verknüpft, wenn der **EN**-Eingang auf TRUE gesetzt ist. Das Ergebnis der UND-Operation wird im Speicherregister **d** abgelegt. Wenn eine äquivalente 16-Bit-Konstante in **s1** oder **s2** angegeben ist, wird die UND-Operation intern ausgeführt und eine Konvertierung in einen binären 16-Bit-Ausdruck durchgeführt. Mit diesem Befehl lassen sich bestimmte Bits der 16-Bit-Daten auf FALSE setzen.



Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0
s1	0 1 0 0	1 1 0 1	1 0 1 1	1 0 0 1

Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0
s2	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1

start: ON

Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0
d	0 0 0 0	0 0 0 0	1 0 1 1	1 0 0 1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F65_WAN (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Konstante oder 16-Bit-Bereich
s2		Konstante oder 16-Bit-Bereich
d		16-Bit-Speicherbereich für Operationsergebnis AND

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel In diesem Beispiel wird die Funktion F65_WAN im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value_1	WORD	2#0000000011001100	
2	VAR	value_2	WORD	2#0000000010101010	
3	VAR	output_value	WORD	0	result after a 0->1 leading edge from start: 2#0000000010001000
4	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

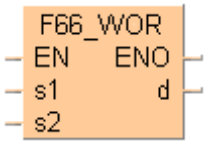
```
IF start THEN
    F65_WAN(value_1, value_2, output_value);
END_IF;
```

F66_WOR

16-Bit-ODER-Verknüpfung

Erklärung

Der 16-Bit-Wert des Speicherregisters oder eine Konstante **s1** und der 16-Bit-Wert des Speicherregisters oder eine Konstante **s2** werden bitweise ODER-verknüpft, wenn der **EN**-Eingang auf TRUE gesetzt ist. Das Ergebnis der ODER-Operation wird im Speicherregister **d** abgelegt. Wenn eine äquivalente 16-Bit-Konstante in **s1** oder **s2** angegeben ist, wird die UND-Operation intern ausgeführt und eine Konvertierung in einen binären 16-Bit-Ausdruck durchgeführt. Mit diesem Befehl lassen sich bestimmte Bits der 16-Bit-Daten auf TRUE setzen.



Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0
s1	0 1 0 0	1 1 0 1	1 0 1 1	1 0 0 1

Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0
s2	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1

start: ON

Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0
d	0 1 0 0	1 1 0 1	1 1 1 1	1 1 1 1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F66_WOR (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Konstante oder 16-Bit-Bereich
s2		Konstante oder 16-Bit-Bereich
d		16-Bit-Speicherbereich für Operationsergebnis OR

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value_1	WORD	2#0000000011001100	
2	VAR	value_2	WORD	2#0000000010101010	
3	VAR	output_value	WORD	0	result after a 0->1 leading edge from start: 2#0000000011101110
4	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

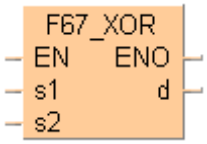
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F66_WOR(value_1, value_2, output_value);
END_IF;
```

F67_XOR

16-Bit-EXCLUSIV-ODER-Verknüpfung

Erklärung Der 16-Bit-Wert des Speicherregisters oder eine Konstante **s1** und der 16-Bit-Wert des Speicherregisters oder eine Konstante **s2** werden bitweise exklusiv ODER-verknüpft, wenn der **EN**-Eingang auf TRUE gesetzt ist. Das Ergebnis der ODER-Operation wird im Speicherregister **d** abgelegt. Wenn eine äquivalente 16-Bit-Konstante in **s1** oder **s2** angegeben ist, wird die exklusive ODER-Operation intern ausgeführt und eine Konvertierung in einen binären 16-Bit-Ausdruck durchgeführt. Mit diesem Befehl können Sie die Anzahl identischer Bits in den beiden 16-Bit-Daten prüfen.



Bit position	15	·	·	12	11	·	·	8	7	·	·	4	3	·	·	0
s1	0	1	0	0	1	1	0	1	1	0	1	1	1	0	0	1

Bit position	15	·	·	12	11	·	·	8	7	·	·	4	3	·	·	0
s2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

start: ON

Bit position	15	·	·	12	11	·	·	8	7	·	·	4	3	·	·	0
d	0	1	0	0	1	1	0	1	0	1	0	0	0	1	1	0

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F67_XOR (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Konstante oder 16-Bit-Bereich
s2		Konstante oder 16-Bit-Bereich
d		16-Bit-Speicherbereich für Operationsergebnis XOR

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value_1	WORD	2#1111000011001100	
2	VAR	value_2	WORD	2#1100000010101010	
3	VAR	output_value	WORD	0	result after a 0->1 leading edge from start: 2#0011000001100110
4	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

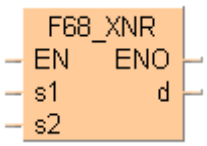
```
IF start THEN
    F67_XOR(value_1, value_2, output_value);
END_IF;
```

F68_XNR

16-Bit-EXCLUSIV-NOR-Verknüpfung invertiert

Erklärung

Der 16-Bit-Wert des Speicherregisters oder eine Konstante **s1** und der 16-Bit-Wert des Speicherregisters oder eine Konstante **s2** werden bitweise exklusiv NOR-verknüpft und invertiert, wenn der **EN**-Eingang auf TRUE gesetzt ist. Das Ergebnis der exklusiven NOR-Verknüpfung wird im Speicherregister **d** abgelegt. Wenn eine äquivalente 16-Bit-Konstante in **s1** oder **s2** angegeben ist, wird die exklusive NOR-Operation intern ausgeführt und eine Konvertierung in einen binären 16-Bit-Ausdruck durchgeführt. Mit diesem Befehl können Sie die Anzahl identischer Bits in den beiden 16-Bit-Daten prüfen.



Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0
s1	0 1 0 0	1 1 0 1	1 0 1 1	1 0 0 1

Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0
s2	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1

start: ON

Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0
d	1 0 1 1	0 0 1 0	1 0 1 1	1 0 0 1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F68_XNR (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Konstante oder 16-Bit-Bereich
s2		Konstante oder 16-Bit-Bereich
d		16-Bit-Speicherbereich für Operationsergebnis NOR

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

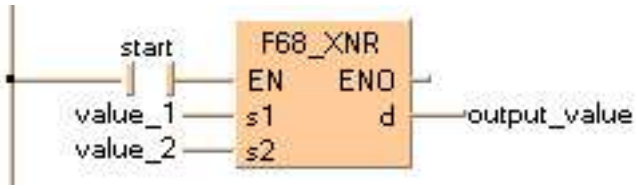
Beispiel In diesem Beispiel wird die Funktion F68_XNR im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value_1	WORD	2#1111000011001100	
2	VAR	value_2	WORD	2#1100000010101010	
3	VAR	output_value	WORD	0	result after a 0->1 leading edge from start: 2#1100111110011001
4	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



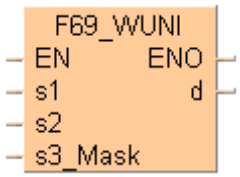
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F68_XNR(value_1, value_2, output_value);
END_IF;
```

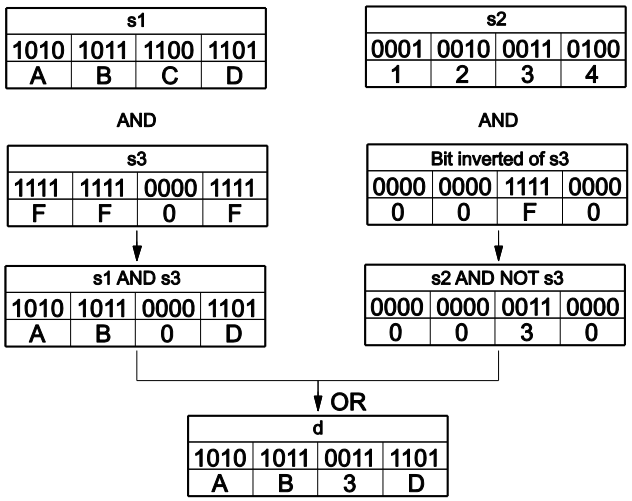
F69_WUNI

16-Bit-Verknüpfung

Erklärung Die Funktion berechnet eine bitweise Verknüpfung zweier Werte an den Eingängen **s1** und **s2** mit einem Wert am Eingang **s3**. Das Funktionsergebnis wird am Ausgang **d** zurückgegeben. Die Verknüpfung wird wie folgt berechnet:



[d] = ([s1] UND [s3]) ODER ([s2] UND (NICHT[s3]))



Wenn der Wert am Eingang **s3** = 16#0, wird der Wert vom Eingang **s2** am Ausgang **d** zurückgegeben.

Wenn der Wert am Eingang **s3** = 16#FFFF, wird der Wert vom Eingang **s1** am Ausgang **d** zurückgegeben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F69_WUNI (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Konstante oder 16-Bit-Bereich
s2		Konstante oder 16-Bit-Bereich
s3		16-Konstante oder 16-Bit-Bereich, womit s1 und s2 wie oben angezeigt verknüpft wird
d		16-Bit-Speicherbereich für Ergebnis

Die Variablen **s1, s2, s3** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

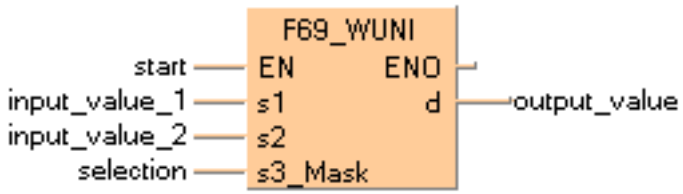
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value1	WORD	16#ABCD	
2	VAR	input_value2	WORD	16#1234	
3	VAR	selection	WORD	16#FF0F	selection:
4	VAR	output_value	WORD	0	result: here 16#AB3D

Hier wurden die Eingangsvariablen **input_value_1** und **input_value _2** deklariert. Stattdessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST

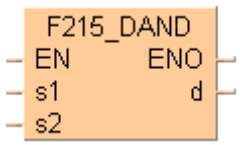
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F69_WUNI( s1:= input_value1,
        s2:= input_value2,
        s3_Mask:= selection,
        d=> output_value);
END_IF;
```

F215_DAND

32-Bit-UND-Verknüpfung

Erklärung Die Funktion berechnet eine bitweise UND-Verknüpfung zweier Werte an den Eingängen **s1** und **s2**. Das Funktionsergebnis wird am Ausgang **d** zurückgegeben.



Wahrheitstabelle:

s1	s2	d
0	0	0
0	1	0
1	0	0
1	1	1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F215_DAND (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Konstante oder 32-Bit-Bereich
s2		Konstante oder 32-Bit-Bereich
d		32-Bit-Speicherbereich für Operationsergebnis AND

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value_1	DWORD	16#12345678	
2	VAR	input_value_2	DWORD	16#90ABCDEF	
3	VAR	output_value	DWORD	0	result: here 16#10204468

Hier wurden die Eingangsvariablen **input_value_1** und **input_value _2** deklariert. Statt dessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

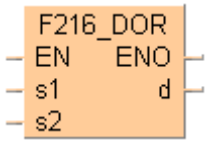
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF START THEN
    F215_DAND(dint1, dint2, dint3);
END_IF;
```

F216_DOR

32-Bit-ODER-Verknüpfung

Erklärung Die Funktion berechnet eine bitweise ODER-Verknüpfung zweier Werte an den Eingängen **s1** und **s2**. Das Funktionsergebnis wird am Ausgang **d** zurückgegeben.



Wahrheitstabelle:

s1	s2	d
0	0	0
0	1	1
1	0	1
1	1	1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F216_DOR (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Konstante oder 32-Bit-Bereich
s2		Konstante oder 32-Bit-Bereich
d		32-Bit-Speicherbereich für Operationsergebnis OR

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value_1	DWORD	16#12345678	
2	VAR	input_value_2	DWORD	16#90ABCDEF	
3	VAR	output_value	DWORD	0	result: here 16#92BFDFFF

Hier wurden die Eingangsvariablen **input_value_1** und **input_value _2** deklariert. Statt dessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

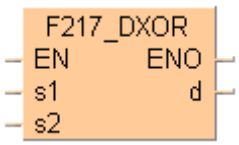
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F216_DOR(input_value_1, input_value_2, output_value);
END_IF;
```

F217_DXOR

32-Bit-EXCLUSIV-ODER-Verknüpfung

Erklärung Die Funktion berechnet eine bitweise ODER-Verknüpfung zweier Werte an den Eingängen **s1** und **s2**. Das Funktionsergebnis wird am Ausgang **d** zurückgegeben.



Wahrheitstabelle:

s1	s2	d
0	0	0
0	1	1
1	0	1
1	1	0

Sie können mit dieser Funktion z.B. prüfen, wie viele verschiedene Bits zwei 32-Bit-Werte enthalten. An jeder Stelle, an der die Bits der Eingänge **s1** und **s2** nicht übereinstimmen, wird im Ergebnis eine 1 erzeugt.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F217_XDOR (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Konstante oder 32-Bit-Bereich
s2		Konstante oder 32-Bit-Bereich
d		32-Bit-Speicherbereich für Operationsergebnis XOR

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

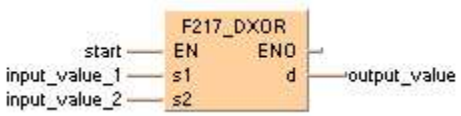
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value_1	DWORD	16#12345678	
2	VAR	input_value_2	DWORD	16#90ABCDEF	
3	VAR	output_value	DWORD	0	result: here 16#829F9B97

Hier wurden die Eingangsvariablen **input_value_1** und **input_value _2** deklariert. Statt dessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



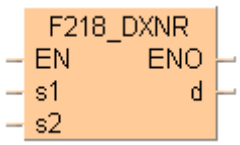
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F217_DXOR(input_value_1, input_value_2, output_value);
END_IF
```

F218_DXNR

32-Bit-EXCLUSIV-ODER-Verknüpfung NEGIERT

Erklärung Die Funktion berechnet eine bitweise EXKLUSIV-ODER-Verknüpfung NEGIERT zweier Werte an den Eingängen **s1** und **s2**. Das Funktionsergebnis wird am Ausgang **d** zurückgegeben.



Wahrheitstabelle:

s1	s2	d
0	0	1
0	1	0
1	0	0
1	1	1

Mit dieser Funktion können Sie z.B. prüfen, wie viele gleiche Bits zwei 32-Bit-Werte enthalten. An jeder Stelle, an der die Bits der Eingänge **s1** und **s2** übereinstimmen, wird zum Ergebnis eine 1 addiert.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F218_DXNR (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Konstante oder 32-Bit-Bereich
s2		Konstante oder 32-Bit-Bereich
d		32-Bit-Speicherbereich für Operationsergebnis XNR

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

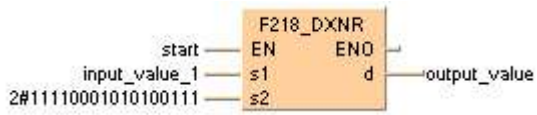
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value_1	DWORD	2#10101110101011110001	bit combination
2	VAR	output_value	DWORD	0	result: here 2#11111111111101001111011110101001

Rumpf Wenn die Variable **output** auf den Wert TRUE gesetzt ist, wird die Funktion F218_DXNR ausgeführt.

KOP



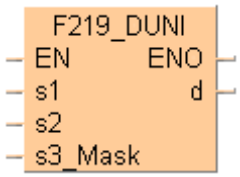
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F218_DXNR(input_value_1, 2#11110001010100111, output_value);
END_IF;
```

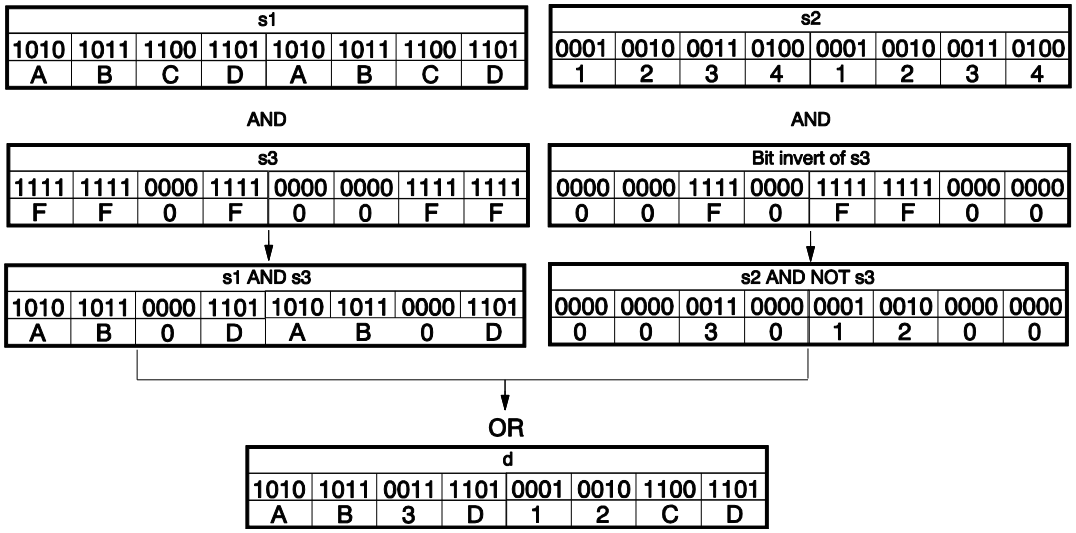
F219_DUNI

32-Bit-Verknüpfung

Erklärung Die Funktion berechnet eine bitweise Verknüpfung zweier Werte an den Eingängen **s1** und **s2** mit einem Wert am Eingang **s3**. Das Funktionsergebnis wird am Ausgang **d** zurückgegeben. Die Verknüpfung wird wie folgt berechnet:



[d] = ([s1] UND [s3]) ODER ([s2] UND (NICHT[s3]))



Wenn der Wert am Eingang **s3** = 16#0 ist, wird der Wert vom Eingang **s2** am Ausgang **d** zurückgegeben.

Wenn der Wert am Eingang **s3** = 16#FFFFFFFF ist, wird der Wert vom Eingang **s1** am Ausgang **d** zurückgegeben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F219_DUNI (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Konstante oder 32-Bit-Bereich
s2		Konstante oder 32-Bit-Bereich
s3		32-Konstante oder 32-Bit-Bereich, womit s1 und s2 wie oben angezeigt verknüpft wird
d		32-Bit-Speicherbereich für Ergebnis

Die Variablen **s1, s2, s3** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das berechnete Ergebnis 0 ist.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

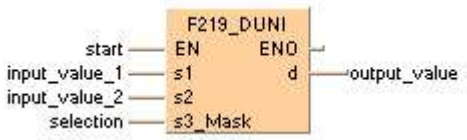
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value_1	DWORD	16#ABCDABCD	
2	VAR	input_value_2	DWORD	16#12341234	
3	VAR	selection	DWORD	16#FF0F00FF	selection:
4	VAR	output_value	DWORD	0	result: here 16#AB3D12CD

Hier wurden die Eingangsvariablen **input_value_1** und **input_value _2** deklariert. Statt dessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



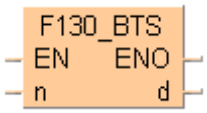
ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F219_DUNI( s1:= input_value1,
               s2:= input_value2,
               s3_Mask:= selection,
               d=> output_value);
END_IF;
```

F130_BTS Bit-Setzen

Erklärung Im Speicherregister oder der Konstanten **d** wird ein Bit an der Bitposition **n** gesetzt, wenn der **EN**-Eingang auf TRUE gesetzt ist. Andere Bits als das angegebene ändern sich nicht. Der Wertebereich von **n** liegt zwischen 0 und 15.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F130_BTS (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich
n	INT	Spezifiziert zu setzende Bit-Position

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

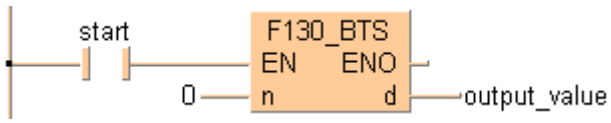
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	output_value	WORD	2#101010	result after a 0->1 leading edge from start: 2#101011
2	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

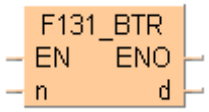


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F130_BTS( n:= 0,
        d=> output_value);
END_IF;
```

F131_BTR Bit-Rücksetzen

Erklärung Im Speicherregister **d** wird ein Bit zurückgesetzt, wenn der **EN**-Eingang auf TRUE gesetzt ist. Mit dem Inhalt im Speicherregister oder der Konstanten **n** wird festgelegt, welches Bit im Quellwort zurückgesetzt wird. Andere Bits als das angegebene ändern sich nicht. Der Wertebereich von **n** liegt zwischen 0 und 15.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F131_BTR (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich
n	INT	Spezifiziert zurückzusetzende Bit-Position

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

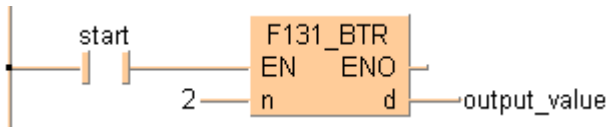
Beispiel In diesem Beispiel wird die Funktion F131_BTR im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	output_value	WORD	2#10101	result after a 0->1 leading edge from start: 2#10001
2	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F131_BTR( n:= 2,
        d=> output_value);
END_IF;
```

F132_BTI Bit-Invertieren

Erklärung Im Speicherregister **d** wird das Bit an Bitposition **n** von [1 (TRUE) → 0 (FALSE) oder 0 (FALSE) → 1 (TRUE)] invertiert, wenn der **EN**-Eingang auf TRUE gesetzt ist. Andere Bits als das angegebene ändern sich nicht. Der Wertebereich von **n** liegt zwischen 0 und 15.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F132_BTI (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich
n	INT	Spezifiziert zu invertierende Bit-Position

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

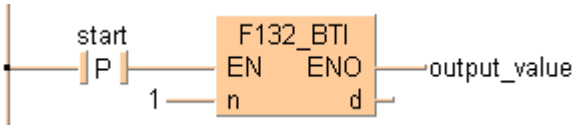
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	output_value	WORD	2#111	result after a 0->1 leading
2	VAR				edge from start: 2#101

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



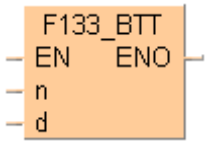
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F132_BTI( n:= 1,
        d=> output_value);
END_IF;
```

F133_BTT Bittest

Erklärung

Es wird der Zustand [1 (TRUE) oder 0 (FALSE)] der Bitposition **n** im Speicherregister **d** geprüft, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Das getestete Bit wird im Zero-Flag (R900B) gespeichert.

- Wenn das getestete Bit "0" (FALSE) ist, ist das Zero-Flag (R900B) kurzzeitig gesetzt (TRUE).
- Wenn das getestete Bit "1" (TRUE) ist, ist das Zero-Flag (R900B) auf FALSE gesetzt.

n spezifiziert die Bitposition, die in den Dezimaldaten getestet werden soll.
Bereich von **n**: 0 bis 15

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F133_BTT (s. S. 1187)

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich
n	INT	Spezifiziert zu testende Bit-Position

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Beispiel

In diesem Beispiel wird die Funktion F133_BTT im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

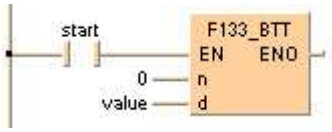
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	bit0_is_TRUE	BOOL	FALSE	TRUE if bit LSB of value is TRUE else FALSE
2	VAR	value	WORD	2#101	result after a 0->1 leading edge: 2#101 zero-flag (R900B) has state FALSE
3	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

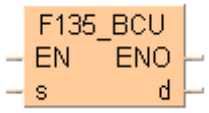
```
IF start THEN
    F133_BTT( n:= 0,
              d:= value);
    IF R900B THEN
        bit0_is_TRUE := FALSE;
    ELSE
        bit0_is_TRUE := TRUE;
    END_IF;
END_IF;
```

F135_BCU

16-Bit-Bittest; Anzahl der gesetzten Bits

Erklärung

Der 16-Bit-Wert des Speicherregisters oder eine Konstante **s** wird auf die Anzahl der gesetzten Bits getestet, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Das Zählergebnis wird im Speicherregister **d** als binärcodierter Wert abgelegt.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F135_BCU (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	Quelle
n	INT	Zielbereich zum Speichern der Anzahl der Bits, die sich im AN-Status (1) befinden

Operanden

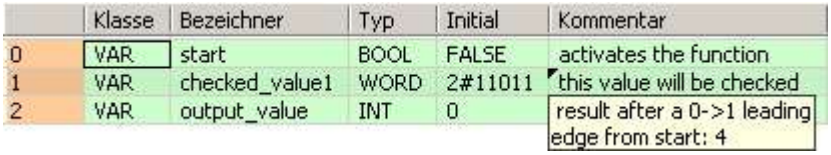
Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Beispiel

In diesem Beispiel wird die Funktion F135_BCU im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

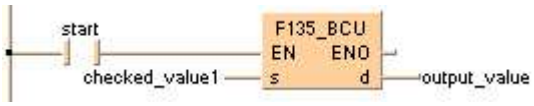


	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	checked_value1	WORD	2#11011	this value will be checked
2	VAR	output_value	INT	0	result after a 0->1 leading edge from start: 4

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST

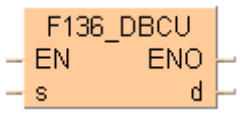
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F135_BCU(checked_value1, output_value);
END_IF;
```

F136_DBCU

32-Bit-Bittest; Anzahl der gesetzten Bits

Erklärung Der 32-Bit-Wert des Speicherregisters oder eine Konstante **s** wird auf die Anzahl der gesetzten Bits getestet, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Das Zählergebnis wird im Speicherregister **d** als binärcodierter Wert abgelegt.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F136_DBCU (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s	ANY32	Quelle
d	INT	Zielbereich zum Speichern der Anzahl der Bits, die sich im AN-Status (1) befinden

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

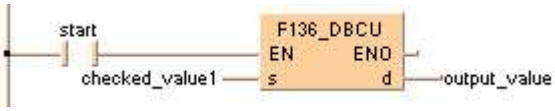
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	checked_value	DWORD	16#1111FFFF	this value will be checked
2	VAR	output_value	INT	0	result after a 0->1 leading edge from start: 20
3	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

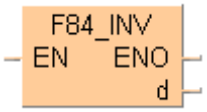
```
IF start THEN
    F136_DBCU(checked_value, output_value);
END_IF;
```

F84_INV

16-Bit-Invertieren (Einerkomplement)

Erklärung

Vom 16-Bit-Wert des Speicherregisters **d** wird das Einerkomplement gebildet, wenn der **EN**-Eingang auf TRUE gesetzt ist. Das Ergebnis wird im Speicherregister **d** abgelegt. Diesen Befehl können Sie zum Beispiel einsetzen, um bei Negativlogik externer Geräte die notwendige Signalumkehrung durchzuführen.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F84_INV (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	zu invertierender 16-Bit-Bereich

Operanden

Für	Merker				T/C		Register			Konst.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	invert_value	WORD	2#1001001101110001	result after a 0->1 leading edge from start: 2#0110110010001110
2	VAR				

Rumpf

Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

ST

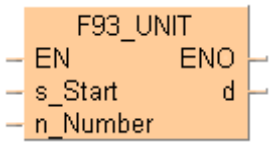
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F84_INV(invert_value);
END_IF;
```

F93_UNIT

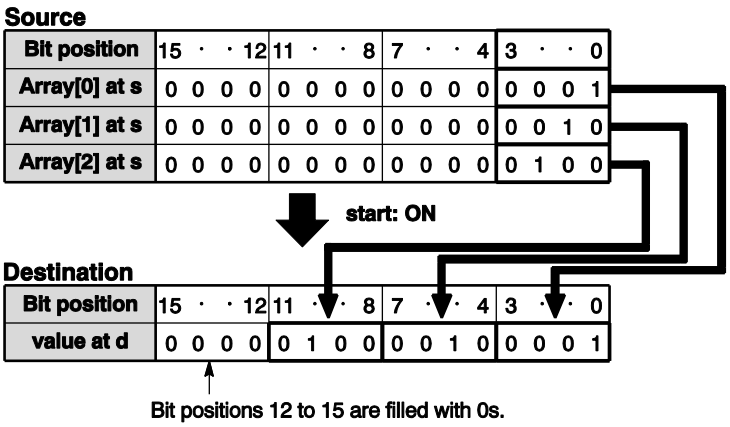
Digit-Kombination

Erklärung Die niederwertigsten Digits der Worte des Speicherbereichs ab der Anfangsadresse **s** werden kombiniert, wenn der **EN**-Eingang auf TRUE gesetzt ist. Das Ergebnis der Kombination wird im Speicherregister **d** abgelegt.



n spezifiziert die Anzahl der Daten die extrahiert werden sollen. Der Wertebereich von **n** liegt zwischen 0 und 4.

Das nachstehende Programmierbeispiel lässt sich wie folgt darstellen:



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F93_UNIT (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	WORD	16-Bit-Anfangsregister das extrahiert werden muss (Quelle)
n	INT	spezifiziert Anzahl der Daten die extrahiert werden sollen
d	WORD	16-Bit-Zielbereich

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	KOP	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ mit Indexmodifizierern definierter Bereich größer als zulässiger Bereich ▪ der Wert an n≥ 5
R9008	%MX0.900.8	kurzzeitig	

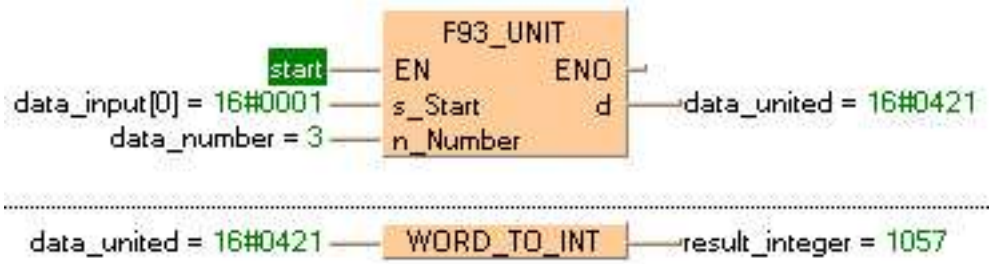
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	TRUE
1	VAR	data_input	ARRAY [0..2] OF WORD	[1,2,4]
2	VAR	data_number	INT	3
3	VAR	data_united	WORD	0
4	VAR	result_integer	INT	0

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Binärwerte in der Abbildung auf der Hauptseite des Befehls dienen als Arraywerte in **data_input**. In diesem Beispiel werden Variablen im POE-Kopf deklariert. Sie können die Werte jedoch direkt an die Eingänge der Funktion schreiben.

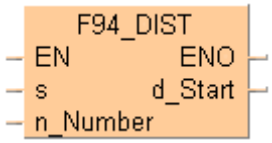
KOP In diesem Beispiel wurde der Monitor mit  aktiviert, so dass Sie die Ergebnisse gleich ablesen können.



F94_DIST

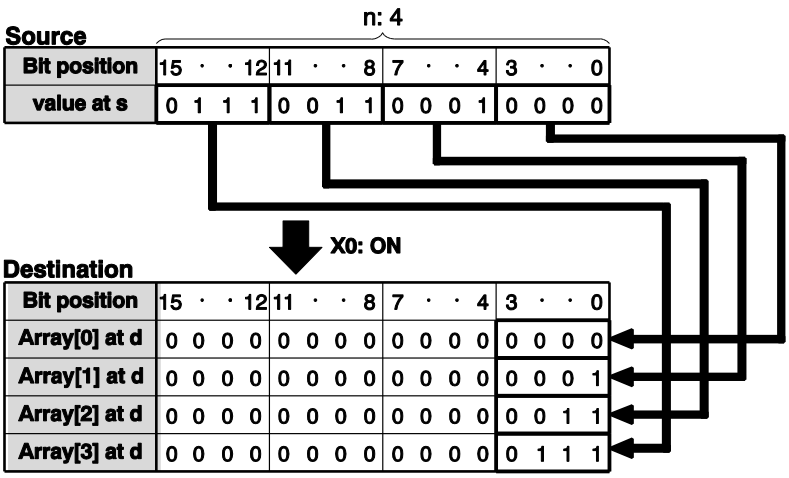
Digit-Verteilung

Erklärung Die Digits des Speicherregisters oder eine Konstanten **s** werden verteilt, wenn der **EN**-Eingang auf TRUE gesetzt ist. Das Ergebnis der Verteilung steht im Speicherbereich ab der Anfangsadresse **d**.



Mit dem Inhalt des Speicherregisters oder der Konstanten **n** wird festgelegt, wie viele Digits des Quellwortes zu verteilen sind. Der Wert für **n_Number** kann zwischen 0 und 4 liegen. Beim Wert 0 wird der Befehl nicht ausgeführt.

Das nachstehende Programmierbeispiel lässt sich wie folgt darstellen:



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F94_DIST (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	WORD	16-Bitregister oder äquivalente Konstante die verteilt werden muss (Quelle)
n	INT	spezifiziert Anzahl der zu verteilenden Daten
d	WORD	16-Bit-Speicherregister zum Speichern verteilter Daten (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s, n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ mit Indexmodifizierern definierter Bereich größer als zulässiger Bereich ▪ der Wert bei n ≥ 5 ist
R9008	%MX0.900.8	kurzzeitig	

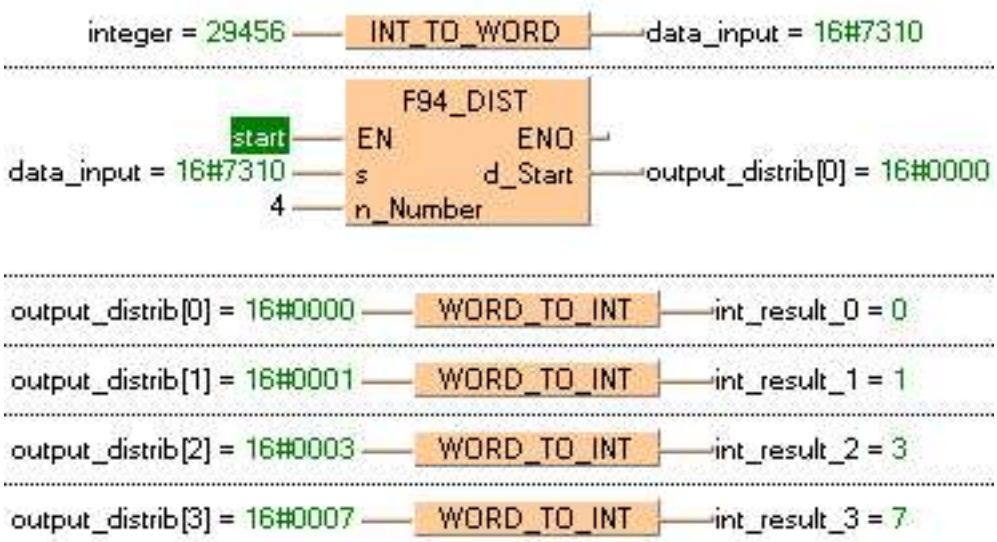
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	integer	INT	29456
1	VAR	data_input	WORD	0
2	VAR	start	BOOL	TRUE
3	VAR	output_distrib	ARRAY [0..3] OF WORD	[4(0)]
4	VAR	int_result_0	INT	0
5	VAR	int_result_1	INT	0
6	VAR	int_result_2	INT	0
7	VAR	int_result_3	INT	0

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Binärwerte in der Abbildung auf der Hauptseite des Befehls sind die berechneten Werte. In diesem Beispiel werden Variablen im POE-Kopf deklariert. Außerdem wird der Wert n = 4 direkt an den Eingang geschrieben.

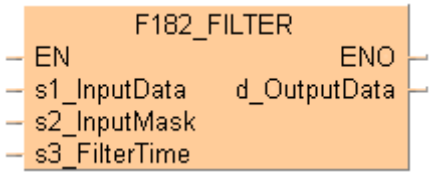
KOP In diesem Beispiel wurde der Monitor mit  aktiviert, so dass Sie die Ergebnisse gleich ablesen können.



F182_FILTER 16-Bit-Signalentprellung

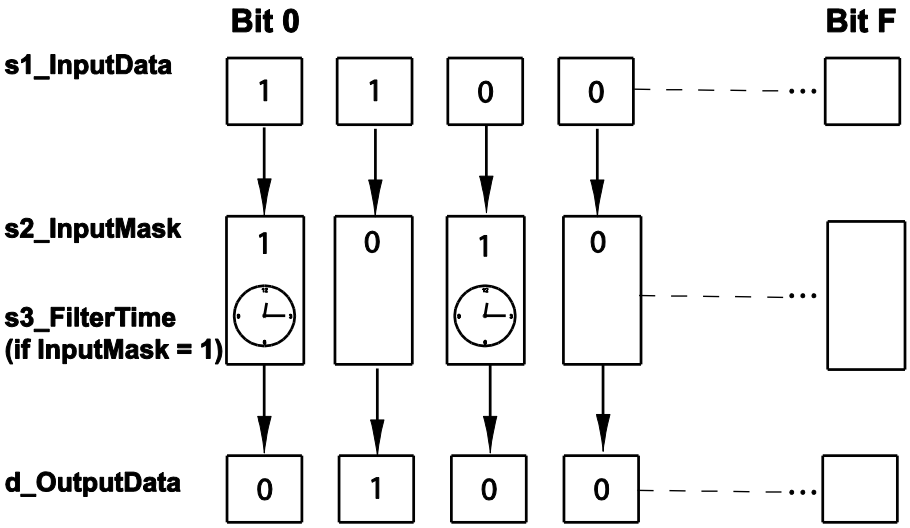
Erklärung

Dieser Befehl führt für bestimmte Bits eine Signalentprellung durch. Dies ist sinnvoll, wenn mechanische Taster oder Schalter an schnellen Eingängen betrieben werden.



Eine Entprellung wird für die **s1_InputData**-Bits ausgeführt, die über **s2_InputMask** mit "1" ausgewählt sind. Das Ergebnis der Entprellung wird an **d_OutputData** ausgegeben.Die Entprellzeit wird über **s3_FilterTime** (0 bis 30000ms) vorgegeben. Bei Maskenbits in **s2_InputMask** mit "0" wird keine Entprellung vorgenommen und das entsprechende Bit am Eingang **s1_InputData** direkt durchgereicht.

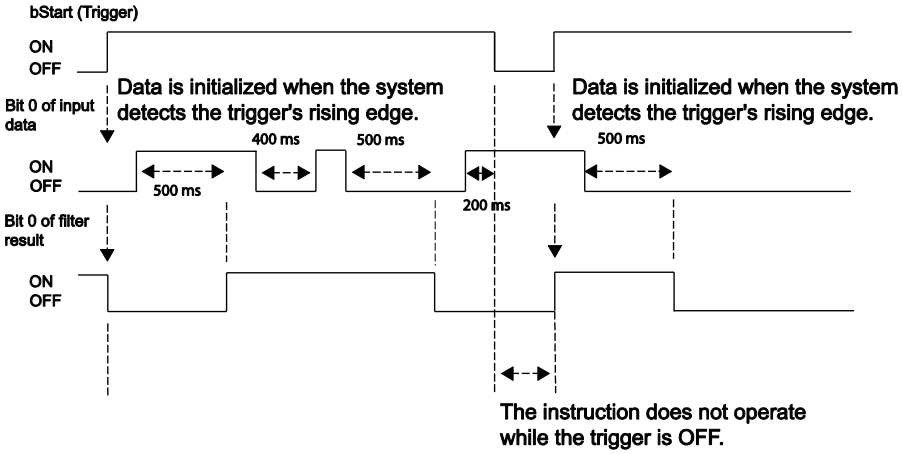
In der nachstehenden Abbildung sind die verschiedenen Filter- und Entprellungsvarianten dargestellt.



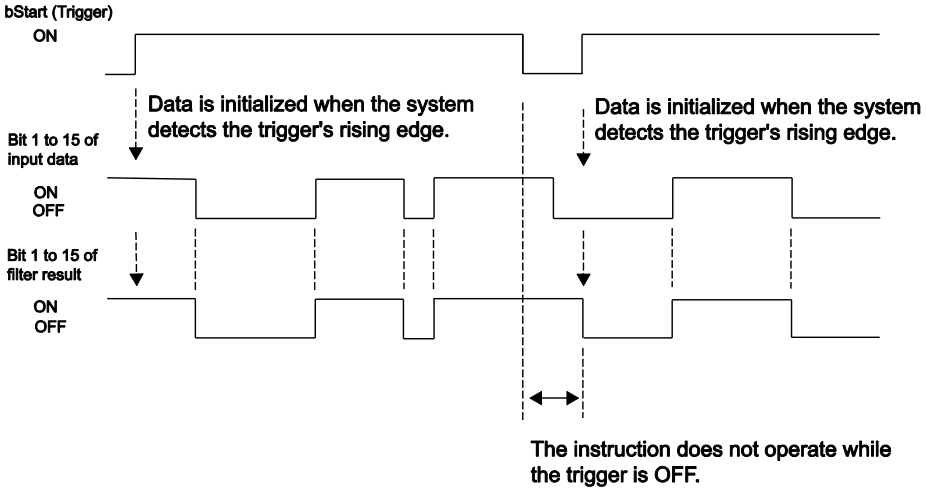
Vorsichtsmaßnahmen bei der Programmierung

Beim ersten Aufruf (pos. Flanke) der Funktion 182 wird der Zustand von **s1_InputData** direkt an **d_OutputData** ausgegeben; es findet dann also keine Entprellung statt. Die Filterzeit kann sich maximal um die Länge eines SPS-Zyklus verlängern.

Zeitdiagramme für eine Entprellung per Filter: Der Variablen s2_InputMask wurde der Wert 1 (16#0001) zugewiesen. Bit 0 wird zur Entprellung gefiltert, aber die anderen Bits werden nicht gefiltert bzw. entprellt. s3_FilterTime wurde der Wert 500ms zugewiesen.



Der Variablen s2_InputMask wurde der Wert 0 (16#0000) zugewiesen. Bit 0 bis F werden nicht entprellt



SPS-Typen **Verfügbarkeit von F182_FILTER (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s1_InputData	ANY16	Eingangsdaten, deren Bit entsprechend der Maske gefiltert werden.
s2_InputMask		Eingangsmaske, die festlegt, welche Bits zur Entprellung gefiltert werden.
s3_FilterTime		Minimale Ein-/Aus-Zeit in ms
d_OutputData		Gefilterte Daten

Operanden

Für	Bitmerker				T/C		Register			Konst.
s1	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2, s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Filterverarbeitungszeit in **s3s3_FilterTime** kleiner als 0 oder größer als 30000 ist.
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet.

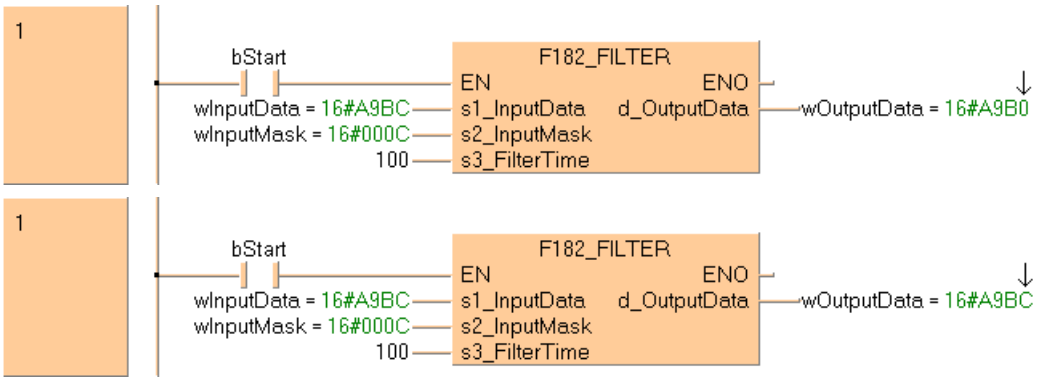
POE Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bStart	BOOL	FALSE	
1	VAR	wInputData	WORD	16#A9BC	
2	VAR	wInputMask	WORD	16#000C	2#0000000000001100 i.e. bits 2 and 3 filtered
3	VAR	wOutputData	WORD	0	
4	VAR	iFilterTime	INT	100	0.1 seconds

In diesem Beispiel werden die Eingangsvariablen **wInputData**, **wInputMask** und **iFilterTime** deklariert. Stattdessen können Sie für **wInputMask** und **iFilterTime** im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben (Freigabe-Eingang z. B. zum Testen). Ferner wurde die Variable **bStart** deklariert, um die Filterfunktion zu starten; die Variable **wOutputData** wurde zum Speichern der Ergebnisse deklariert.

Rumpf Die geprellten Bits werden erst in die Variable **wOutputData** geschrieben, wenn die Filterzeit abgelaufen ist (siehe KOP-Beispiel). In den Zeitdiagrammen (s. S. 533) finden Sie eine genaue Erläuterung. Die Variable **wOutputdata** hat 100 ms lang den Wert 16#A9B0; sobald diese Zeit abgelaufen ist erhält **wOutputData** den Wert 16#A9BC.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF bStart Then
        F182_FILTER(wInputData, wInputMask, iFilterTime, wOutputData);
End_If;
```

Kapitel 17

Bit-Schiebefunktionen

LSR

Links-Schieberegister

Erklärung Mit dem LSR-Befehl können Sie ein 16-Bit Schieberegister im Merkerbereich (**d_WR**) realisieren. Mit jeder steigenden Flanke am Takteingang **ShiftLeftTrigger** wird der Signalzustand am Dateneingang **DataInput** in das Bit 0 des Schieberegisters übernommen. Die dabei aus dem Schieberegister herausgeschobenen Bits (aus Bit 15) gehen verloren.

DataInput: gibt die neuen Daten zum Hineinschieben an.

- Neue Daten = TRUE: wenn der Eingang TRUE ist
- Neue Daten = FALSE: wenn der Eingang FALSE ist

ShiftLeftTrigger: verschiebt ein 1 Bit nach links, wenn die steigende Flanke des Triggers erkannt wird

ReSetTrigger: setzt alle Bits des Datenbereichs auf 0, wenn der Trigger TRUE ist.

Der für diesen Befehl verfügbare Bereich ist nur der interne Doppelwortmerker (WR)

SPS-Typen **Verfügbarkeit von LSR (s. S. 1194)**

☞ **Der Bereich für die Wortregister (WR) von internen Merkern hängt von dem unter Extras → Optionen → Compiler-Optionen → Adressbereiche definierten freien Bereich ab.**

Datentypen

Variable	Datentyp	Funktion
DataInput:	BOOL	wenn TRUE, Daten = 1, wenn FALSE, Daten = 0
shiftLeftTrigger:	BOOL	verschiebt bei TRUE ein Bit nach links
ReSetTrigger:	BOOL	setzt Datenbereich bei TRUE auf 0 zurück
d_WR	ANY16	angegebener Datenbereich für die Datenverschiebung

Operanden

Für	Merker				T/C	
DataInput, ShiftLeftTrigger, ReSetTrigger	X	Y	R	L	T	C
d_WR	-	-	WR	-	-	-

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Output	INT	0
1	VAR	DataInput	BOOL	FALSE
2	VAR	ShiftTrigger	BOOL	FALSE
3	VAR	ResetTrigger	BOOL	FALSE

Rumpf

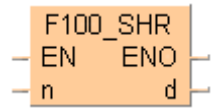
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

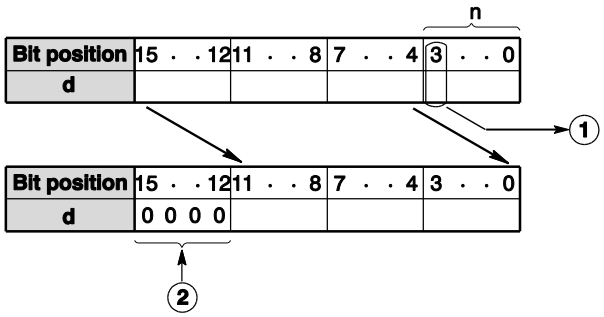
```
Output:=LSR(DataInput, ShiftTrigger, ResetTrigger);
```

F100_SHR

16-Bit-Rechtsschieben

Erklärung Der 16-Bit-Wert des Speicherregisters **d** wird innerhalb des Wortes um **n** Bits nach rechts verschoben, wenn der **EN**-Eingang auf TRUE gesetzt ist.





Wenn **n** Bits nach rechts verschoben werden, werden die Daten im **n**-ten Bit ① im Sondermerker R9009 (Carry-Flag) gespeichert. Die durch die Verschiebung frei werdenden **n** Bitstellen des Speicherregisters **d** ② werden mit 0 aufgefüllt.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F100_SHR (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich, der nach rechts verschoben wird
n	INT	Anzahl der Bits, die verschoben werden

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

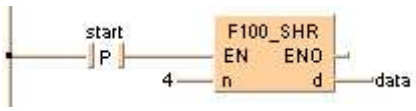
Beispiel In diesem Beispiel wird die Funktion F100_SHR im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data	WORD	16#1234	result after a 0->1 leading edge from start: 16#0123
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



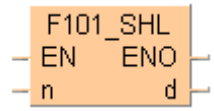
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

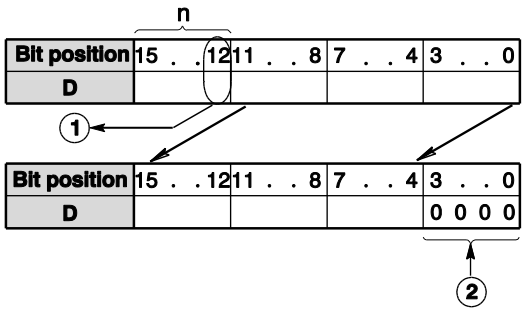
```
IF DF(start) THEN
    F100_SHR( n:= 4 ,
        d=> data );
END_IF;
```

F101_SHL

16-Bit-Linksschieben

Erklärung Der 16-Bit-Wert des Speicherregisters **d** wird innerhalb des Wortes um **n** Bits nach links verschoben, wenn der **EN**-Eingang auf TRUE gesetzt ist.





Wenn **n** Bits nach links verschoben werden, werden die Daten im **n**-ten Bit ①im Sondermerker R9009 (Carry-Flag) gespeichert. Die durch die Verschiebung frei werdenden **n**Bitstellen werden mit 0 aufgefüllt ②.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F101_SHL (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich, der nach links verschoben wird
n	INT	Anzahl der Bits, die verschoben werden

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

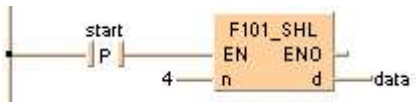
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data	WORD	16#1234	result after a 0->1 leading edge from start: 16#2340
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



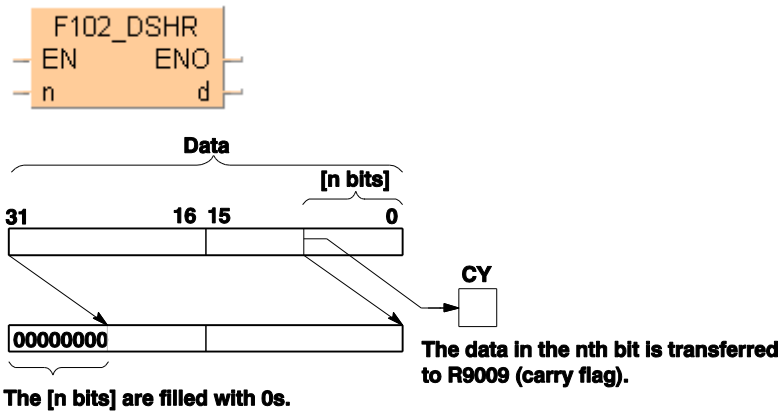
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F101_SHL( n:= 4,
        d=> data);
END_IF;
```

F102_DSHR

32-Bit-Rechtsschieben

Erklärung Die Funktion schiebt den Wert am Ausgang **d** nach rechts. Die Anzahl der Bits um die der Wert am Ausgang **d** nach rechts geschoben werden soll, wird mit dem Wert am Eingang **n** festgelegt. Dieser kann zwischen 0 und 255 liegen (nur das niederwertige Byte von **n** ist wirksam). Die durch die Verschiebung freiwerdenden Bitstellen werden mit Nullen aufgefüllt. Bei Eingang **n** = 0 findet keine Verschiebung statt. Eine Verschiebedistanz größer als 32 ist nicht sinnvoll, da bereits mit **n** = 32 der Wert am Ausgang **d** mit Nullen belegt ist. Das Bit an der Position **n** - 1 (das letzte rechts herausgeschobene Bit) wird gleichzeitig im Sondermerker R9009 (Carry-Flag) gespeichert und kann entsprechend ausgewertet werden. Bei **n** = 0 bleibt der Inhalt des Carry-Flag unverändert.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F102_DSHR**

Datentypen

Variable	Datentyp	Funktion
n	INT	Anzahl der Bits, die verschoben werden (Bereich: (Bereich:16#0 bis 16#FF)
d	ANY32	32-Bit-Bereich, der nach rechts verschoben wird

Operanden

Für	Merker				T/C		Register			Konstante
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9009	%MX0.900.9	kurzzeitig	▪ das Bit an der Position n - 1 den Wert 1 hat.

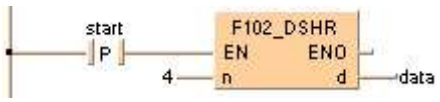
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data	DWORD	16#1234ABCD	result: after a 0->1 leading edge from start: 16#01234ABC
2	VAR				

Rumpf Wenn die Variable **start**von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie schiebt 4 Bits (entspricht einer Stelle in der hexadezimalen Darstellung) nach rechts heraus. Die durch die Verschiebung entstandenen 4 Bits in **data** werden mit Nullen aufgefüllt. Am Eingang n wird die Konstante 4 direkt an die Funktion geschrieben. Statt dessen können sie auch im POE-Kopf eine Eingangsvariable deklarieren.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

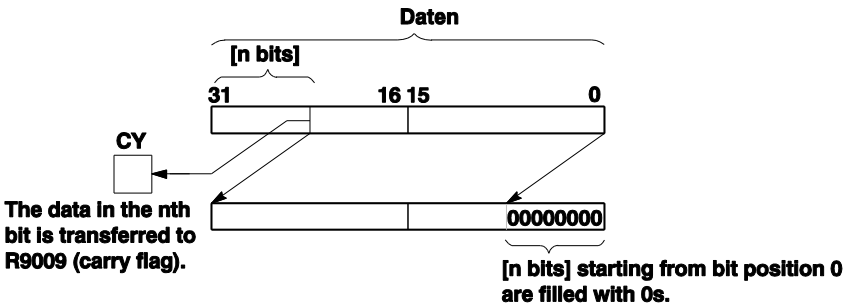
```
IF DF(start) THEN
    F102_DSHR( n:= 4 ,
        d=> data);
END_IF;
```

F103_DSHL

32-Bit-Linksschieben

Erklärung

Die Funktion schiebt den Wert am Ausgang **d** nach links. Die Anzahl der Bits um die der Wert am Ausgang **d** nach links geschoben werden soll, wird mit dem Wert am Eingang **n** festgelegt. Dieser kann zwischen 0 und 255 liegen (nur das niederwertige Byte von **n** ist wirksam). Die durch die Verschiebung freiwerdenden Bitstellen werden mit Nullen aufgefüllt. Bei Eingang **n** = 0 findet keine Verschiebung statt. Eine Verschiebedistanz größer als 32 ist nicht sinnvoll, da bereits mit **n** = 32 der Wert am Ausgang **d** mit Nullen belegt ist. Das Bit an der Position 31- **n** (das letzte links herausgeschobene Bit) wird gleichzeitig im Sondermerker R9009 (Carry-Flag) gespeichert und kann entsprechend ausgewertet werden. Bei **n** = 0 bleibt der Inhalt des Carry-Flag unverändert.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F103_DSHL**

Datentypen

Variable	Datentyp	Funktion
n	INT	Anzahl der Bits, die verschoben werden (Bereich: (Bereich:16#0 bis 16#FF)
d	ANY32	32-Bit-Bereich, der nach links verschoben wird

Operanden

Für	Merker				T/C		Register			Konstante
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9009	%MX0.900.9	kurzzeitig	▪ das Bit an der Position 31 - n den Wert 1 hat.

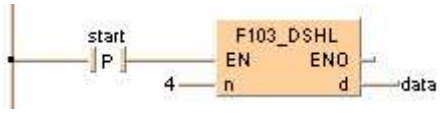
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data	DWORD	16#1234ABCD	result after a 0->1 leading edge from start: 16#234ABCD0
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie schiebt 4 Bits (entspricht einer Stelle in der hexadezimalen Darstellung) nach links heraus. Die durch die Verschiebung entstandenen 4 Bits in **data** werden mit Nullen aufgefüllt. Am Eingang n wird die Konstante 4 direkt an die Funktion geschrieben. Statt dessen können sie auch im POE-Kopf eine Eingangsvariable deklarieren.

KOP



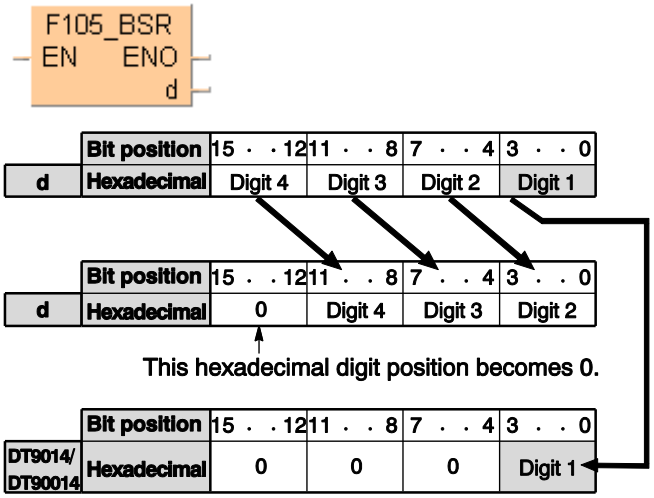
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F103_DSHL( n:= 4,
        d=> data);
END_IF;
```

F105_BSR

4-Digit-BCD-Rechtsschieben

Erklärung Der 16-Bit-Wert des Speicherregisters **d** wird innerhalb des Wortes um 1 Digit nach rechts verschoben, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Wenn ein hexadezimales Digit nach rechts verschoben wird,

- wird die durch **d** angegebene hexadezimale Digit-Position 0 (Bit-Position 0 bis 3) der Daten herausgeschoben und im niederwertigsten Digit des Sonder-Datenregisters DT9014 (DT90014 für FP2/2SH, FP10/10S/10SH) gespeichert.
- wird die hexadezimale Digit-Position 3 (Bit-Position 12 bis 15) des 16-Bit-Bereichs, angegeben durch **d**, gleich Null.
- Dieser Befehl ist sinnvoll, wenn der BCD-codierte 16-Bit-Wert verwendet wird.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F105_BSR (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich, der nach rechts verschoben wird

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

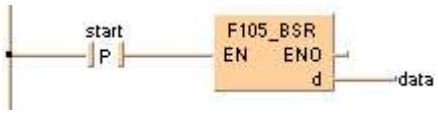
Beispiel In diesem Beispiel wird die Funktion F105_BSR im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data	WORD	16#1234	result after a 0->1 leading edge
2	VAR				from start: 16#0123

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

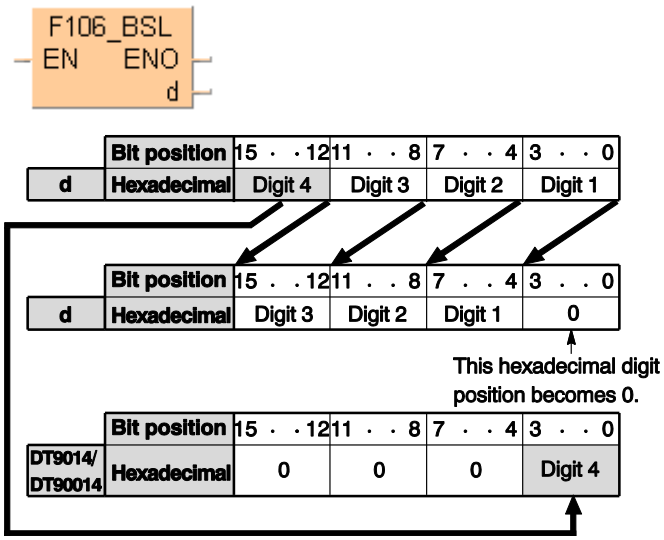


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F105_BSR(data);
END_IF;
```

F106_BSL 4-Digit-BCD-Linksschieben

Erklärung Der 16-Bit-Wert des Speicherregisters **d** wird innerhalb des Wortes um 1 Digit nach links verschoben, wenn der **EN**-Eingang auf TRUE gesetzt ist.



- Wenn ein hexadezimales Digit nach links verschoben wird,
- wird die durch **d** angegebene hexadezimale Digit-Position 3 (Bit-Position 12 bis 15) der Daten herausgeschoben und im niederwertigsten Digit des Sonder-Datenregisters DT9014 (DT90014 für FP2/2SH, FP10/10S/10SH) gespeichert.
- wird die hexadezimale Digit-Position 0 (Bit-Position 0 bis 3) des 16-Bit-Bereichs, angegeben durch **d**, gleich Null.

Dieser Befehl ist sinnvoll, wenn der BCD-codierte 16-Bit-Wert verwendet wird.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F106_BSL (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich, der nach links verschoben wird

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data	WORD	16#1234	result after a 0->1 leading edge from start: 16#2340
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

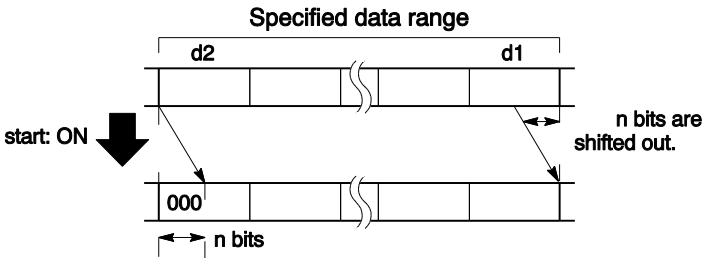
KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F106_BSL(data);
END_IF;
```

F108_BITR Bereich-Rechtsschieben

Erklärung Die Funktion schiebt die Bits eines Speicherbereichs, dessen Anfang und Ende an den Ausgängen **d1** und **d2** anliegt nach rechts. Die Anzahl der Bits um die der Speicherbereich nach rechts geschoben werden soll, wird mit dem Wert am Eingang **n** festgelegt. Die durch die Verschiebung freiwerdenden Bitstellen werden mit Nullen aufgefüllt. Bei Eingang **n** = 0 findet keine Verschiebung statt. Bei einem Eingang **n** = 16 findet eine Verschiebung um ein WORT statt, d.h. es wird wie beim Befehl F110_WHSL (s. S. 555) vorgegangen.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F108_BITR (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d1	ANY16	16-Bit-Anfangsadresse
d2		16-Bit-Endadresse
n	INT	Anzahl der Bits, die verschoben werden

Die Adressen der Variablen von den Eingängen **d1** und **d2** müssen vom gleichen Adresstyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ wenn die Adresse der Variablen an den Ausgängen von d1 > d2 ist oder der Wert am Eingang n ≥ 16 ist.
R9008	%MX0.900.8	kurzzeitig	▪ wenn die Adresse der Variablen an den Ausgängen von d1 > d2 ist oder der Wert am Eingang n ≥ 16 ist.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

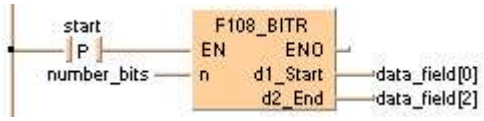
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..2] OF WORD	[16#1234,16#ABCD,16#5678]	Arbitrarily large data field, result: after a 0->1 leading edge of start data_field[0] = 16#D123 data_field[1] = 16#0ABC data_field[2] = 16#0567
2	VAR	number_bits	INT	4	
3	VAR				

Hier wurde die Eingangsvariable **number_bits** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start**von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie schiebt 4 Bits (entspricht einer Stelle in der hexadezimalen Darstellung) nach rechts heraus. Die durch die Verschiebung entstandenen 4 Bits in **data_field[2]** werden mit Nullen aufgefüllt.

KOP



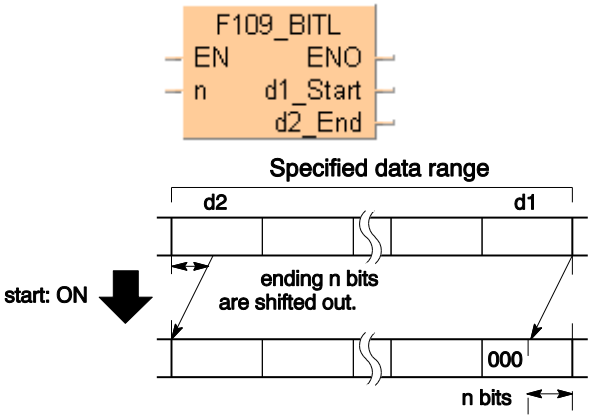
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F108_BITR( n:=number_bits,
         d1_Start=> data_field[0],
         d2_End=> data_field[2]);
END_IF;
```

F109_BITL

Bereich-Linksschieben

Erklärung Die Funktion schiebt die Bits eines Speicherbereichs, dessen Anfang und Ende an den Ausgängen **d1** und **d2** anliegt nach links. Die Anzahl der Bits um die der Speicherbereich nach links geschoben werden soll, wird mit dem Wert am Eingang **n** festgelegt. Die durch die Verschiebung freiwerdenden Bitstellen werden mit Nullen aufgefüllt. Bei Eingang **n** = 0 findet keine Verschiebung statt. Bei einem Eingang **n** = 16 findet eine Verschiebung um ein WORT statt, d.h. es wird wie beim Befehl F111_WSHL (s. S. 557) vorgegangen.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F109_BITL (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d1	ANY16	16-Bit-Anfangsadresse
d2		16-Bit-Endadresse
n	INT	Anzahl der Bits, die verschoben werden

Die Adressen der Variablen von den Eingängen **d1** und **d2** müssen vom gleichen Adresstyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ wenn die Adresse der Variablen an den Ausgängen von d1 > d2 ist oder der Wert am Eingang n ≥ 16 ist.
R9008	%MX0.900.8	kurzzeitig	▪ wenn die Adresse der Variablen an den Ausgängen von d1 > d2 ist oder der Wert am Eingang n ≥ 16 ist.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field	ARRAY [0..2] OF WORD	[16#1234,16#ABCD,16#5678]	Arbitrarily large data field, result: after a 0->1 leading edge of start: data_field[0] = 16#2340 data_field[1] = 16#BCD1 data_field[2] = 16#678A
2	VAR				

Rumpf Wenn die Variable **start**von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie schiebt 4 Bits (entspricht einer Stelle in der hexadezimalen Darstellung) nach links heraus. Die durch die Verschiebung entstandenen 4 Bits in **data_field[0]** werden mit Nullen aufgefüllt. Am Eingang **n** wird die Konstante 4 direkt an die Funktion geschrieben. Statt dessen können sie auch im POE-Kopf eine Eingangsvariable deklarieren.

KOP

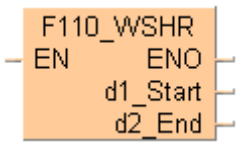
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
     F109_BITL( n:=4,
          d1_Start=> data_field[0],
          d2_End=> data_field[2]);
END_IF;
```

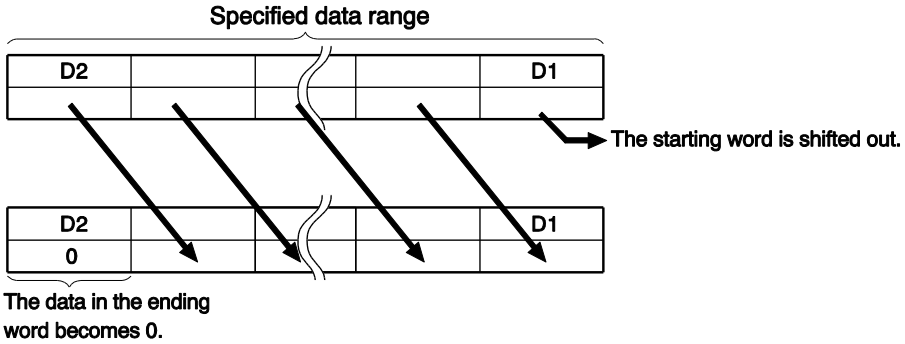
F110_WSHR

Bereich-Rechtsschieben (16-Bit)

Erklärung Der Speicherbereich mit der Anfangsadresse **d1** und der Endadresse **d2** wird um 1 Wort nach niedrigeren Adressen hin (nach rechts) verschoben, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Die durch die Verschiebung freiwerdenden Bitstellen werden mit 0 aufgefüllt. Das Wort mit der niedrigsten Adresse des mit d1 und d2 angegebenen Verschiebebereichs wird gelöscht



Die Adressen **d1** und **d2** müssen

- durch gleiche Operanden spezifiziert werden.
- Es gilt **d1** ≤ **d2**.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F110_WSHR (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d1	16-Bit-Anfangsadresse	
d2	ANY16	16-Bit-Endadresse

Die Variablen **d1** und **d2** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel In diesem Beispiel wird die Funktion F110_WSHR im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	source_array	ARRAY [0..3] OF INT	[2,3,4,5]	result after a 0->1 leading edge from start: [2,4,5,0]
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

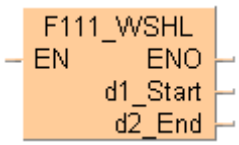
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F110_WSHR( d1_Start=> source_array[1],
        d2_End=> source_array[3]);
END_IF;
```

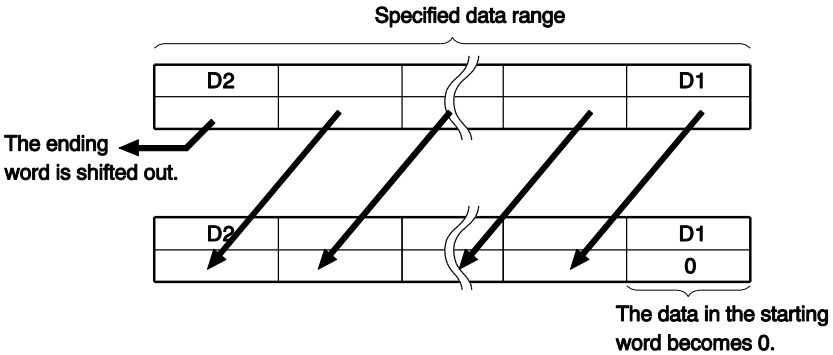
F111_WSHL

Bereich-Linksschieben (16-Bit)

Erklärung Der Speicherbereich mit der Anfangsadresse **d1** und der Endadresse **d2** wird um 1 Wort nach höheren Adressen hin (nach links) verschoben, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Die durch die Verschiebung freiwerdenden Bitstellen werden mit 0 aufgefüllt. Das Wort mit der höchsten Adresse des mit d1 und d2 angegebenen Verschiebebereichs wird gelöscht.



Die Adressen **d1** und **d2** müssen

- durch gleiche Operanden spezifiziert werden.
- Es gilt **d1** ≤ **d2**.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F111_WSHL (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d1	16-Bit-Anfangsadresse	
d2	ANY16	16-Bit-Endadresse

Die Variablen **d1** und **d2** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-

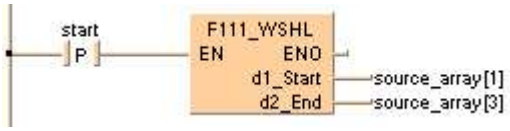
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	source_array	ARRAY [0..3] OF INT	[2,3,4,5]	result after a 0->1 leading edge from start: [2,0,3,4]
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



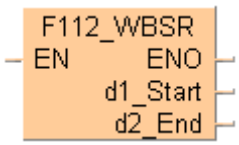
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F111_WSHL( d1_Start=> source_array[1],
        d2_End=> source_array[3]);
END_IF;
```

F112_WBSR

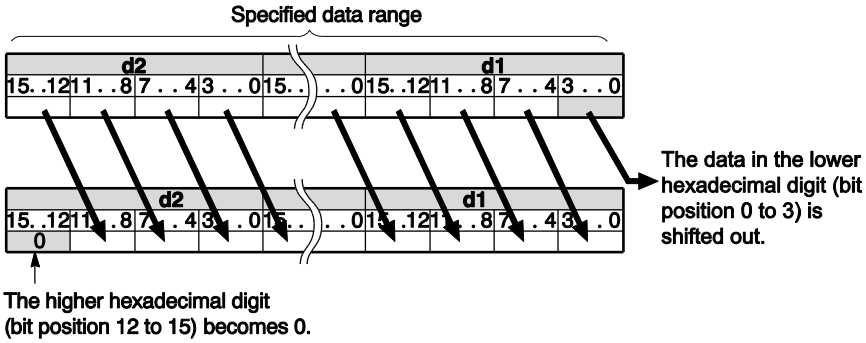
Bereich-BCD-Rechtsschieben

Erklärung Der Speicherbereich mit der Anfangsadresse **d1** und der Endadresse **d2** wird um 1 Digit nach niedrigeren Adressen hin (nach rechts) verschoben, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Wenn ein hexadezimales Digit nach rechts verschoben wird:

- wird das durch **d1** angegebene hexadezimale Digit mit der niedrigsten Adresse (Bit-Position 0 bis 3) herausgeschoben.
- wird das durch **d2** angegebene hexadezimale Digit mit der höchsten Adresse (Bit-Position 12 bis 15) gleich Null.



Die Adressen **d1** und **d2** müssen

- durch gleiche Operanden spezifiziert werden.
- Es gilt **d1** ≤ **d2**.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F112_WBSR (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d1	16-Bit-Anfangsadresse	
d2	ANY16	16-Bit-Endadresse

Die Variablen **d1** und **d2** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel In diesem Beispiel wird die Funktion F112_WBSR im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	source_array	ARRAY [0..3] OF WORD	[16#3456,16#9012,16#5678,16#1234]	result after a 0->1 leading edge from start: [16#3456,16#8901, 16#4567,16#0123]
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

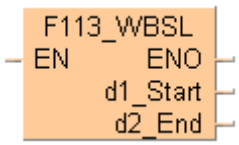
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F112_WBSR( d1_Start=> source_array[1],
        d2_End=> source_array[3]);
END_IF;
```

F113_WBSL

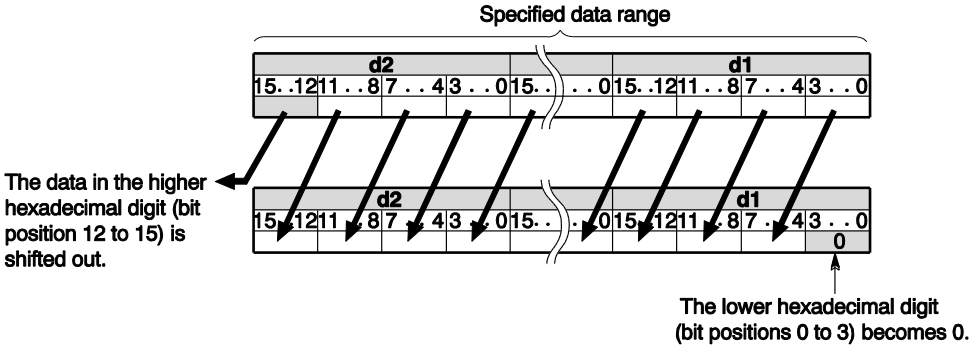
Bereich-BCD-Linksschieben

Erklärung Der Speicherbereich mit der Anfangsadresse **d1** und der Endadresse **d2** wird um 1 Digit nach höheren Adressen hin (nach links) verschoben, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Wenn ein hexadezimales Digit nach links verschoben wird,

- wird das durch **d2** angegebene hexadezimale Digit mit der höchsten Adresse (Bit-Position 12 bis 15) herausgeschoben.
- wird das durch **d1** angegebene hexadezimale Digit mit der niedrigsten Adresse (Bit-Position 0 bis 3) gleich Null.



Die Adressen **d1** und **d2** müssen

- durch gleiche Operanden spezifiziert werden.
- Es gilt **d1** ≤ **d2**.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F113_WBSL (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d1	ANY16	16-Bit-Anfangsadresse
d2		16-Bit-Endadresse

Die Variablen **d1** und **d2** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-

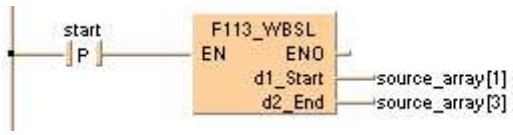
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	source_array	ARRAY [0..3] OF WORD	[16#3456,16#9012,16#5678,16#1234]	result after a 0->1 leading edge from start: [16#3456,16#0120, 16#6789,16#2345]
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

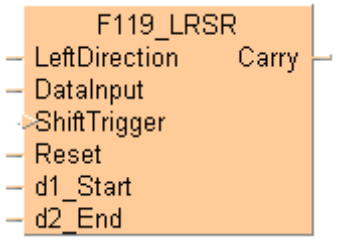


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
     F112_WBSR( d1_Start=> source_array[1],
          d2_End=> source_array[3]);
END_IF;
```

F119_LRSR LINKS/RECHTS-Schieberegister

Erklärung Verschiebt 1 Bit des 16-Bit-Datenbereichs nach links oder rechts.



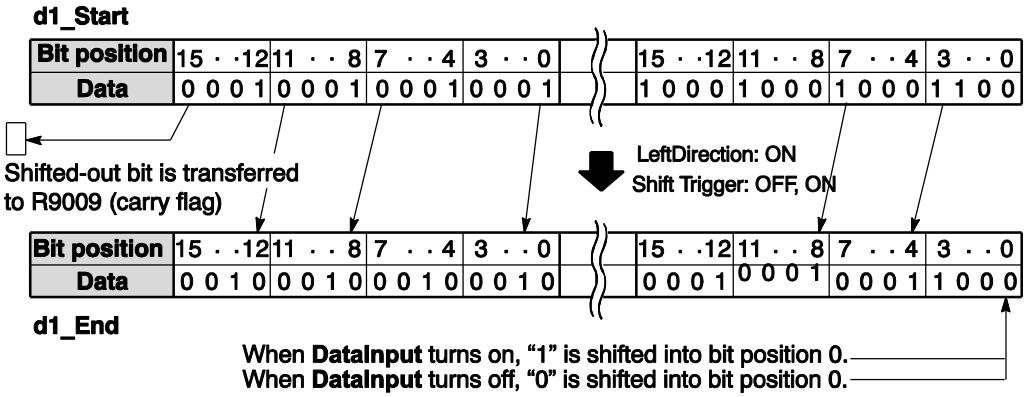
Links-/Rechts-Schieben ist ein Schieberegister, das 1 Bit im festgelegten Datenbereich nach links (auf die höherwertige Bitposition) oder nach rechts (auf die niederwertige Bitposition) verschiebt.

LeftDirection	Links-/rechts-Trigger; gibt die Richtung des Herausschiebens an.	
LeftDirection	= TRUE	Herausschieben nach links.
LeftDirection	= FALSE	Herausschieben nach rechts.
DataInput:	Gibt die neuen Daten zum Hineinschieben an.	
	Neue Daten = TRUE:	wenn DataInput TRUE ist.
	Neue Daten = FALSE:	wenn DataInput FALSE ist.
ShiftTrigger	Verschiebt 1 Bit nach links oder rechts, wenn die steigende Flanke des Trigger ermittelt wird (FALSE → TRUE).	
Reset	Setzt alle Bits des von **d1** und **d2** angegebenen Datenbereichs auf 0, wenn sich dieser Trigger im Zustand TRUE befindet.	
d1	16-Bit-Anfangsadresse	
d2	16-Bit-Endadresse	
Carry	Herausgeschobenes Bit.	

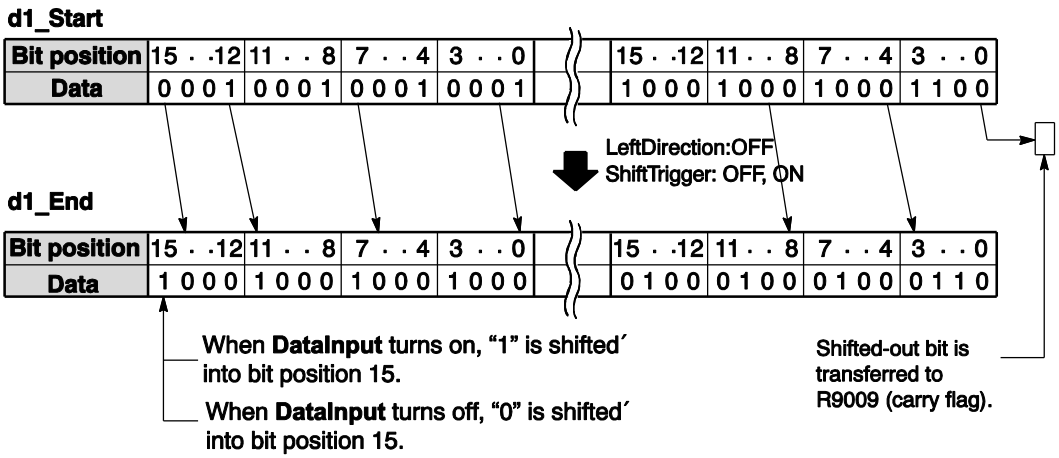
SPS-Typen **Verfügbarkeit von F119_LRSR (s. S. 1186)**

☞
- **Die Variablen d1 und d2 müssen vom gleichen Typ sein.**
- **Diese Funktion erfordert am Ausgang "Carry" keine Variable.**

Left shift operation



Right shift operation



Datentypen

Variable	Datentyp	Funktion
LeftDirection	BOOL	gibt die Richtung der Verschiebung an, TRUE = links, FALSE = rechts
DataInput:	BOOL	Hineingeschobene Daten, TRUE = 1, FALSE = 0
ShiftTrigger	BOOL	aktiviert Verschiebung
Reset	BOOL	setzt die Daten in dem durch d1 und d2 angegebenen Bereich auf Null.
Carry	BOOL	herausgeschobenes Bit
d1	ANY16	16-Bit-Anfangsadresse
d2		16-Bit-Endadresse

Operanden

Für	Merker				T/C		Register			Konstante
LeftDirection, DataInput, ShiftTrigger, Reset	X	Y	R	L	T	C	-	-	-	-
Carry	-	Y	R	L	T	C	-	-	-	-
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel In diesem Beispiel wird die Funktion F119_LRSR im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	data_array	ARRAY [0..2] OF INT	[2#0000000000000001,2#0011111111111111,2#0011111111111110]	
1	VAR	enable_leftShift	BOOL	FALSE	(*function shifts left if TRUE,
2	VAR	reset	BOOL	FALSE	if TRUE, the whole array
3	VAR	input	BOOL	TRUE	specifies the new shift-in data
4	VAR	shift_trigger	BOOL	FALSE	activates the function at a 0->1
5	VAR	carry_out_value	BOOL	FALSE	result after a 0->1 leading edge
6	VAR				

Rumpf Wenn die Variable **enable_leftShift** auf TRUE gesetzt ist, wird nach links verschoben, sonst nach rechts.

KOP

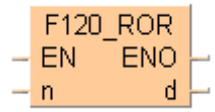
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
carry_out_value:=F119_LRSR( LeftDirection:= enable_leftShift,
                            DataInput:= input,
                            ShiftTrigger:= shift_trigger,
                            Reset:= reset,
                            d1_Start:= data_array[0],
                            d1_End:= data_array[2]);
```

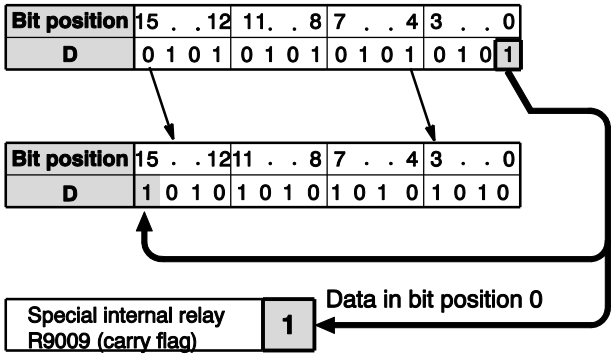
F120_ROR 16-Bit-Rechtsrotieren

Erklärung

Der binäre 16-Bit-Wert des Speicherregister **d** wird innerhalb des Wortes um **n** Bits nach rechts rotiert (ohne Carry-Flag), wenn der **EN**-Eingang auf TRUE gesetzt ist. Rechtsrotieren bedeutet, dass die aus dem Bit 0 (LSB) des Speicherregisters d herausgeschobenen Bits wieder über das Bit 15 (MSB) in das Speicherregister hineingeschoben werden.



Im folgenden Beispiel wird ein Bit nach rechts rotiert:



Wenn **n** Bits nach rechts rotiert werden,

- werden die Daten in der Bitposition **n**-1 (**n**-tes Bit beginnend an der Bitposition 0) im Sondermerker R9009 (Carry-Flag) gespeichert.
- werden **n** Bits, beginnend an der Bitposition 0, nach rechts in die höheren Bitpositionen der von **d** angegebenen 16-Bit-Daten verschoben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F120_ROR (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich
n	INT	Anzahl der Bits, die rotiert werden

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

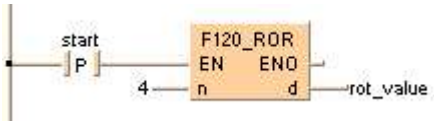
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	rot_value	WORD	16#1234	result after a 0->1 leading edge from start: 16#4123
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

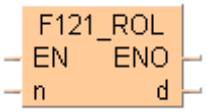
```
IF DF(start) THEN
    F120_ROR( n:= 4,
        d=> rot_value);
END_IF;
```

F121_ROL

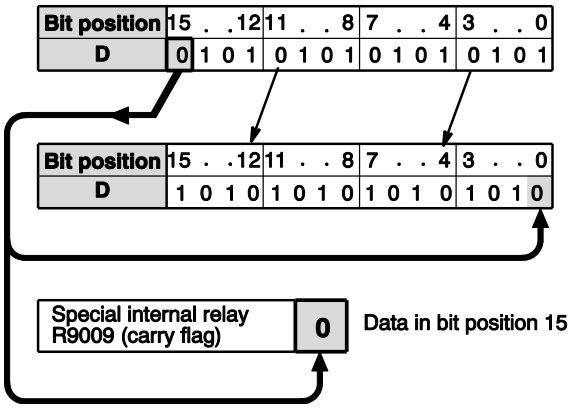
16-Bit-Linksrotieren

Erklärung

Der binäre 16-Bit-Wert des Speicherregister **d** wird innerhalb des Wortes um **n** Bits nach links rotiert (ohne Carry-Flag), wenn der **EN**-Eingang auf TRUE gesetzt ist. Linksrotieren bedeutet, dass die aus dem Bit 0 (LSB) des Speicherregisters d herausgeschobenen Bits wieder über das Bit 15 (MSB) in das Speicherregister hineingeschoben werden.



Im folgenden Beispiel wird ein Bit nach links rotiert:



Wenn **n** Bits nach links rotiert werden,

- werden die Daten in der Bitposition 16-**n** (**n**-tes Bit beginnend an der Bitposition 15) im Sondermerker R9009 (Carry-Flag) gespeichert.
- werden **n** Bits, beginnend an der Bitposition 15, nach links in die niedrigeren Bitpositionen der von **d** angegebenen 16-Bit-Daten verschoben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F121_ROL (s. S. 1186)

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich
n	INT	Anzahl der Bits, die rotiert werden

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Beispiel In diesem Beispiel wird die Funktion F121_ROL im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	rot_value	WORD	16#1234	result after a 0->1 leading edge from start: 16#2341
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

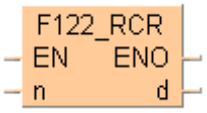
```
IF DF(start) THEN
    F121_ROL( n:= 4,
        d=> rot_value);
END_IF;
```

F122_RCR

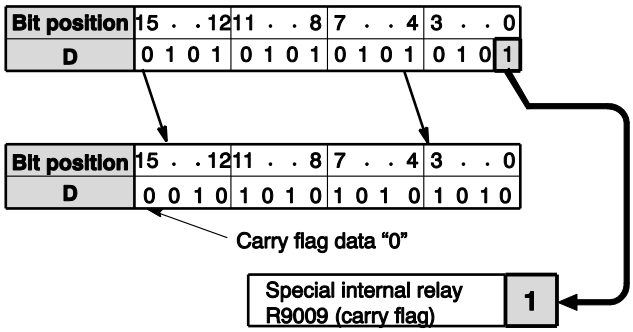
16-Bit-Rechtsrotieren über das Carry-Flag

Erklärung

Der binäre 16-Bit-Wert des Speicherregisters **d** wird innerhalb des Wortes über das Carry-Flag um **n** Bits nach rechts rotiert, wenn der **EN**-Eingang auf TRUE gesetzt ist. Rechtsrotieren über das Carry-Flag bedeutet, dass die aus dem Bit 0 (LSB) des Speicherregisters d herausgeschobenen Bits erst in das Carry-Flag und dann wieder über das Bit 15 (MSB) in das Speicherregister hineingeschoben werden.



Im folgenden Beispiel wird ein Bit nach rechts rotiert:



Wenn **n** Bits mit Carry-Flag-Daten nach rechts rotiert werden,

- werden die Daten in der Bitposition **n**-1 (**n**-tes Bit beginnend an der Bitposition 0) im Sondermerker R9009 (Carry-Flag) gespeichert.
- werden **n** Bits beginnend an der Bitposition 0 nach rechts herausgeschoben und die Carry-Flag-Daten und **n**-1 Bits, beginnend an der Bitposition 0 werden nacheinander in die höheren Bitpositionen der durch **d** angegebenen 16-Bit-Daten verschoben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F122_RCR (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich
n	INT	Anzahl der Bits, die rotiert werden

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

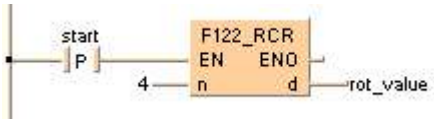
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	rot_value	WORD	16#1234	result after a 0->1 leading edge from start: 16#8123 (!) (carry flag)

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

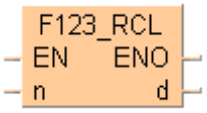
```
IF DF(start) THEN
    F122_RCR( n:= 4,
        d=> rot_value);
END_IF;
```

F123_RCL

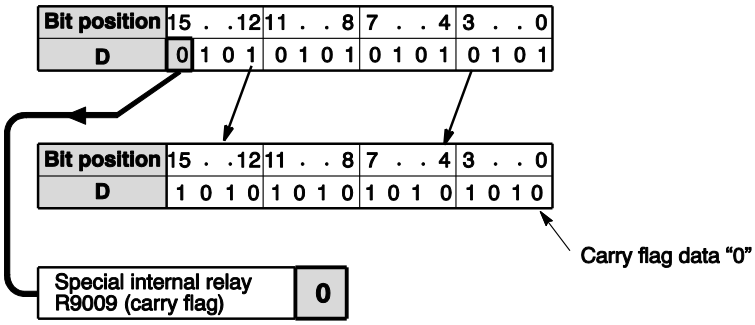
16-Bit-Linksrotieren über das Carry-Flag

Erklärung

Der binäre 16-Bit-Wert des Speicherregisters **d** wird innerhalb des Wortes über das Carry-Flag um **n** Bits nach links rotiert, wenn der **EN**-Eingang auf TRUE gesetzt ist. Linksrotieren über das Carry-Flag bedeutet, dass die aus dem Bit 15 (MSB) des Speicherregisters d herausgeschobenen Bits erst in das Carry-Flag und dann wieder über das Bit 0 (LSB) in das Speicherregister hineingeschoben werden.



Im folgenden Beispiel wird ein Bit nach links rotiert:



Wenn **n** Bits mit Carry-Flag-Daten nach links rotiert werden,

werden die Daten in der Bitposition 16-**n** (**n**--tes Bit beginnend an der Bitposition 15) im Sondermerker R9009 (Carry-Flag) gespeichert.

werden **n** Bits beginnend an der Bitposition 15 nach links herausgeschoben und die Carry-Flag-Daten und **n**-1 Bits, beginnend an der Bitposition 15 werden nacheinander in die niedrigeren Bitpositionen der durch **d** angegebenen 16-Bit-Daten verschoben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F123_RCL (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich
n	INT	Anzahl der Bits, die rotiert werden

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

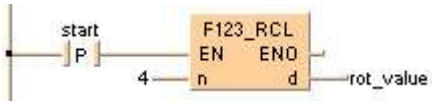
Beispiel In diesem Beispiel wird die Funktion F125_RCL im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	rot_value	WORD	16#1234	result after a 0->1 leading edge from start: 16#2340 (!) (carry flag)
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



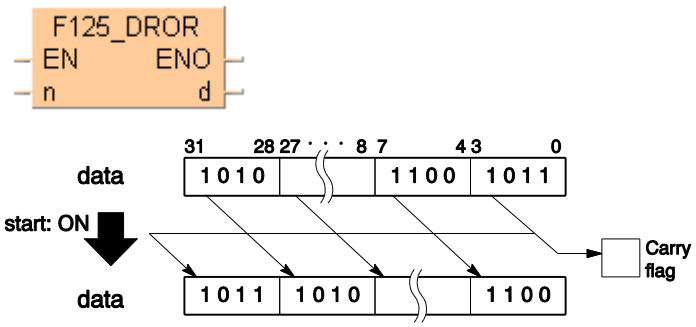
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F123_RCL( n:= 4,
        d=> rot_value);
END_IF;
```

F125_DROR 32-Bit Rechtsrotieren

Erklärung

Die Funktion rotiert den Wert am Ausgang **d** nach rechts. Die Anzahl der Bits, um die der Wert am Ausgang **d** nach rechts rotiert werden soll, wird mit dem Wert am Eingang **n** festgelegt. Dieser kann zwischen 0 und 255 liegen (nur das niederwertige Byte von **n** ist wirksam). Rechtsrotieren bedeutet, dass die aus dem Bit 0 (LSB) herausgeschobenen Bits wieder über das Bit 31 (MSB) in den Wert am Ausgang **d** hineingeschoben werden. Bei einem Eingang **n** = 0 findet kein Rotieren statt. Bei einem Eingang **n** > 32 kann das gleiche Ergebnis mit einer Zahl **n** < 32 erreicht werden: z.B. **n** = 32 gleiches Ergebnis wie **n** = 0; **n** = 33 gleiches Ergebnis wie **n** = 1. Das Bit an der Position **n** - 1 (das letzte rechts herausgeschobene Bit) wird gleichzeitig im Sondermerker R9009 (Carry-Flag) gespeichert und kann entsprechend ausgewertet werden.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F125_DROR (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
n	INT	Anzahl der Bits, die gedreht werden (Bereich: 0 bis 255)
d	ANY32	32-Bit-Bereich

Operanden

Für	Merker				T/C		Register			Konstante
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9009	%MX0.900.9	kurzzeitig	▪ das Bit an der Position n-1 von d den Wert 1 hat.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

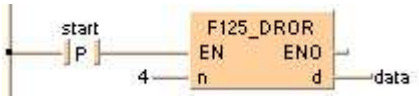
POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data	DWORD	16#1234ABCD	result: after a 0->1 leading edge of start: 16D1234ABC
2	VAR				

Rumpf Wenn die Variable **start**von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie rotiert 4 Bits (entspricht einer Stelle in der hexadezimalen Darstellung) nach rechts. Am Eingang n wird die Konstante 4 direkt an die Funktion geschrieben. Statt dessen können sie auch im POE-Kopf eine Eingangsvariable deklarieren.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
     F125_DROR( n:= 4,
          d=> data);
END_IF;
```

F126_DROL

32-Bit-Linksrotieren

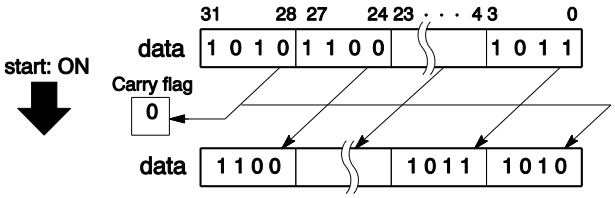
Erklärung

Die Funktion rotiert den Wert am Ausgang d nach links. Die Anzahl der Bits um die der Wert am Ausgang d nach links rotiert werden soll, wird mit dem Wert am Eingang n festgelegt. Dieser kann zwischen 0 und 255 liegen (nur das niederwertige Byte von n ist wirksam). Linksrotieren bedeutet, dass die aus dem Bit 31 (MSB) herausgeschobenen Bits wieder über das Bit 0 (LSB) in den Wert am Ausgang d hineingeschoben werden.

Bei einem Eingang n = 0 findet kein Rotieren statt.

Bei einem Eingang n > 32 kann das gleiche Ergebnis mit einer Zahl n < 32 erreicht werden: z.B. n = 33 gleiches Ergebnis wie n = 0; n = 34 gleiches Ergebnis wie n = 1.

Das Bit an der Position 32- n (das letzte rechts herausgeschobene Bit) wird gleichzeitig im Sondermerker R9009 (Carry-Flag) gespeichert und kann entsprechend ausgewertet werden.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F126_DROR (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
n	INT	Anzahl der Bits, die gedreht werden (Bereich: 0 bis 255)
d	ANY32	32-Bit-Bereich

Operanden

Für	Merker				T/C		Register			Konstante
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9009	%MX0.900.9	kurzzeitig	▪ das Bit an der Position 32 - n von d den Wert 1 hat.

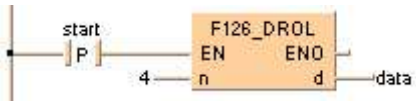
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	output	BOOL	FALSE	activates the function
1	VAR	data	DWORD	16#1234ABCD	result: after a 0->1 leading edge of output: 16#234ABCD1
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie rotiert 4 Bits (entspricht einer Stelle in der hexadezimalen Darstellung) nach links. Am Eingang n wird die Konstante 4 direkt an die Funktion geschrieben. Statt dessen können sie auch im POE-Kopf eine Eingangsvariable deklarieren.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
    F126_DROL( n:= 4,
         d=> data);
END_IF;
```

F127_DRCR 32-Bit-Linksrotieren mit Carry-Flag

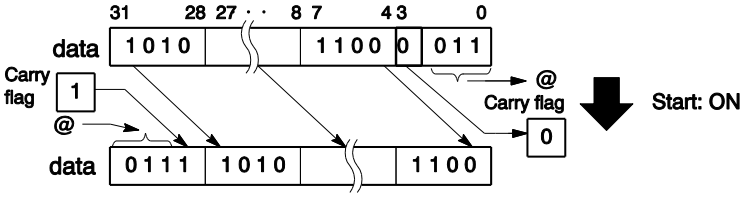
Erklärung

Die Funktion rotiert den Wert am Ausgang **d** mit dem Carry-Flag nach rechts. Die Anzahl der Bits, um die der Wert am Ausgang **d** nach rechts rotiert werden soll, wird mit dem Wert am Eingang **n** festgelegt. Dieser kann zwischen 0 und 255 liegen (nur das niederwertige Byte von **n** ist wirksam).

Der Bitwert an der Bit-Position **n** - 1 wird in das Carry-Flag geschrieben. Die Funktion schiebt **n**-Bits über das Bit 0 nach rechts heraus und mit dem invertierten Carry-Flag zuerst, wieder über das Bit 31 in den Speicherbereich hinein. Der invertierte Wert des Carry-Flags hat jetzt die Position 32 - **n**.

Bei einem Eingang **n** = 0 findet kein Rotieren statt und das Carry-Flag bleibt unverändert.

Bei einem Eingang **n** > 32 kann das gleiche Ergebnis mit einer Zahl **n** < 32 erreicht werden: z.B. **n** = 33 gleiches Ergebnis wie **n** = 0; **n** = 34 gleiches Ergebnis wie **n** = 1.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F127_DRCR (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
d	ANY32	32-Bit-Datenbereich
n	INT	Anzahl der Bits, die gedreht werden (Bereich: 0 bis 255)

Operanden

Für	Merker				T/C		Register			Konstante
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9009	%MX0.900.9	kurzzeitig	▪ das Bit an der Position n - 1 den Wert 1 hat.

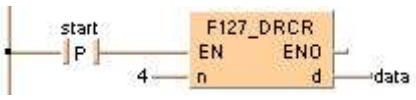
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data	DWORD	16#1234ABCD	result: after a 0->1 leading edge of start: 16#A1234ABC
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Hier wurde der Funktion am Eingang n eine Konstante (4) übergeben. Stattdessen können Sie im POE-Kopf auch eine Variable deklarieren.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

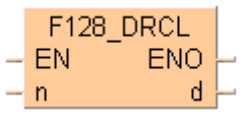
```
IF DF(start) THEN
    F127_DRCR( n:= 4,
         d=> data);
END_IF;
```

F128_DRCL

32-Bit-Linksrotieren mit Carry-Flag

Erklärung

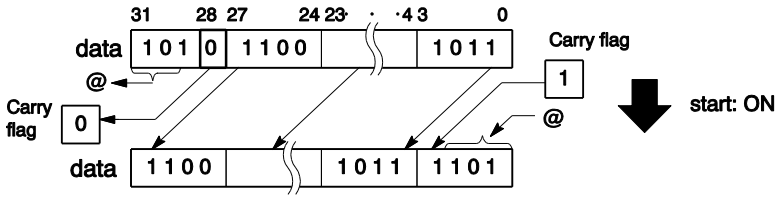
Die Funktion rotiert den Wert am Ausgang **d** mit dem Carry-Flag nach links. Die Anzahl der Bits, um die der Wert am Ausgang **d** nach links rotiert werden soll, wird mit dem Wert am Eingang **n** festgelegt. Dieser kann zwischen 0 und 255 liegen (nur das niederwertige Byte von **n** ist wirksam).



Der Bitwert an der Bit-Position 32 -**n** wird in das Carry-Flag geschrieben. Die Funktion schiebt **n**-Bits über das Bit 31 (MSB) nach links heraus und mit dem invertierten Carry-Flag zuerst, wieder über das Bit 0 (LSB) in den Speicherbereich hinein. Der invertierte Wert des Carry-Flags hat jetzt die Position **n** - 1.

Bei einem Eingang **n** = 0 findet kein Rotieren statt und das Carry-Flag bleibt unverändert.

Bei einem Eingang **n** > 32 kann das gleiche Ergebnis mit einer Zahl **n** < 32 erreicht werden: z.B. **n** = 33 gleiches Ergebnis wie **n** = 0; **n** = 34 gleiches Ergebnis wie **n** = 1.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F128_DRCL (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
d	ANY32	32-Bit-Bereich
n	INT	Anzahl der Bits, die gedreht werden (Bereich: 0 bis 255)

Operanden

Für	Merker				T/C		Register			Konstante
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9009	%MX0.900.9	kurzzeitig	▪ das Bit an der Position 32 - n den Wert 1 hat.

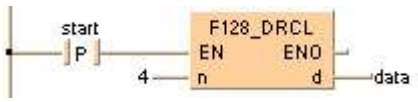
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data	DWORD	16#1234ABCD	result: after a 0->1 leading edge of start: 16#234ABCD0
2	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Hier wurde der Funktion am Eingang n eine Konstante (4) übergeben. Stattdessen können Sie im POE-Kopf auch eine Variable deklarieren.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

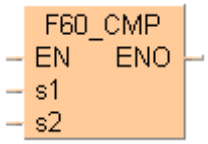
```
IF DF(start) THEN
    F128_DRCL( n:= 4,
        d=> data);
END_IF;
```

Kapitel 18

Vergleichsfunktionen

F60_CMP 16-Bit-Vergleich

Erklärung Der 16-Bit-Wert des Speicherregisters oder eine Konstante **s1** und der 16-Bit-Wert des Speicherregisters oder eine Konstante **s2** werden miteinander verglichen. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis des Vergleichs wird in die Sondermerker R9009 und R900A bis R900C geschrieben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden der Vergleichsbefehle. Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

Daten	Vergleich zwischen S1 und S2	Merker			
		R900A (>Merker)	R900B (=Merker)	R900C (<Merker)	R9009 (Carry-Flag)
16-Bit-Daten mit Vorzeichen	s1<s2	AUS	AUS	AN	#
	s1=s2	AUS	AN	AUS	AUS
	s1>s2	AN	AUS	AUS	#
16-Bit-Daten ohne Vorzeichen	s1<s2	#	AUS	#	AN
	s1=s2	AUS	AN	AUS	AUS
	s1>s2	#	AUS	#	AUS

Zustand wechselt je nach Bedingung

SPS-Typen **Verfügbarkeit von F60_CMP (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	INT, WORD	16-Bit-Bereich oder äquivalente 16-Bit-Konstante zum Vergleich
s2	INT, WORD	16-Bit-Bereich oder äquivalente 16-Bit-Konstante zum Vergleich

Die Variablen **s1** und **s2** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

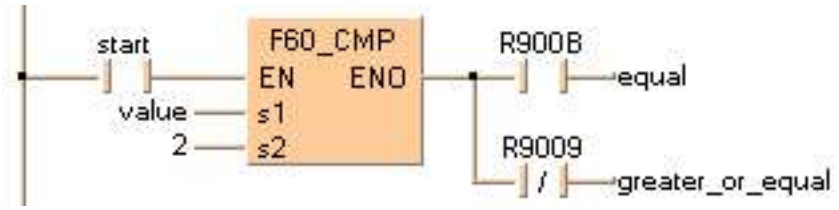
Beispiel In diesem Beispiel wird die Funktion F60_CMP im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value	INT	5	
2	VAR	equal	BOOL	FALSE	set to TRUE depending on the status of
3	VAR	greater_or_equal	BOOL	FALSE	set not to TRUE depending on the status of R9009 (carry flag)
4	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

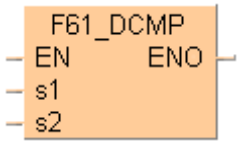


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
equal := FALSE;
greater_or_equal := FALSE;
IF start THEN
    F60_CMP(value, 2);
    IF R900B THEN
        equal := TRUE;
    END_IF;
    IF NOT(R9009) THEN
        greater_or_equal := TRUE;
    END_IF;
END_IF;
```

F61_DCMP 32-Bit-Vergleich

Erklärung Der 32-Bit-Wert des Speicherregisters s1 oder eine Konstante **s1** und der 32-Bit-Wert des Speicherregisters **s2** oder eine Konstante **s2** werden miteinander verglichen. Dazu muss der Trigger **EN** auf EIN gesetzt sein. Das Ergebnis des Vergleichs wird in die Sondermerker R9009 und R900A bis R900C geschrieben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden der Vergleichsbefehle. Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

Daten	Vergleich zwischen s1 und s2	Merker			
		R900A (>flag)	R900B (= Merker)	R900C (< Merker)	R9009 (Carry-Flag)
16-Bit-Daten mit Vorzeichen	s1<s2	AUS	AUS	EIN	#
	s1=s2	AUS	EIN	AUS	AUS
	s1>s2	EIN	AUS	AUS	#
16-Bit-Daten ohne Vorzeichen	s1<s2	#	AUS	#	EIN
	s1=s2	AUS	EIN	AUS	AUS
	s1>s2	#	AUS	#	AUS

Zustand wechselt je nach Bedingung

SPS-Typen **Verfügbarkeit von F61_DCMP (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	32-Bit-Bereich oder äquivalente 32-Bit-Konstante zum Vergleich
s2		32-Bit-Bereich oder äquivalente 32-Bit-Konstante zum Vergleich

Die Variablen **s1** und **s2** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.

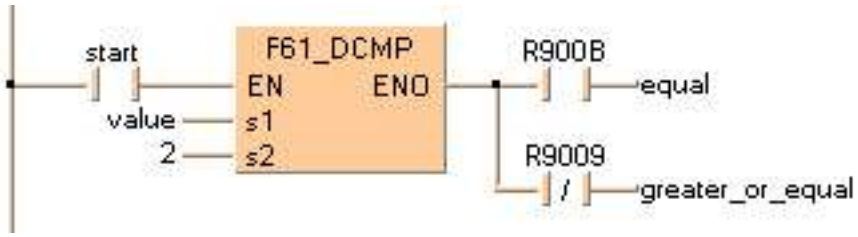
Beispiel In diesem Beispiel wird die Funktion F61_DCMP im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value	DINT	5	
2	VAR	equal	BOOL	FALSE	set to TRUE depending on
3	VAR	greater_or_equal	BOOL	FALSE	set not to TRUE depending on the status of R9009 (carry flag)
4	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



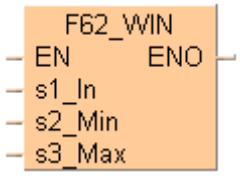
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
equal := FALSE;
greater_or_equal := FALSE;
IF start THEN
    F61_DCMP(value, 2);
    IF R900B THEN
        equal := TRUE;
    END_IF;
    IF NOT(R9009) THEN
        greater_or_equal := TRUE;
    END_IF;
END_IF;
```

F62_WIN

16-Bit-Vergleich Bereich

Erklärung Der 16-Bit-Wert des Speicherregisters oder eine Konstante **s1** wird mit einem Wertebereich verglichen, der durch **s2** und **s3** angegeben wird, wenn der **EN**-Eingang auf TRUE gesetzt ist. Dieser Befehl prüft, ob **s1** im Datenbereich zwischen **s2** (unterer Grenzwert des Wertebereichs) und **s3** (oberer Grenzwert des Wertebereichs) liegt, größer als **s3** oder kleiner als **s2** ist. Für den Vergleich lassen sich +/-Zeichen verwenden. 16-Bit Dezimalzahlen werden mit Vorzeichen verglichen. Ihr Wertebereich reicht von -32768 bis +32767. BCD-codierte 16-Bit Zahlen besitzen einen Wertebereich von 0 bis 7999. Das Ergebnis des Vergleichs steht in den Sondermerkern R900A, R900B und R900C.



Vergleich zwischen s1, s2 und s3	Merker		
	R900A (> Merker)	R900B (= Merker)	R900C (< Merker)
s1<s2	AUS	AUS	EIN
s2 ≤ s1 ≤ s3	AUS	EIN	AUS
s1 > s3	EIN	AUS	AUS

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F62_WIN (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	16-Bit-Bereich oder äquivalente 16-Bit-Konstante zum Vergleich
s2		unterer Grenzwert, 16-Bit-Bereich oder äquivalente 16-Bit-Konstante
s3		oberer Grenzwert, 16-Bit-Bereich oder äquivalente 16-Bit-Konstante

Die Variablen **s1, s2** und **s3** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

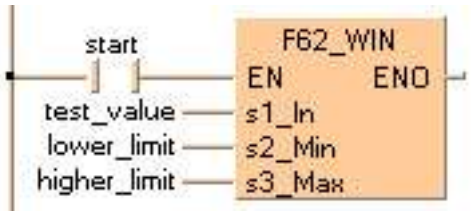
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	test_value	INT	35	this value will be compared with the data band
2	VAR	lower_limit	INT	0	lower limit
3	VAR	higher_limit	INT	100	higher limit

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



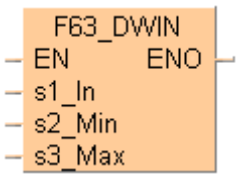
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F62_WIN( s1_In:= test_value,
             s2_Min:= lower_limit,
             s3_Max:= higher_limit);
END_IF;
```

F63_DWIN

32-Bit-Vergleich Bereich

Erklärung Der 32-Bit-Wert des Speicherregisters oder eine Konstante **s1** wird mit einem Wertebereich verglichen, der durch **s2** und **s3** angegeben wird, wenn der **EN**-Eingang auf TRUE gesetzt ist. Dieser Befehl prüft, ob **s1** im Datenbereich zwischen **s2** (unterer Grenzwert des Wertebereichs) und **s3** (oberer Grenzwert des Wertebereichs) liegt, größer als **s3** oder kleiner als **s2** ist. Für den Vergleich lassen sich +/-Zeichen verwenden. 32-Bit Dezimalzahlen werden mit Vorzeichen verglichen. Ihr Wertebereich reicht von -32768 bis +32767. BCD-codierte 16-Bit Zahlen besitzen einen Wertebereich von 0 bis 79999999. Das Ergebnis des Vergleichs steht in den Sondermerkern R900A, R900B und R900C.



Vergleich zwischen s1, s2 und s3	Merker		
	R900A (> Merker)	R900B (= Merker)	R900C (< Merker)
s1<s2	AUS	AUS	EIN
s2 ≤ s1 ≤ s3	AUS	EIN	AUS
s1 > s3	EIN	AUS	AUS

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F63_DWIN (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	32-Bit-Bereich oder äquivalente 32-Bit-Konstante zum Vergleich
s2		unterer Grenzwert, 32-Bit-Bereich oder äquivalente 32-Bit-Konstante
s3		oberer Grenzwert, 32-Bit-Bereich oder äquivalente 32-Bit-Konstante

Die Variablen **s1, s2** und **s3** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.

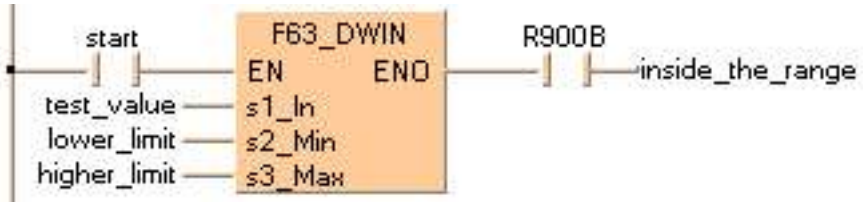
Beispiel In diesem Beispiel wird die Funktion F63_DWIN im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	test_value	DINT	35	this value will be compared with the data band
2	VAR	lower_limit	DINT	0	lower limit
3	VAR	higher_limit	DINT	100	higher limit
4	VAR	inside_the_range	BOOL	FALSE	

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



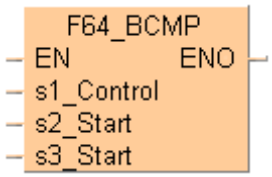
ST

```
inside_the_range := FALSE;
IF start THEN
    F63_DWIN( s1_In := test_value,
              s2_Min := lower_limit,
              s3_Max := higher_limit);
    IF R900B THEN
        inside_the_range := TRUE;
    END_IF;
END_IF;
```

F64_BCMP

Blockdaten-Vergleich

Erklärung Vergleicht zwei Daten miteinander. Die Daten des Operanden **s2** werden mit denen des Operanden **s3** im Hinblick auf die Daten des Operanden **s1** verglichen, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Spezifikationen für s1:

16#	1	0	0 4
	⇑	⇑	⇑
	A	B	C

A = Startbyteposition des Datenblocks von **s3**
1: Vergleich vom höheren Byte
0: Vergleich vom niederen Byte

B = Startbyteposition des Datenblocks von **s2**
1: Vergleich vom höheren Byte
0: Vergleich vom niederen Byte

C = Anzahl der Bytes, die verglichen werden sollen
Bereich: 16#01 bis 16#99 (BCD)

Das Ergebnis des Vergleichs steht in den Sondermerkern R900B. Wenn **s2 = s3**, ist der Sondermerker im Status TRUE.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F64_BCMP (s. S. 1191)**

☞ **Der Merker R900B, der für den Vergleichsbefehl verwendet wird, wird immer dann zurückgesetzt, wenn ein Vergleichsbefehl ausgeführt wird. Das Programm, das R900B verwendet, sollte deshalb direkt auf F64_BCMP folgen.**

Datentypen

Variable	Datentyp	Funktion
s1	WORD	Kontrollcode, der die Bytepositionen und die Anzahl der Bytes, die verglichen werden, spezifiziert
s2	ANY16	16-Bit-Anfangsbereich, der mit s3 verglichen wird
s3		16-Bit-Anfangsbereich, der mit s2 verglichen wird

Die Variablen **s2** and **s3** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
s2, s3:	WX	WY	WR	WL	SV	EV	DT	LD	FL	-

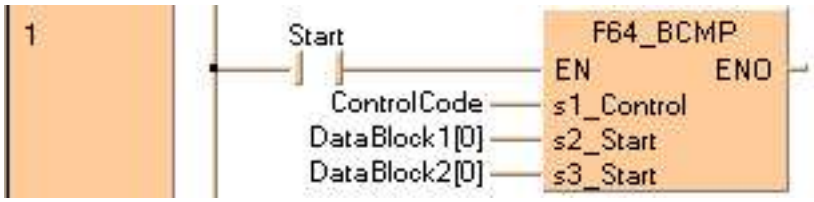
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	ControlCode	WORD	16#1106	s2 starting from upper byte
2	VAR	DataBlock1	ARRAY [0..5] OF INT	[6(1234)]	s3 starting from upper byte
3	VAR	DataBlock2	ARRAY [0..5] OF INT	[6(1234)]	compare 6 bytes
4	VAR	equal_block	BOOL	FALSE	

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



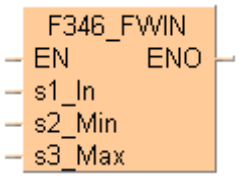
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F64_BCMP( s1_Control:= ControlCode,
          s2_Start:= DataBlock1[0],
          s3_Start:= DataBlock2[0]);
END_IF;
```

F346_FWIN

Fließkommawert-Bereichsvergleich

Erklärung Die Funktion vergleicht einen Bereich, dessen untere und obere Grenze den Eingängen **s2** und **s3** übergeben werden mit einem Wert, der am Eingang **s1** anliegt. Das Funktionsergebnis wird wie folgt zurückgegeben:



- Ist der Wert an **s1** kleiner als der Wert an **s2** (untere Grenze des Bereichs), wird der < Merker (R900C) auf TRUE gesetzt.
- Ist der Wert an **s1** größer als der Wert an **s3** (obere Grenze des Bereichs), wird der > Merker (R900A) auf TRUE gesetzt. Der < Merker (R900C) und der = Merker (R900B) werden auf FALSE gesetzt.
- Ist der Wert an **s1** im Bereich der Werte **s2** und **s3**, wird der = Merker (R900B) kurzzeitig gesetzt. Der < Merker (R900C) und der > Merker (R900A) werden auf FALSE gesetzt.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F346_FWIN (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s1	REAL	Fließkommawert, der mit s2 und s3 verglichen wird
s2	REAL	Untere Grenze
s3	REAL	Obere Grenze

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Werte an den Eingängen s1, s2 und s3 keine REAL-Zahlen sind oder der Wert an s2 > s3 ist.
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

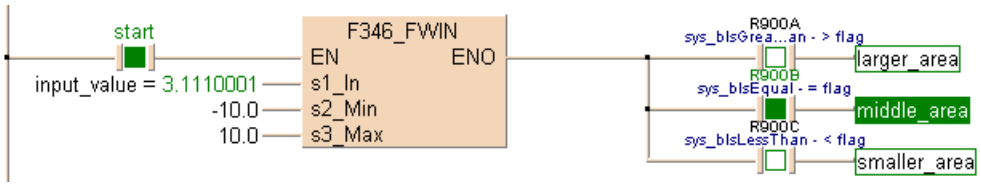
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	REAL	3.111	
2	VAR	larger_area	BOOL	FALSE	result: here FALSE
3	VAR	middle_area	BOOL	FALSE	result: here TRUE
4	VAR	smaller_area	BOOL	FALSE	result: here FALSE

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf An den Eingängen **s2_Min** (untere Grenze) und **s3_Max** (obere Grenze) stehen die Konstanten -10.0 und 10.0. Statt dessen können Sie auch im POE-Kopf zwei Variablen deklarieren. Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Werte der Register R900A (> Vergleichsmerker), R900B (= Vergleichsmerker) und R900C (< Vergleichsmerker) werden den Variablen **larger_area, middle_area** und **smaller_area** übergeben. Da sich der **input_value** = 3.111 im Bereich der Grenzwerte -10.0 bis 10.0 befindet, wird der = Merker und damit die Variable **middle_area** auf TRUE gesetzt.

KOP



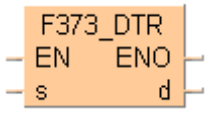
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
input_value:=3.111;
IF start THEN
    F346_FWIN( s1_In:= input_value , s2_Min:= -10.0 , s3_Max:= 10.0 );
END_IF;(* -10.0 =lower limit, 10.0 upper limit *)

IF R900A THEN
    larger_area:=TRUE;
END_IF;
IF R900B THEN
    middle_area:=TRUE;
END_IF;
IF R900C THEN
    smaller_area:=TRUE;
END_IF;
```

F373_DTR

F355_PID_DUT

Erklärung Die Funktion erkennt Änderungen eines Wertes am Eingang s indem sie ihn mit dem vorherigen Wert, der am Ausgang d liegt, vergleicht. Wenn der neue Eingangswert an s nicht mit dem alten Wert übereinstimmt, gibt die Funktion den neuen Wert am Ausgang d aus. Gleichzeitig wird als Zeichen der Änderung das Carry-Flag (R9009) gesetzt.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F373_DTR (s. S. 1191)**

☞ **Der Zustand des Carry-Flags wird nach jeder Funktionsausführung erneuert. Deshalb sollten Programme, die das Carry-Flag nutzen, es unmittelbar nach einer Funktionsausführung von F373_DTR weiterverarbeiten.**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	16-Bit-Datenbereich, in dem Änderungen geprüft werden
d		Speicherbereich für Daten der vorherigen Ausführung

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9009	%MX0.900.9	auf TRUE	▪ wenn der Eingangswert an s sich im Vergleich zum vorherigen Wert verändert hat.

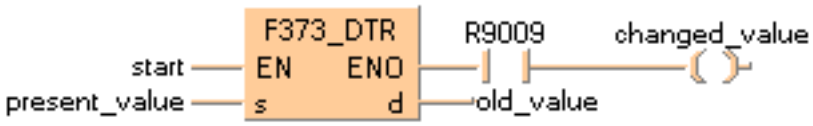
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	present_value	INT	0	value whose status should be monitored
2	VAR	old_value	INT	0	dummy value for storing the former present value
3	VAR	changed_value	BOOL	FALSE	signal for changing present value

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Falls sich der Eingangswert **present_value** im Vergleich zum Ausgangswert **old_value** verändert hat, wird das Carry-Flag R9009 gesetzt. Der Zustand des Carry-Flags wird der Variablen **changed_value** übergeben.

KOP



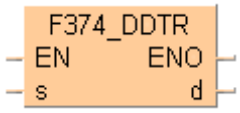
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F373_DTR(present_value, old_value);
    IF R9009 THEN
        changed_value:=TRUE;
    END_IF;
END_IF;
```

F374_DDTR

32-Bit-Daten Änderungserkennung

Erklärung Die Funktion erkennt Änderungen eines Wertes am Eingang s indem sie ihn mit dem vorherigen Wert, der am Ausgang d liegt, vergleicht. Wenn der neue Eingangswert an s nicht mit dem alten Wert übereinstimmt, gibt die Funktion den neuen Wert am Ausgang d aus. Gleichzeitig wird als Zeichen der Änderung das Carry-Flag (R9009) gesetzt.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F374_DDTR (s. S. 1191)**

☞ **Der Zustand des Carry-Flags wird nach jeder Funktionsausführung erneuert. Deshalb sollten Programme, die das Carry-Flag nutzen, es unmittelbar nach einer Funktionsausführung von F374_DDTR weiterverarbeiten.**

Datentypen

Variable	Datentyp	Funktion
s	ANY32	32-Bit-Datenbereich, in dem Änderungen geprüft werden
d		32-Bit-Speicherbereich für Daten der vorherigen Ausführung

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9009	%MX0.900.9	auf TRUE	▪ wenn der Eingangswert an s sich im Vergleich zum vorherigen Wert verändert hat.

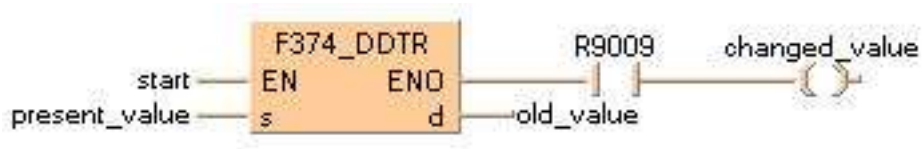
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	present_value	DINT	0	value whose status should be monitored
2	VAR	old_value	DINT	0	dummy value for storing the former present value
3	VAR	changed_value	BOOL	FALSE	signal for changing present value

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Falls sich der Eingangswert **present_value** im Vergleich zum Ausgangswert **old_value** verändert hat, wird das Carry-Flag R9009 gesetzt. Der Zustand des Carry-Flags wird der Variablen **changed_value** übergeben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F374_DDTR(present_value, old_value);
    IF R9009 THEN
        changed_value:=TRUE;
    END_IF;
END_IF;
```

18.1 Weitere Vergleichsbefehle

Weitere Informationen zu folgenden FP-Befehlen entnehmen Sie bitte den Standard-Operatoren:

ST=	AN=	OR=	STD=	AND=	ORD=
ST<>	AN<>	OR<>	STD<>	AND<>	ORD<>
ST>	AN>	OR>	STD>	AND>	ORD>
ST>=	AN>=	OR>=	STD>=	AND>=	ORD>=
ST<	AN<	OR<	STD<	AND<	ORD<
ST<=	AN<=	OR<=	STD<=	AND<=	ORD<=

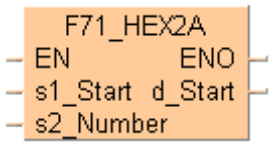
Kapitel 19

Umwandlungsfunktionen

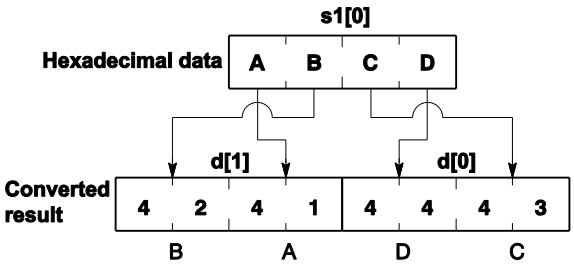
F71_HEX2A

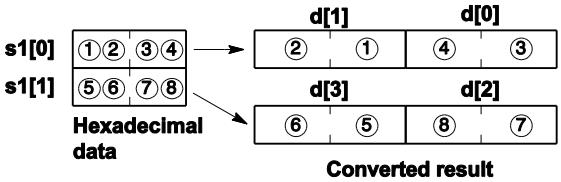
HEX -> ASCII-Umwandlung

Erklärung Die hexadezimalen Daten ab der Anfangsadresse **s1_Start** werden byteweise in hexadezimale ASCII-Daten konvertiert, wenn der Trigger **EN** den Status TRUE hat. Die Anzahl der zu konvertierenden Bytes wird in **s2_Number** festgelegt. Das Ergebnis der Umwandlung wird ab der hexadezimalen Anfangsadresse **d_Start** im Zieldatenbereich gespeichert. Im ASCII-Code werden für die Darstellung eines hexadezimalen Zeichens 8 Bit (= 1 Byte) benötigt. Nach der ASCII-Umwandlung ist der Zieldatenbereich daher doppelt so groß wie der Quelldatenbereich.



Die beiden Zeichen, die ein Byte darstellen, werden bei der Speicherung vertauscht. Zwei Bytes werden als einzelnes Datensegment konvertiert.





ASCII HEX-Codes zur Darstellung hexadezimaler Zeichen:

Hexadezimalzahl	ASCII-HEX-Code
0	16#30
1	16#31
2	16#32
3	16#33
4	16#34
5	16#35
6	16#36
7	16#37
8	16#38
9	16#39
A	16#41
B	16#42
C	16#43
D	16#44
E	16#45
F	16#46

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F71_HEX2A (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1_Start	ANY16	16-Bit-Anfangsadresse für Hexadezimalzahl (Quelle)
s2_Number	INT	Legt die Anzahl der zu konvertierenden Bytes des Quelldatenbereichs fest.
d_Start	ANY16	16-Bit-Anfangsadresse für das Speichern des ASCII-Codes (Ziel)

Operanden

Für	Relais				T/C		Register			Konstante
s1_Start	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2_Number	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez. oder hex.
d_Start	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	dauerhaft	▪ die durch **s2_Number** festgelegte Bytezahl den in **s1_Start** festgelegten Speicherplatz überschreitet. ▪ die Berechnungsergebnisse den in **d_Start** festgelegten Speicherplatz überschreiten. ▪ der in **s2_Number** festgelegte Wert als "0" erkannt wird.
R9008	%MX0.900.8	kurzzeitig	

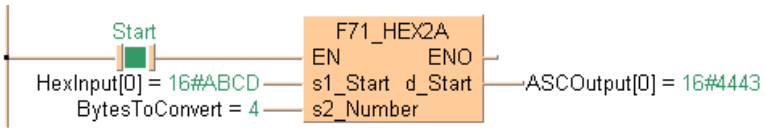
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Start	BOOL	FALSE	
1	VAR	HexInput	ARRAY [0..1] OF WORD	[16#abcd,16#ef]	
2	VAR	BytesToConvert	INT	4	3 bytes will be converted
3	VAR	ASCOutput	ARRAY [0..3] OF WORD	[4(0)]	3 bytes hex. require 6 bytes for ASCII code ARRAY[3] will be filled with two zero characters = 16#3030
4	VAR				

Rumpf Wenn die Variable **Start** gesetzt (TRUE) ist, wird die in **BytesToConvert** festgelegte Anzahl der Bytes aus **HexInput** in den ASCII-Code konvertiert und in **ASCOutput** gespeichert. Beachten Sie, dass die beiden Zeichen, die ein Byte darstellen, bei der Speicherung untereinander getauscht werden. Ein Monitorkopf zeigt die Hex-Werte, der andere die ASCII-Werte.

KOP



F71_HEX2A_LD	Struktur
Start	2#1 at R90
-HexInput	Structure
[0]	16#ABCD at DT3271
[1]	16#00EF at DT3272
BytesToConvert	4 at DT3273
-ASCOutput	Struktur
[0]	16#4443 at DT3274
[1]	16#4241 at DT3275
[2]	16#4645 at DT3276
[3]	16#3030 at DT3277

F71_HEX2A_LD	Struktur
Start	2#1 at R90
-HexInput	Struktur
[0]	16#ABCD at DT3271
[1]	16#00EF at DT3272
BytesToConvert	4 at DT3273
-ASCOutput	Struktur
[0]	'CD' at DT3274
[1]	'AB' at DT3275
[2]	'EF' at DT3276
[3]	'00' at DT3277

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

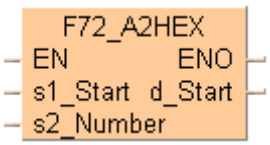
```
IF start THEN
    F71_HEX2A( s1_Start := HexInput[0],
          s2_Number := BytesToConvert,
          d_Start => ASCOutput[0]);
END_IF;
```

F72_A2HEX

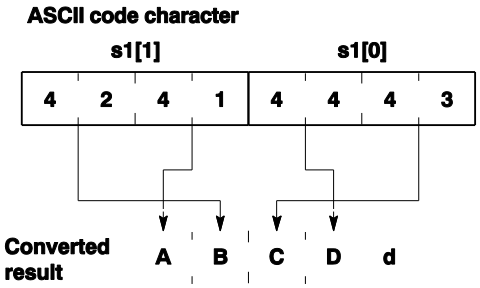
ASCII -> HEX Umwandlung

Erklärung

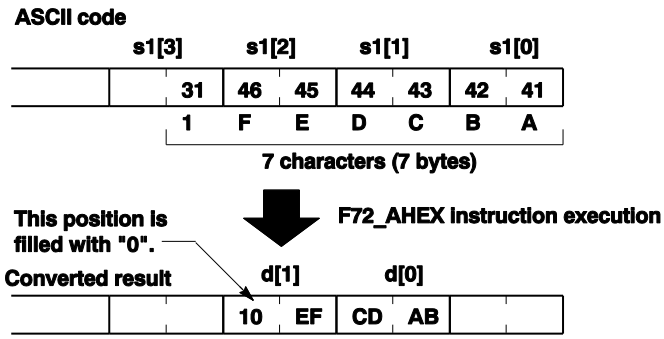
Die hexadezimalen Daten ab der Anfangsadresse **s1_Start** werden byteweise in hexadezimale ASCII-Daten konvertiert, wenn der Trigger **EN** den Status ON hat. **s2_Number** legt die Anzahl der zu konvertierenden Bytes (Anzahl der Zeichen) fest. Das Ergebnis der Konvertierung wird ab der Anfangsadresse **d_Start** im Zieldatenbereich gespeichert. Der ASCII-Code benötigt 8 Bit, um ein hexadezimales Zeichen auszudrücken. Entsprechend der Konvertierungsvorschrift ist der Zieldatenbereich halb so groß wie der Quelldatenbereich.



Die Werte zweier ASCII-Code-Zeichen wird in zwei numerische Ziffern für ein Wort konvertiert. Dabei werden die Zeichen niedrigwertiger und höherwertiger Bytes miteinander getauscht. Vier Zeichen werden in ein Datensegment konvertiert.



Das Konvertierungsergebnis wird in Bytes gespeichert. Soll eine ungerade Anzahl von Zeichen konvertiert werden, wird "0" für die Bits 0 bis 3 des letzten Wertes (Byte) des Konvertierungsergebnisses eingegeben. Konvertierung einer ungeraden Anzahl von Werten:



Hexadezimalzeichen und ASCII-Codes:

ASCII-HEX-Code	Hexadezimalzahl
16#30	0
16#31	1
16#32	2
16#33	3
16#34	4
16#35	5
16#36	6
16#37	7
16#38	8
16#39	9
16#41	A
16#42	B
16#43	C
16#44	D
16#45	E
16#46	F

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F72_A2HEX (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1_Start	WORD	Anfangsadresse für ASCII_Code (Quelle)
s2_Number	INT	Legt die Anzahl der zu konvertierenden Bytes des Quelldatenbereichs fest.
d_Start	ANY16	Legt die Anzahl der zu konvertierenden Bytes des Quelldatenbereichs fest.

Operanden

Für	Merker				T/C		Register			Konstante
s1_Start	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2_Number	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d_Start	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die durch **s2_Number** festgelegte Bytezahl den in **s1_Start** festgelegten Speicherplatz überschreitet. ▪ die berechneten Ergebnisse den in **d_Start** festgelegten Speicherplatz überschreiten. ▪ der in **s2_Number** festgelegte Wert als "0" erkannt wird. ▪ ein ASCII-Code, aber keine Hexadezimalzahl (0 bis F) festgelegt wurde.
R9008	%MX0.900.8	kurzzeitig	

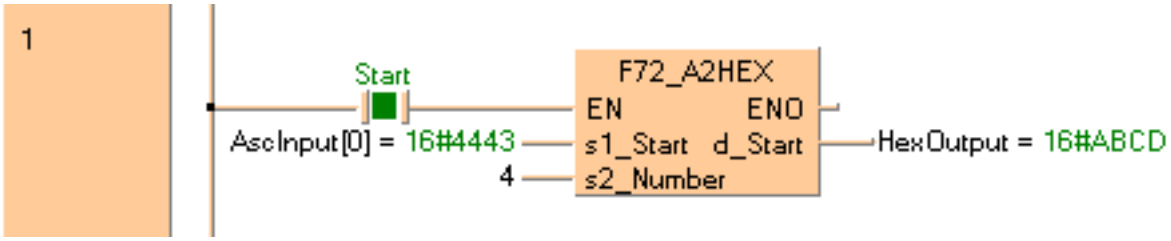
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Start	BOOL	FALSE
1	VAR	AscInput	ARRAY [0..1] OF WORD	[16#4443,16#4241]
2	VAR	HexOutput	WORD	0

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. In diesem Beispiel wurde der Wert für **s2**, d.h. die Anzahl der Bytes, die vom ASCII-Code in den Hexadezimal-Code konvertiert werden sollen, direkt am Kontakt eingegeben.

KOP



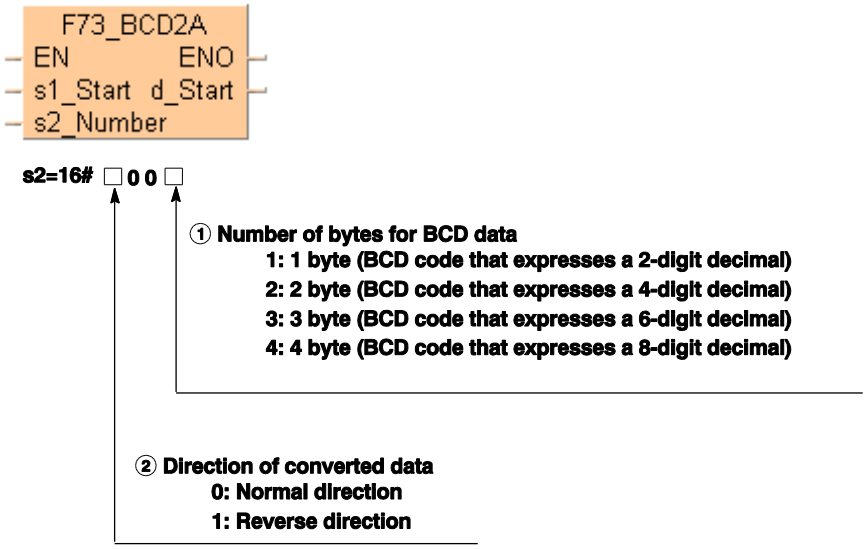
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F72_A2HEX( s1_Start := AscInput[0],
               s2_Number := 4,
               d_Start => HexOutput);
```

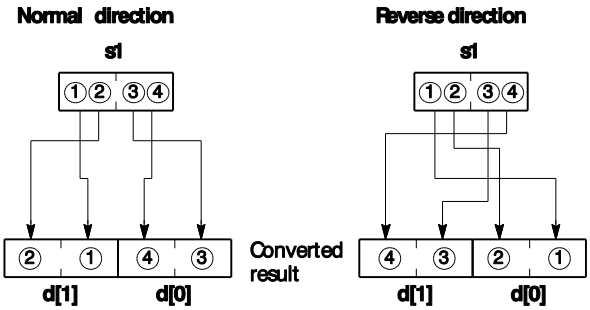
F73_BCD2A

BCD -> ASCII Umwandlung

Erklärung Wenn der Trigger **EN** den Status ON hat, werden die BCD-codierten Daten ab der Anfangsadresse **s1** in den ASCII-Code, der die equivalenten Dezimalen entsprechend dem in **s2** festgelegten Inhalt wiedergibt, konvertiert. **s2** legt die Anzahl der Bytes des Quelldatenbereichs und die Richtung der Konvertierungsdaten (Normal/Rückwärtsrichtung) fest.



Die beiden Zeichen, die ein Byte darstellen, werden bei der Speicherung untereinander getauscht. Zwei Bytes werden als ein Datensegment konvertiert:



Die Konvertierungsergebnisse werden im Speicherregister **d** abgelegt. Der ASCII-Code benötigt 8 Bit (1 Byte), um ein BCD-Zeichen auszudrücken. Nach der ASCII-Konvertierung ist der Zieldatenbereich doppelt so groß wie der BCD-Quelldatenbereich.

ASCII HEX-Code zur Darstellung der BCD-Zeichen:

BCD-Zeichen	ASCII-HEX-Code
0	H30
1	H31
2	H32
3	H33
4	H34
5	H35
6	H36
7	H37
8	H38
9	H39

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die

Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F73_BCD2A (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1	WORD	Anfangsadresse für BCD-Daten (Quelle)
s2	ANY16	Legt die Anzahl der zu konvertierenden Bytes des Quelldatenbereichs und die Anordnung fest.
d	WORD	Anfangsadresse für das Speichern der konvertierten Daten (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s1	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die in s1 festgelegten Werte keine BCD-Werte sind. ▪ die durch s2 festgelegte Bytezahl den in s1 festgelegten Speicherplatz überschreitet. ▪ die berechneten Ergebnisse den in d festgelegten Speicherplatz überschreiten. ▪ der in s2 festgelegte Wert als "0" erkannt wird. ▪ die Anzahl der in s2 festgelegten Bytes größer als 16#4 ist.
R9008	%MX0.900.8	kurzzeitig	

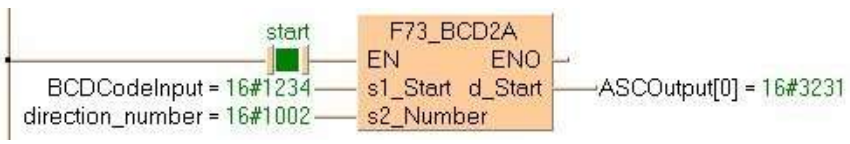
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	BCDCodeInput	WORD	16#1234	
2	VAR	direction_number	WORD	16#1002	specifies the operation:
3	VAR	ASCOutput	ARRAY [0..1] OF WORD	[2(0)]	result after a 0->1 leading

Rumpf Wenn die Variable **Enable** auf TRUE gesetzt wird, wird die Funktion ausgeführt. In diesem Beispiel legt die Variable **direction_number** fest, dass von der Eingangsvariablen **BCDCodeInput** 2 Bytes in umgekehrter Richtung konvertiert und in **ASCOutput** gespeichert werden.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

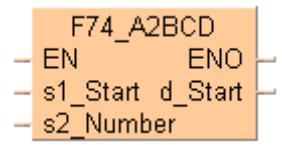
```
IF start THEN
    F73_BCD2A( s1_Start:= BCDCodeInput ,
          s2_Number:= direction_number ,
          d_Start=> ASCOutput[0] );
END_IF;
```

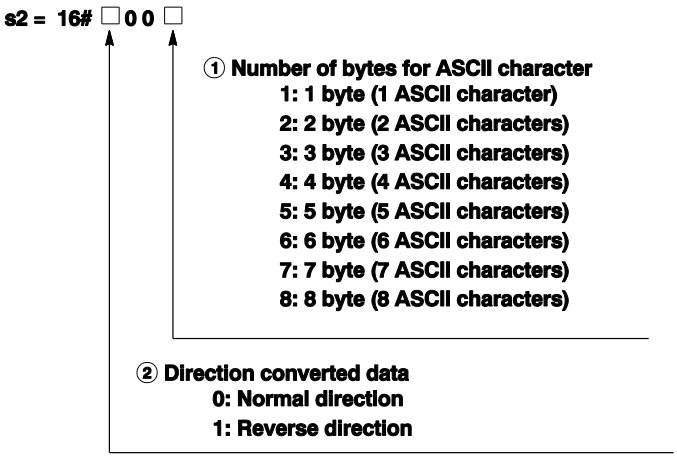
F74_A2BCD

ASCII -> BCD Umwandlung

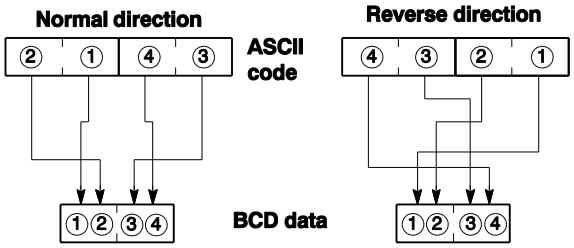
Erklärung

Der hexadezimale ASCII-Code, beginnend von der Anfangsadresse **s1** wird in BCD-Daten konvertiert, wenn der Trigger **EN** den Status ON hat. **s2** legt die Anzahl der zu konvertierenden Bytes und die Richtung der konvertierten Quellcodedaten fest.



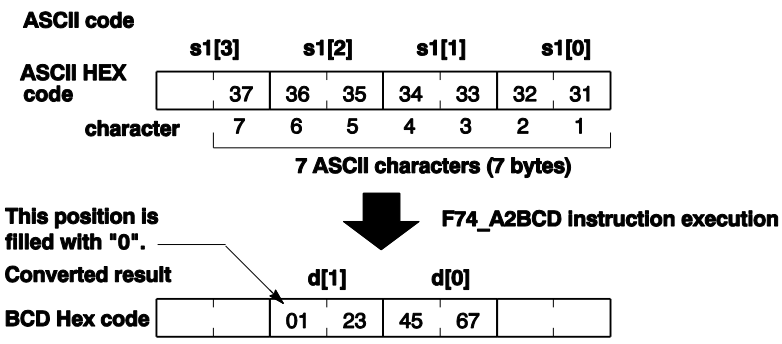


Vier Zeichen werden als ein Datensegment konvertiert:



Das Ergebnis der Konvertierung wird ab der Anfangsadresse **d** im Zieldatenbereich in Bytes gespeichert. Der ASCII-Code benötigt 8 Bit (1 Byte), um ein BCD-Zeichen auszudrücken. Entsprechend der Konvertierungsvorschrift ist der Zieldatenbereich halb so groß wie der Quelldatenbereich.

Wenn eine ungerade Anzahl von Zeichen konvertiert wird, wird "0" für die Bit-Position 0 bis 3 des letzten Wertes (Byte) der Konvertierungsergebnisse eingegeben, wenn die Daten in Vorwärtsrichtung durchlaufen werden. Beim rückwärtigen Durchlaufen der Daten wird "0" für die Bit-Position 4 bis 7 eingegeben.



ASCII HEX-Code zur Darstellung der BCD-Zeichen:

BCD-Zeichen	ASCII-HEX-Code
0	H30
1	H31
2	H32
3	H33
4	H34
5	H35
6	H36
7	H37
8	H38
9	H39

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F74_A2BCD (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1	WORD	Anfangsadresse für die Speicherung des ASCII-Code (Quelle)
s2	ANY16	Legt die Anzahl der zu konvertierenden Bytes des Quelldatenbereichs und die Anordnung fest.
d	WORD	Anfangsadresse für das Speichern der konvertierten Daten (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s1	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ ein ASCII-Code ohne Übereinstimmung mit Dezimalzahlen (0 bis 9) festgelegt wurde. ▪ die durch **s2** festgelegte Bytezahl den in **s1** festgelegten Speicherplatz überschreitet. ▪ die berechneten Ergebnisse den in **d** festgelegten Speicherplatz überschreiten. ▪ der in **s2** festgelegte Wert als "0" erkannt wird. ▪ die Anzahl der in **s2** festgelegten ASCII-Zeichen größer als 16#8 ist.
R9008	%MX0.900.8	kurzzeitig	

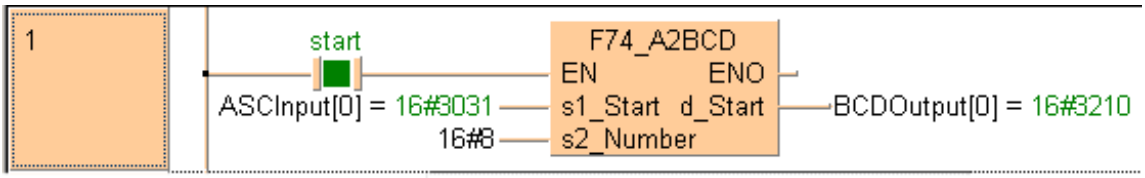
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	ASCInput	ARRAY [0..3] OF WORD	[16#3031,16#3233,16#3435,16#3637]
2	VAR	BCDOutput	ARRAY [0..1] OF WORD	[2(0)]

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Für die Variable in **s1** muss ein ARRAY mit mehr als 4 Elementen definiert werden, weil 8 ASCII-Zeichen 8 Bytes Speicherplatz benötigen und die Funktion nicht mehr als 8 Bytes konvertieren kann. In diesem Beispiel wird der Wert für **s2** wird direkt am Kontakt eingegeben.

KOP



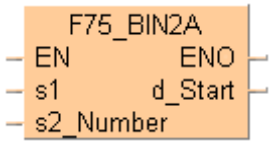
-F74_A2BCD_LD	Struktur
start	2#1 an R250
-ASCInput	Struktur
[0]	16#3031 an DT1200
[1]	16#3233 an DT1201
[2]	16#3435 an DT1202
[3]	16#3637 an DT1203
-BCDOutput	Struktur
[0]	16#3210 an DT1204
[1]	16#7654 an DT1205

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

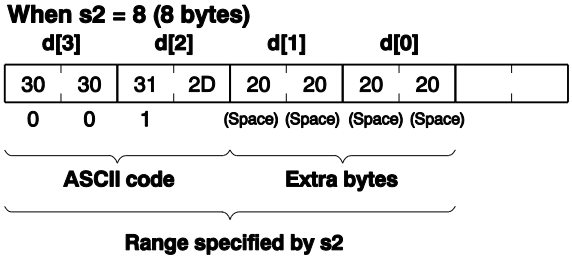
```
IF start THEN
    F74_A2BCD( s1_Start:= ASCInput[0] ,
          s2_Number:= 16#8 ,
          d_Start=> BCDOutput[0] );
END_IF;
```

F75_BIN2A — 16-Bit BIN -> ASCII Umwandlung

Erklärung Der binäre 16-Bit-Wert, der in **s1** festgelegt ist, wird in den hexadezimalen ASCII-Code konvertiert. Das Ergebnis der Konvertierung wird ab der Anfangsadresse **d** im Zieldatenbereich **s2** gespeichert. Die Anzahl der Bytes in Dezimalzahlen wird in **s2** festgelegt. (Diese Festlegung kann nicht mit BCD-Daten vorgenommen werden.)



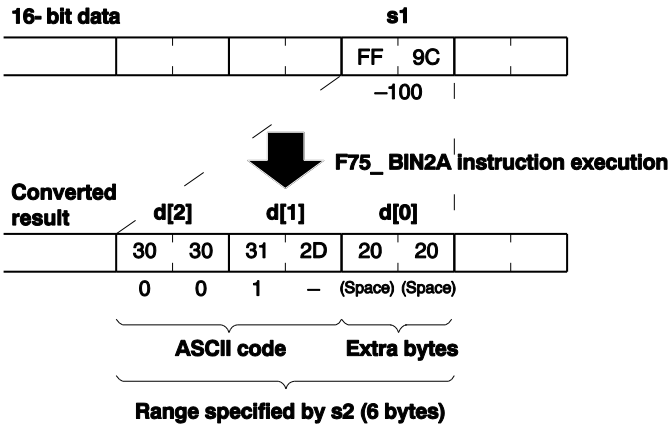
- Wenn eine positive Zahl konvertiert wird, wird das Zeichen "+" nicht konvertiert.
- Wenn eine negative Zahl konvertiert wird, wird das Zeichen "-" ebenfalls in den ASCII-Code (ASCII-HEX-Code: 16#2D) konvertiert.
- Wenn der Speicherbereich **s2** größer ist als durch die konvertierten Daten benötigt, wird der ASCII-Code für "SPACE" (ASCII-HEX-Code: 16#20) in einem Extrabereich gespeichert.
- Die Werte werden in Richtung der Endadresse gespeichert, so dass sich die Position des ASCII-Code in Abhängigkeit von der Größe des Datenspeichers ändern kann.



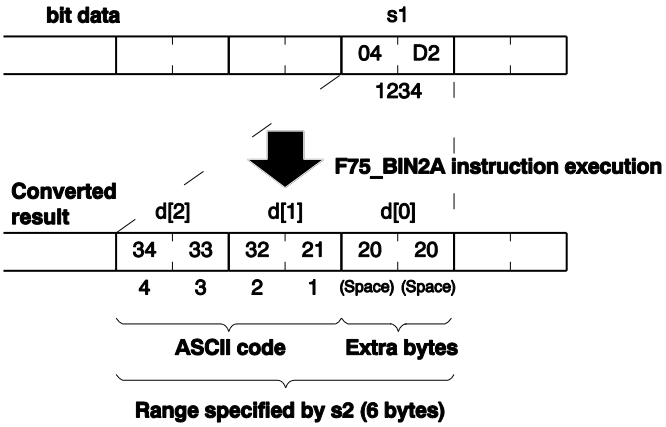
- Wenn die Anzahl der Bytes des ASCII-Codes (inklusive der Minuszeichen) nach der Konvertierung größer ist als die Anzahl der Bytes, die durch **s2** festgelegt wurden, tritt eine Funktionsfehler auf. Stellen Sie sicher, dass das Vorzeichen beachtet wird, wenn das Konvertierungsobjekt für **s2** festgelegt wird.

Die folgenden Abbildungen zeigen die Konvertierung von 16-Bit Dezimaldaten in ASCII-Daten.

Ein negativer Wert wird konvertiert:



Ein positiver Wert wird konvertiert:



Dezimalzeichen für die Darstellung des ASCII-HEX-Codes:

Dezimalzahlen	ASCII-HEX-Code
SPACE	16#20
-	16#2D
0	16#30
1	16#31
2	16#32
3	16#33
4	16#34
5	16#35
6	16#36
7	16#37
8	16#38
9	16#39

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F75_BIN2A (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	16-Bit Bereich, der konvertiert werden soll (Quelle)
s2	INT	Legt die Anzahl der Bytes fest, die die Zielwerte (ASCII-Code) ausdrücken.
d	WORD	16-Bit Bereich für die Speicherung des ASCII-Codes (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die durch **s2** festgelegte Bytezahl den in **d** festgelegten Speicherplatz überschreitet. ▪ der in **s2** festgelegte Wert als "0" erkannt wird. ▪ die berechneten Konvertierungsergebnisse den in **d** festgelegten Speicherplatz überschreiten. ▪ die Anzahl der Bytes des Konvertierungsergebnisses die in **s2** festgelegte Bytezahl überschreiten.
R9008	%MX0.900.8	kurzzeitig	

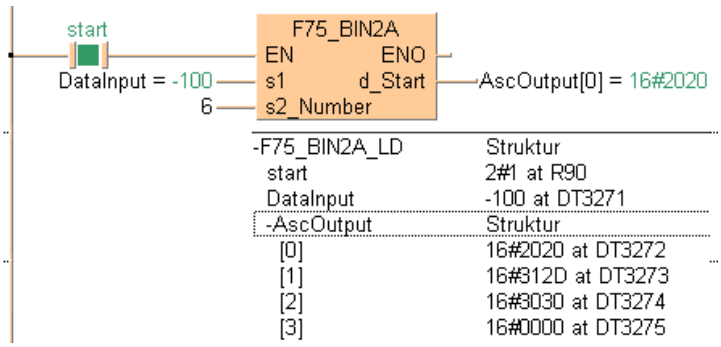
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	DataInput	INT	-100	
2	VAR	AscOutput	ARRAY [0..3] OF WORD	[4(0)]	result after a 0->1 leading edge
3	VAR				from start: 100

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Dieses Programmierbeispiel basiert auf dem oben riebenen Beispiel für die Konvertierung negativer Zahlen. Das Symbol "Werte monitoren" ist sowohl für KOP- als auch für den AWL-Rumpf aktiviert, das Symbol "Monitor Kopf" ist für den KOP-Rumpf aktiviert.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

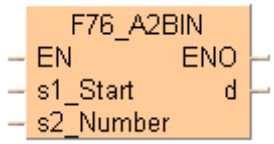
```
IF start THEN
    F75_BIN2A( s1:= DataInput ,
          s2_Number:= 6 ,
          d_Start=> ASCOutput[0] );
END_IF;
```

F76_A2BIN

ASCII -> 16-Bit BIN Umwandlung

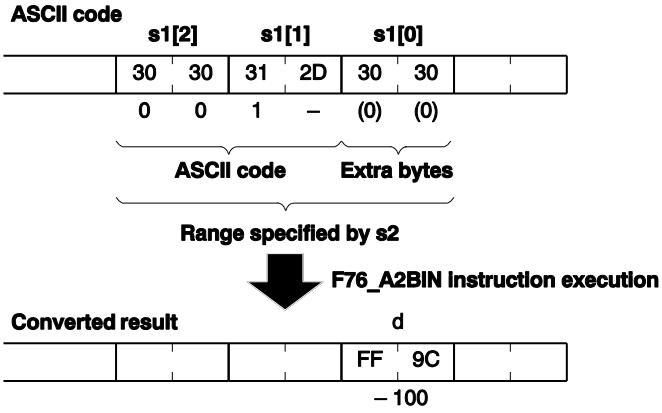
Erklärung

Die hexadezimalen ASCII-Daten ab der Anfangsadresse **s1** werden, wie in **s2** festgelegt, in 16-Bit-Werte konvertiert. Die Konvertierungsergebnisse werden im Speicherregister **d** abgelegt. **s2** legt fest, wie viele Bytes des Quelldatenbereichs konvertiert werden. (Diese Festlegung kann nicht mit BCD-Daten vorgenommen werden.)

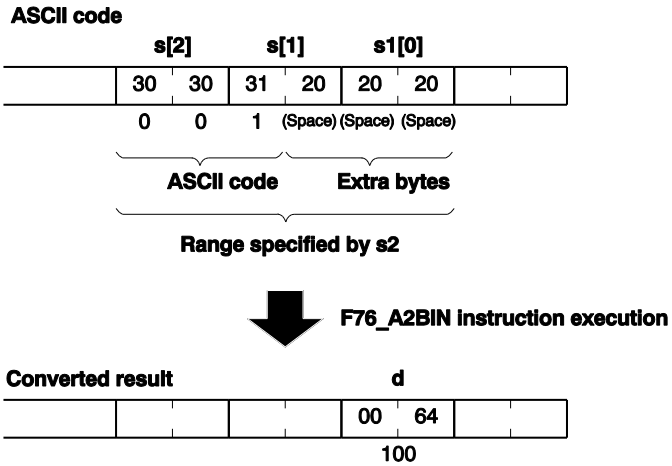


- Der zu konvertierende ASCII-Code sollte in Richtung der letzten Adresse des festgelegten Speicherbereichs abgelegt werden.
- Wenn der Speicherbereich **s1** und **s2** größer ist als durch die konvertierten Daten benötigt, schreiben Sie "0" (ASCII-HEX-Code: 16#30) oder "SPACE" (ASCII-HEX-Code: 16#20) in die zusätzlichen Bytes.
- ASCII-Codes mit Zeichen (wie + : 16#2B und -: 16#2D) werden ebenfalls konvertiert. Die + Codes können wegfallen.

Konvertierung eines ASCII-Codes, der eine negative Zahl anzeigt



Konvertierung eines ASCII-Codes, der eine positive Zahl anzeigt



ASCII HEX-Code zur Darstellung von Dezimalzahlen:

ASCII-HEX-Code	Dezimalzahlen
16#20	SPACE
16#2B	+
16#2D	-
16#30	0
16#31	1
16#32	2
16#33	3
16#34	4
16#35	5
16#36	6
16#37	7
16#38	8
16#39	9

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F76_A2BIN (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1	WORD	Anfangsadresse für ASCII-Code (Quelle)
s2	INT	Legt die Anzahl der zu konvertierenden Bytes des Quelldatenbereichs fest.
d	ANY16	Zieladresse für das Speichern der konvertierten Daten (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s1	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die durch s2 festgelegte Bytezahl den in s1 festgelegten Speicherplatz überschreiten. ▪ der in s2 festgelegte Wert als "0" erkannt wird. ▪ die Konvertierungsergebnisse den in d festgelegten Speicherplatz überschreiten. ▪ der ASCII-Code nicht mit Dezimalzahlen (0 bis 9) übereinstimmt oder mit dem ASCII-Zeichen (+, -, und Leerzeichen), das festgelegt wurde.
R9008	%MX0.900.8	kurzzeitig	

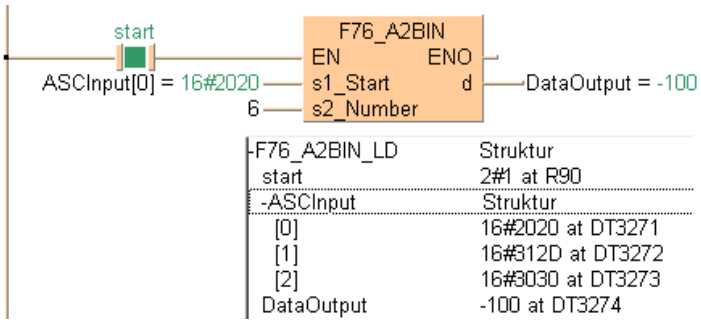
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	ASCInput	ARRAY [0..2] OF WORD	[16#2020,16#312D,16#3030]
2	VAR	DataOutput	INT	0

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Anzahl der zu konvertierenden Bytes wird direkt am Kontakt für s2 eingegeben. Dieses Programmierbeispiel basiert auf dem Beispiel für die Konvertierung negativer Zahlen von F76_A2BIN auf der Hauptseite.

KOP



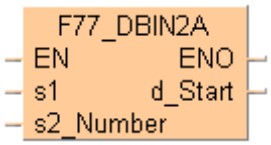
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F76_A2BIN( s1_Start:= ASCInput[0] ,
          s2_Number:= 6 ,
          d=> DataOutput );
END_IF;
```

F77_DBIN2A

32-bit BIN -> ASCII Umwandlung

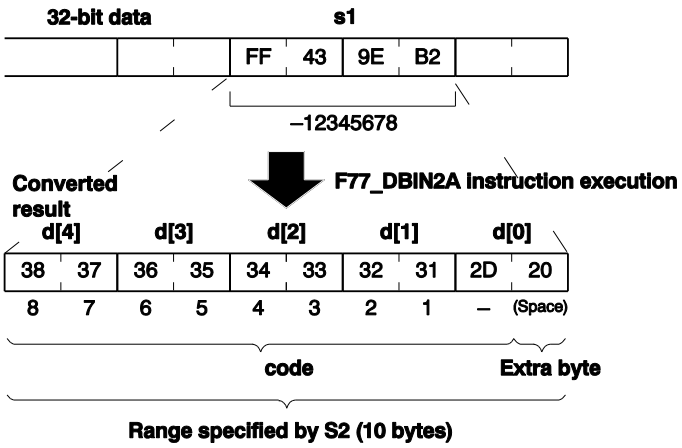
Erklärung Der binäre 32-Bit-Wert, der in **s1** festgelegt ist, wird in den hexadezimalen ASCII-Code konvertiert. Das Ergebnis der Konvertierung wird ab der Anfangsadresse **d** im Zieldatenbereich **s2** gespeichert. **s2** legt fest, wie viele Bytes der Zieldatenbereich enthalten soll (Dezimalzahlen).



- Wenn eine positive Zahl konvertiert wird, wird das Zeichen "+" nicht konvertiert.
- Wenn eine negative Zahl konvertiert wird, wird das Zeichen "-" ebenfalls in den ASCII-Code (ASCII-HEX-Code: 16#2D) konvertiert.
- Wenn der Speicherbereich **s2** größer ist als durch die konvertierten Daten benötigt, wird der ASCII-Code für "SPACE" (ASCII-HEX-Code: 16#20) in einem Extrabereich gespeichert.
- Die Werte werden in Richtung der Endadresse gespeichert, so dass sich die Position des ASCII-Code in Abhängigkeit von der Größe des Datenspeichers ändern kann.

Wenn die Anzahl der Bytes des ASCII-Codes (inklusive der Minuszeichen) nach der Konvertierung größer ist als die Anzahl der Bytes, die durch **s2** festgelegt wurden, tritt eine Funktionsfehler auf. Stellen Sie sicher, dass das Vorzeichen beachtet wird, wenn das Konvertierungsobjekt für **s2** festgelegt wird.

Beispiel für die Konvertierung negativer Zahlen aus dem 32–Bit Format in den ASCII-Code



Dezimalzeichen für die Darstellung des ASCII-HEX-Codes:

Dezimalzahlen	ASCII-HEX-Code
SPACE	16#20
+	16#2B
-	16#2D
0	16#30
1	16#31
2	16#32
3	16#33
4	16#34
5	16#35
6	16#36
7	16#37
8	16#38
9	16#39

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F77_DBIN2A (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	32-Bit Bereich, der konvertiert werden soll (Quelle)
s2	INT	Legt die Anzahl der Bytes fest, die die Zielwerte (ASCII-Code) ausdrücken.
d	WORD	16-Bit Bereich für die Speicherung des ASCII-Codes (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s1	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker:

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die durch s2 festgelegte Bytezahl den in d festgelegten Speicherplatz überschreitet. ▪ der in s2 festgelegte Wert als "0" erkannt wird.
R9008	%MX0.900.8	kurzzeitig	▪ die berechneten Ergebnisse den in d festgelegten Speicherplatz überschreiten. ▪ die Anzahl der Bytes des Konvertierungsergebnisses die in s2 festgelegte Bytezahl überschreiten.

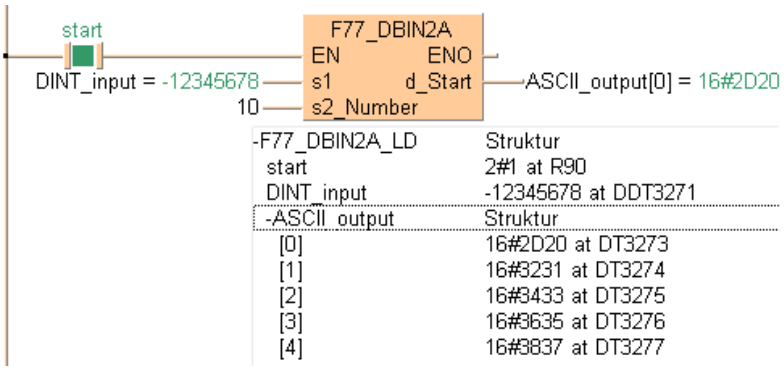
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	start	BOOL	FALSE
1	VAR	DINT_input	DINT	-12345678
2	VAR	ASCII_output	ARRAY [0..4] OF WORD	[5(0)]

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Anzahl der zu konvertierenden Bytes wird direkt am Kontakt für **s2** eingegeben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

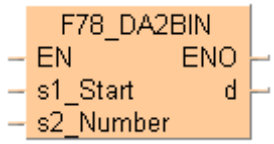
```
IF start THEN
    F77_DBIN2A( s1:= DINT_input ,
          s2_Number:= 10 ,
          d_Start=> ASCII_output[0] );
END_IF;
```

F78_DA2BIN

ASCII -> 32 bit BIN Umwandlung

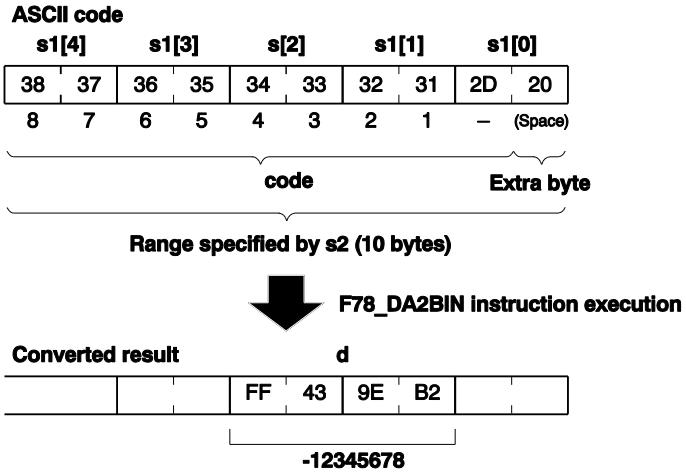
Erklärung

Die hexadezimalen ASCII-Daten ab der Anfangsadresse **s1** werden, wie in **s2** festgelegt, in 32-Bit-Werte konvertiert. Das Ergebnis der Konvertierung wird ab der Anfangsadresse **d** im Zieldatenbereich gespeichert. **s2** legt fest, wie viele Bytes der Zieldatenbereich enthalten soll (Dezimalzahlen).



- Der zu konvertierende ASCII-Code sollte in Richtung der letzten Adresse des festgelegten Speicherbereichs abgelegt werden.
- Wenn der Speicherbereich **s1** und **s2** größer ist als durch die konvertierten Daten benötigt, schreiben Sie "0" (ASCII-HEX-Code: 16#30) oder "SPACE" (ASCII-HEX-Code: 16#20) in die zusätzlichen Bytes.
- ASCII-Codes mit Zeichen (wie + : 16#2B und -: 16#2D) werden ebenfalls konvertiert. Die + Codes können wegfallen.

Konvertierung eines ASCII-Codes, der eine negative Zahl anzeigt



ASCII HEX-Code zur Darstellung von Dezimalzahlen:

ASCII-HEX-Code	Dezimalzahlen
16#20	SPACE
16#2B	+
16#2D	-
16#30	0
16#31	1
16#32	2
16#33	3
16#34	4
16#35	5
16#36	6
16#37	7
16#38	8
16#39	9

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im

Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F78_DA2BIN (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1	WORD	Anfangsadresse für ASCII_Code (Quelle)
s2	INT	Legt die Anzahl der zu konvertierenden Bytes des Quelldatenbereichs fest.
d	ANY32	Speicherbereich für 32-Bit-Daten (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s1	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die durch s2 festgelegte Bytezahl den in s1 festgelegten Speicherplatz überschreitet. ▪ der in s2 festgelegte Wert als "0" erkannt wird. ▪ die berechneten Ergebnisse den in d festgelegten Speicherplatz überschreiten. ▪ die Konvertierungsergebnisse den 32-Bit-Speicherplatz überschreiten. ▪ der ASCII-Code nicht mit Dezimalzahlen (0 bis 9) übereinstimmt oder mit dem ASCII-Zeichen (+, -, und Leerzeichen), das festgelegt wurde.
R9008	%MX0.900.8	kurzzeitig	

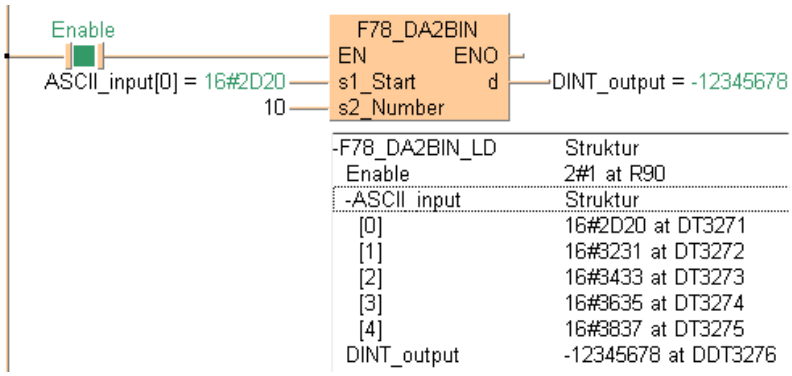
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Enable	BOOL	FALSE
1	VAR	ASCII_input	ARRAY [0..4] OF WORD	[16#2D20,16#3231,16#3433,16#3635,16#3837]
2	VAR	DINT_output	DINT	0

Rumpf Wenn die Variable **Enable** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Anzahl der zu konvertierenden Bytes wird direkt am Kontakt für **s2** eingegeben. Dieses Programmierbeispiel basiert auf dem oben riebenen Beispiel für die Konvertierung negativer Zahlen.

KOP



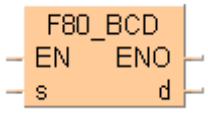
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF Enable THEN
    F78_DA2BIN( s1_Start:= ASCII_input[0] ,
          s2_Number:= 10 ,
          d=> DINT_output );
END_IF;
```

F80_BCD

16-Bit BIN -> 4-digit BCD Umwandlung

Erklärung Die binärcodierten 16-Bit-Werte des Speicherregisters **s** werden BCD-codiert (4-wertige Dezimale), wenn der Trigger **EN** den Status ON hat. Die konvertierte Daten werden in Speicherregister **d** abgelegt. Die binärcodierten Daten, die in den BCD-Code konvertiert werden können, liegen im Bereich von 0 (0 Hex) bis 9999 (270F Hex).



Source [s]: 16

Bit position	15 . . 12	11 . . 8	7 . . 4	3 . . 0
Binary data	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 0
Decimal	16			

Conversion (to BCD code)

Destination [d]: 16#16 (BCD)

Bit position	15 . . 12	11 . . 8	7 . . 4	3 . . 0
Binary data	0 0 0 0	0 0 0 0	0 0 0 1	0 1 1 0
BCD Hex code	0	0	1	6

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Siehe auch: BCD

SPS-Typen **Verfügbarkeit von F80_BCD (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	Binärcodierte Daten (Quelle), Bereich: 0 bis 9999
d	WORD	Speicheradresse des 4-stelligen BCD-Codes (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ 16-Bit binärcodierte Werte außerhalb des Bereichs 0 (16#0) bis 9999 (16#270F) konvertiert werden.
R9008	%MX0.900.8	kurzzeitig	

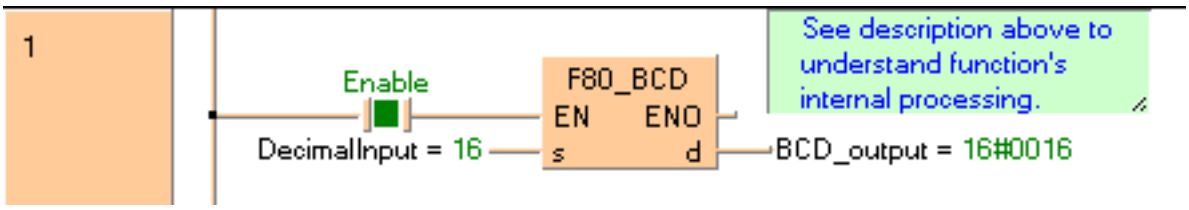
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Enable	BOOL	FALSE	activates the function
1	VAR	DecimalInput	WORD	16	
2	VAR	BCD_output	WORD	0	

Rumpf Wenn die Variable **Enable** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Der Dezimalwert in **DecimalInput** wird in hexadezimale BCD-Werte konvertiert und in der Variablen **BCD_output** gespeichert.

KOP



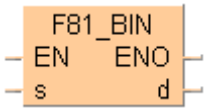
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF Enable THEN
    F80_BCD(DecimalInput, BCD_output);
END_IF;
```

F81_BIN

4-Digit BCD -> 16-Bit BIN Umwandlung

Erklärung Die BCD-codierten 16-Bit-Werte des Speicherregisters **s** werden binärcodiert, wenn der Trigger **EN** den Status ON hat. Die Konvertierungsergebnisse werden im Speicherregister **d** abgelegt.



Source [s]: 16#15 (BCD)

Bit position	15 . . 12	11 . . 8	7 . . 4	3 . . 0
BCD code	0 0 0 0	0 0 0 0	0 0 0 1	0 1 0 1
BCD Hex code	0	0	1	5

Conversion (to binary data)

Destination [d]: 15

Bit position	15 . . 12	11 . . 8	7 . . 4	3 . . 0
Binary data	0 0 0 0	0 0 0 0	0 0 0 0	1 1 1 1
Decimal	15			

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Siehe auch: BCD

SPS-Typen **Verfügbarkeit von F81_BIN (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	WORD	Speicherregister für die 4-stelligen BCD-Werte (Quelle)
d	ANY16	Zieladresse für das Speichern der konvertierten Daten (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die in s festgelegten Werte keine BCD-Werte sind.
R9008	%MX0.900.8	kurzzeitig	

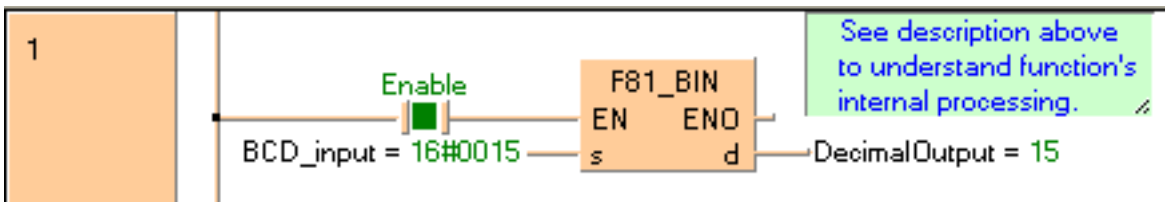
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Enable	BOOL	FALSE	activates the function
1	VAR	BCD_input	WORD	16#0015	
2	VAR	DecimalOutput	INT	0	

Rumpf Wenn die Variable **Enable** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Der BCD-Wert, der der Variablen **BCD_input** zugeordnet ist, wird in einen Dezimalwert konvertiert und in der Variablen **DecimalOutput** abgelegt. Das Monitorwertsymbol ist sowohl für den KOP- als auch den AWL-Rumpf aktiviert.

KOP



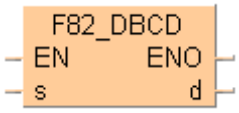
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF Enable THEN
    F81_BIN(BCD_Input, DecimalOutput);
END_IF;
```

F82_DBCD

32-bit BIN -> 8-digit BCD Umwandlung

Erklärung Die binärcodierten 32-Bit-Werte des Speicherregisters **s** werden BCD-codiert (8-wertige Dezimale), wenn der Trigger **EN** den Status ON hat. Die konvertierte Daten werden in Speicherregister **d** abgelegt. Die binärcodierten Daten, die in den BCD-Code konvertiert werden können, liegen im Bereich von 0 (0 Hex) bis 99.999.999 (5F5E0FF Hex).



Source (s): 72811730

Bit position	15 · · · 12	11 · · · 8	7 · · · 4	3 · · · 0	15 · · · 12	11 · · · 8	7 · · · 4	3 · · · 0
Binary data	0 0 0 0	0 1 0 0	0 1 0 1	0 1 1 1	0 0 0 0	0 1 0 0	1 1 0 1	0 0 1 0
Decimal	72811730							

32-bit area

Destination (d): 16#72811730

Bit position	15 · · · 12	11 · · · 8	7 · · · 4	3 · · · 0	15 · · · 12	11 · · · 8	7 · · · 4	3 · · · 0
BCD code	0 1 1 1	0 0 1 0	1 0 0 0	0 0 0 1	0 0 0 1	0 1 1 1	0 0 1 1	0 0 0 0
BCD Hex code	7	2	8	1	1	7	3	0

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Siehe auch: BCD

SPS-Typen **Verfügbarkeit von F82_DBCD (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	ANY32	Binärcodierte Daten (Quelle), Bereich: 0 bis 99.999.999
d	DWORD	Speicheradresse des 8-stelligen BCD-Codes (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die in s festgelegten 32-Bit-Werte außerhalb des Bereichs 0 (16#0) bis 99999999 (16#5F5E0FF) konvertiert werden.
R9008	%MX0.900.8	kurzzeitig	

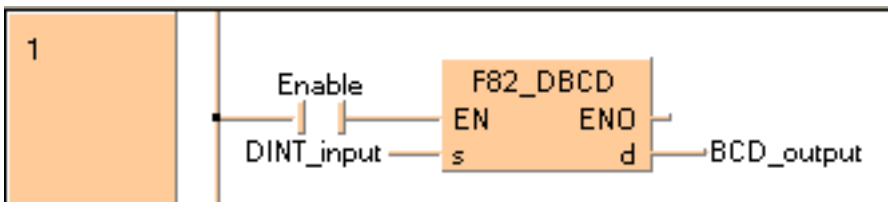
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	DWORD	16#00000...	
2	VAR	output_value	DWORD	0	result after 0->1 leading edge from start: 16#00000018

Rumpf Wenn die Variable **Enable** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Der Dezimalwert in **DINT_input** wird in einen hexadezimalen BCD-Wert konvertiert und in der Variablen **BCD_output** gespeichert. Sie können auch einen Dezimalwert, einen binärcodierten Wert (Präfix 2#) oder einen Hexadezimalwert (Präfix 16#) direkt am Kontakt s eingeben.

KOP



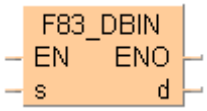
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF Enable THEN
    F82_DBCD(DINT_input, BCD_output);
END_IF;
```

F83_DBIN

8-digit BCD -> 32-Bit BIN Umwandlung

Erklärung Die BCD-codierten Werte des Speicherregisters **s** werden binärcodiert, wenn der Trigger **EN** den Status ON hat. Die Konvertierungsergebnisse werden im Speicherregister **d** abgelegt.



Source (s): 16#72811730 (BCD)

Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0	15 · · 12	11 · · 8	7 · · 4	3 · · 0
BCD code	0 1 1 1	0 0 1 0	1 0 0 0	0 0 0 1	0 0 0 1	0 1 1 1	0 0 1 1	0 0 0 0
BCD Hex code	7	2	8	1	1	7	3	0

32-bit area

Destination (d): 72811730

Bit position	15 · · 12	11 · · 8	7 · · 4	3 · · 0	15 · · 12	11 · · 8	7 · · 4	3 · · 0
Binary data	0 0 0 0	0 1 0 0	0 1 0 1	0 1 1 1	0 0 0 0	0 1 0 0	1 1 0 1	0 0 1 0
Decimal	72811730							

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Siehe auch: BCD

SPS-Typen **Verfügbarkeit von F83_DBIN (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	DWORD	Speicherregister der 8-stelligen BCD-Werte (Quelle)
d	ANY32	Zieladresse für das Speichern der konvertierten 32-Bit Daten (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die in **s** festgelegten Werte keine BCD-Werte sind.
R9008	%MX0.900.8	kurzzeitig	

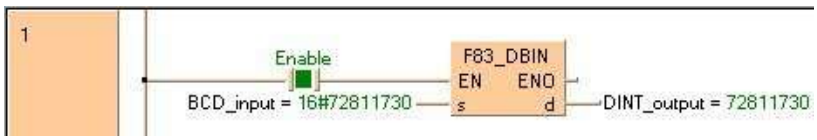
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	DWORD	16#00000...	
2	VAR	output_value	DWORD	0	result after 0->1 leading edge from start: 16#0000000C

Rumpf Wenn die Variable **Enable** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Der BCD-Wert, der der Variablen **BCD_input** zugeordnet ist, wird in einen Dezimalwert konvertiert und in der Variablen **DINT_output** abgelegt.

KOP

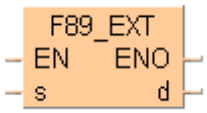


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

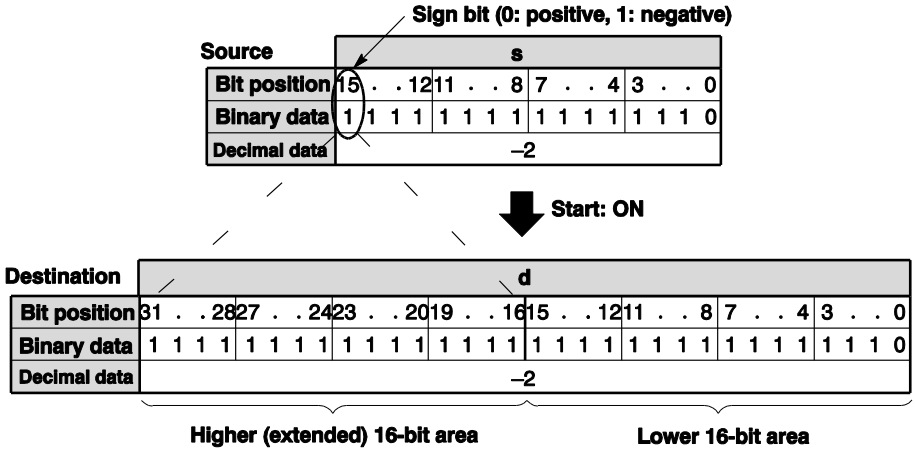
```
IF Enable THEN
    F83_DBIN(BCD_input, DINT_Output);
END_IF;
```

F89_EXT Vorzeichenbit verschieben

Erklärung 16-Bit Daten werden in 32-Bit Daten konvertiert, ohne das die Vorzeichen und Werte verändert werden. F89 konvertiert eine 16-Bit-Zahl in eine 32-Bit-Zahl unter Berücksichtigung des Vorzeichens.



Wenn das durch **s** spezifizierte Vorzeichen-Bit (Bitposition 15) der 16-Bit-Daten 0 ist, sind alle höherwertigen 16-Bit die der Variablen **d** zugeordnet sind, 0. Wenn das Vorzeichen-Bit von **s** gleich 1 ist, sind die höherwertigen 16-Bit von **d** ebenfalls 1.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F89_EXT (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	16-Bit-Quelldatenregister, Bit 15 ist Vorzeichen-Bit
d	ANY32	32-Bit-Zielregister, s kopiert in niederwertigen 16-Bit-Bereich, höherwertige 16-Bits mit Vorzeichen-Bit s

Operanden

Für	Merker				T/C		Register			Konstante
s	-	WY	WR	WL	SV	EV	DT	LD	FL	-
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

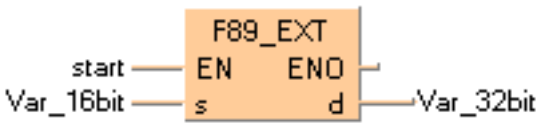
Beispiel In diesem Beispiel wird die Funktion F89_EXT im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	Var_16bit	INT	0	16bit value
2	VAR	Var_32bit	DINT	0	32bit value

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

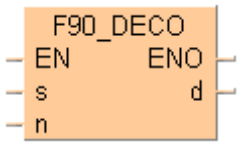


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F89_EXT(Var_16bit, Var_32bit);
END_IF;
```

F90_DECO Decodierung

Erklärung Der 16-Bit-Wert des Speicherregisters **s** wird, gemäß den Inhalten im Speicherregister **n** decodiert, wenn der Trigger EN den Status ON hat. Das Ergebnis der Konvertierung wird ab der Anfangsadresse **d** im Zieldatenbereich gespeichert.



n spezifiziert die Position des Startbits und die Bitanzahl, die mit Hilfe von hexdezimalen Daten decodiert wird:

- Bit Nr. 0 bis 3: Anzahl der zu decodierenden Bits (1 bis 8)
- Bit Nr. 8 bis 11: Bitposition des ersten Bits (0 bis 15)

(Die Bits Nr. 4 bis 7 und Nr. 12 bis 15 sind nicht relevant.)

z.B. wenn **n** = 16#0404, werden 4 Bits ab Bitposition 4 decodiert.

Beziehung zwischen der Anzahl der Bits und der zur Decodierung notwendigen Datenbereiche:

Zahl der zu decodierenden Bits	Datenbereich nötig für das Ergebnis	Gültige Bits im Ergebnisbereich
1	1-Wort	2-Bit*
2	1-Wort	4-Bit*
3	1-Wort	8-Bit*
4	1-Wort	16-Bit
5	2-Wort	32-Bit
6	4-Wort	64-Bit
7	8-Wort	128-Bit
8	16-Wort	256-Bit

*Ungültige Bits im Datenbereich, der für das Ergebnis nötig ist, werden auf 0 gesetzt.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F90_DECO (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	16-Bit-Quellregister oder äquivalente Konstante müssen decodiert werden
n		Kontrolldaten um die Startposition zu spezifizieren und Anzahl der Bits die decodiert werden sollen
d		16-Bit-Anfangsregister um die decodierten Daten (Ziel) zu speichern

Die Variablen **s, n** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s, n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

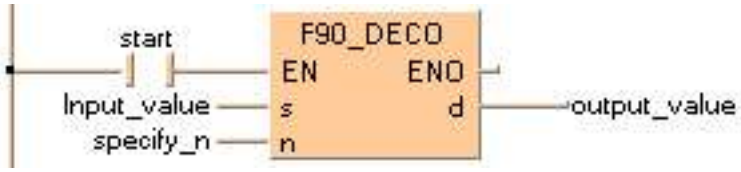
Beispiel In diesem Beispiel wird die Funktion F90_DECO im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	WORD	2#1100011000011110	
2	VAR	specify_n	WORD	16#0003	specifies decoding
3	VAR	output_value	WORD	0	result after a 0->1 leading edge from start: 2#0000000001000000

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



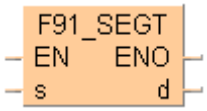
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F90_DECO( s:= input_value ,
              n:= specify_n ,
              d=> output_value );
END_IF;
```

F91_SEGT

16-Segment-Codierung

Erklärung Der binäre 16-Bit-Wert des Speicherregisters **s** oder einer Konstante s wird zum 7-Segment-Code codiert. Das Ergebnis der Codierung steht im Speicherregister **d** subtrahiert. Dazu muss der Trigger auf **EN** EIN gesetzt sein. Das Ergebnis der Konvertierung wird ab der Anfangsadresse **d** im Zieldatenbereich gespeichert. Daten im 7-Segment-Code besetzen 8 Bit (1 Byte), um eine Stelle auszudrücken.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F91_SEGT (s. S. 1192)**

Datentypen

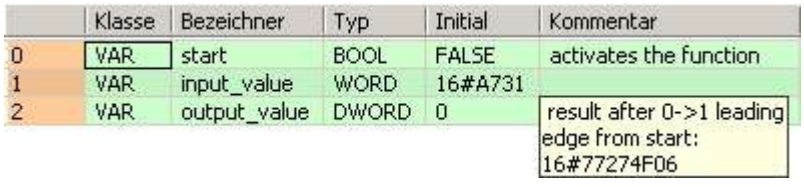
Variable	Datentyp	Funktion
s	ANY16	16-Bitregister oder äquivalente Konstante die zu 7-Segment-Code (Quelle) konvertiert zu werden
d	ANY32	32-Bit-Speicherregister für 4-Digit-Daten für 7-Segment-Code (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

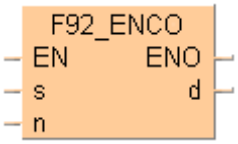
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F91_SEGT(input_value, output_value);
END_IF;
```

F92_ENCO Codierung

Erklärung

Die Daten des Speicherbereichs werden ab der Anfangsadresse **s**, gemäß den Inhalten im Speicherregister **n** codiert. Dabei wird die Bitstelle des gesetzten Bits der Quelldaten dezimal in das Ergebnis geschrieben. Der Trigger **EN** hat dabei den Status ON. Das Ergebnis der Codierung steht in Speicherregister **d**. Ungültige Bits im Datenbereich, der für das Ergebnis der Codierung festgelegt wurde, werden auf 0 gesetzt.



n spezifiziert die Position des Startbits im Speicherregister **d** und die Bitanzahl, die mit Hilfe von hexdezimalen Daten decodiert wird:

Bit Nr. 0 bis 3	Anzahl der zu codierenden Bits
Bit Nr. 8 bis 11	Bitposition des ersten Bits des Ergebnisses der Codierung in Speicherregister d

(Die Bits Nr. 4 bis 7 und Nr. 12 bis 15 sind nicht relevant.)

☞

- **Setzen Sie in dem zu prüfenden Bereich mindestens ein Bit, um eine Fehlermeldung der SPS zu vermeiden.**
- **Wenn mehrere Bits gesetzt sind, dann wird immer das höchstwertige Bit ausgewertet.**

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F92_ENCO (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	16-Bit-Anfangsregister, dass codiert werden muss (Quelle)
n		Kontrolldaten um die Startposition zu spezifizieren und Anzahl der Bits die codiert werden sollen
d		16-Bit-Speicherregister zum Speichern der codierten Daten (Ziel)

Die Variablen **s, n** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

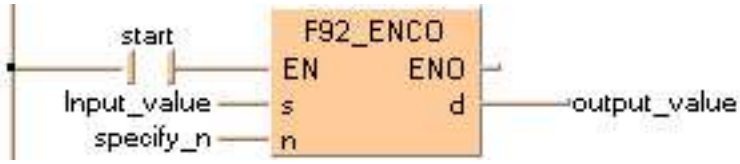
Beispiel In diesem Beispiel wird die Funktion F92_ENCO im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	WORD	2#0000000001000000	
2	VAR	specify_n	WORD	16#0003	specifies the encodation
3	VAR	output_value	WORD	0	result after a 0->1 leading edge from start: 2#0000000000000110

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



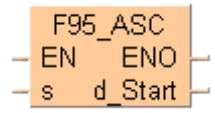
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F92_ENCO( s:= input_value ,
              n:= specify_n ,
              d=> output_value );
END_IF;
```

F95_ASC

12 Zeichen -> ASCII Transfer

Erklärung Die Zeichenkonstante **s** (maximal 12 Zeichen) wird in ASCII-Code umgewandelt. Der konvertierte ASCII-Code wird in 6 16-Bit-Speicherbereichen ab der Anfangsadresse **d_Start** gespeichert.



[s]	Character constants	A B C 1 2 3 0 _ D E F

[d]	Data register	d[5]		d[4]		d[3]		d[2]		d[1]		d[0]	
	ASCII HEX code	2 0	4 6	4 5	4 4	2 0	3 0	3 3	3 2	3 1	4 3	4 2	4 1
	ASCII character		F	E	D		0	3	2	1	C	B	A

SPACE

☞ **Wenn die Anzahl der in s festgelegten Zeichenkonstanten kleiner als 12 ist, wird der ASCII-Code 16#20 (SPACE) in einem extra Zielbereich gespeichert, z.B. s = '12345', d[0] = 3231, d[1] = 3433, d[2] = 2034, d[3] - d[5] = 2020.**

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F95_ASC (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s	Konstante, keine Variable möglich	Zeichenkonstanten, max. 12 Zeichen (Quelle)
d	ANY16	16-Bit-Anfangsadresse für das Speichern des ASCII-Codes (6 Worte, Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s	-	-	-	-	-	-	-	-	-	dez., hex.
d_Start	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Speicherbereich für den ASCII-Code das Limit überschreitet (6 Worte: 16-Bit-Speicherbereiche)
R9008	%MX0.900.8	kurzzeitig	

ASCII Hex-Code

b7															
b6								0	0	0	0	1	1	1	1
b5								0	0	1	1	0	0	1	1
b4								0	1	0	1	0	1	0	1
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	AXCII-HEX-Code	Höerwertige Bits						
									0	1	2	3	4	5	6	7
				0	0	0	0	0	NUL	DEL	SPACE	0	@	P		p
				0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
				0	0	1	0	2	STX	DC2	"	2	B	R	b	r
				0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
				0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
				0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
				0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
				0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
				1	0	0	0	8	BS	CAN	(	8	H	X	h	x
				1	0	0	1	9	HT	EM	)	9	I	Y	i	y
				1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
				1	0	1	1	B	VT	ESC	+	;	K	[	k	{
				1	1	0	0	C	FF	FS	,	<	L	\	l	⊠
				1	1	0	1	D	CR	GS	–	=	M	]	m	}
				1	1	1	0	E	SO	RS	.	>	N	^	n	~
				1	1	1	1	F	SI	US	/	?	O	_	o	DEL

Niederwertige Bits

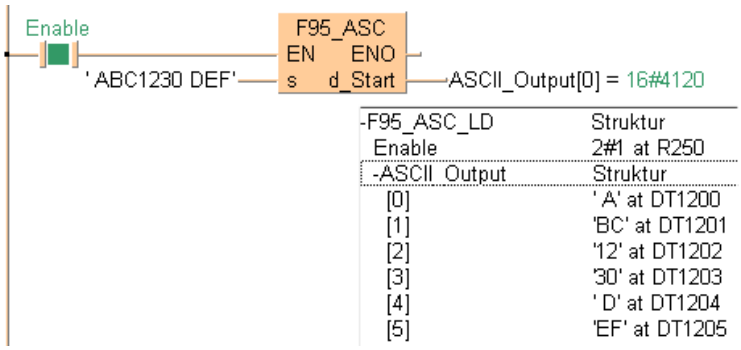
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	output_value	ARRAY [0..5] OF WORD	[6(0)]	result after a 0->1 leading edge from start: [16#3332,16#4241,16#2043, 16#2020,16#2020,16#2020]

Rumpf Wenn die Variable **Enable** freigegeben wird, werden die am Eingang s eingegeben Zeichenkonstanten in den ASCII-Code konvertiert und in der Variablen **ASCII_Output** gespeichert.

KOP



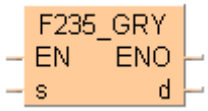
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF Enable THEN
    F95_ASC( s:= 'ABC1230 DEF' ,
        d_Start=> ASCII_Output[0] );
END_IF;
```

F235_GRY

16-Bit-Wert -> 16-Bit-Gray-Code Umwandlung

Erklärung Die Funktion wandelt einen Wert am Eingang **s** in einen Gray-Code-Wert um. Das Umwandlungsergebnis wird am Ausgang **d** zurückgegeben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F235_GRY (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	Zu konvertierende Quelldaten
d		Ziel zum Speichern des Gray-Codes

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

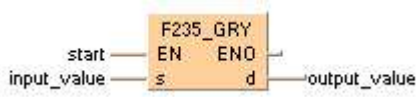
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	INT	23	
2	VAR	output_value	INT	0	result: here 28

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



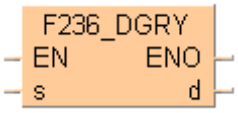
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F235_GRY(input_value, output_value);
END_IF;
```

F236_DGRY

32-Bit-Wert -> 32-Bit-Gray-Code Umwandlung

Erklärung Die Funktion wandelt einen Wert am Eingang **s** in einen Gray-Code-Wert um. Das Umwandlungsergebnis wird am Ausgang **d** zurückgegeben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F236_DGRY (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s	ANY32	Zu konvertierende Quelldaten
d		Ziel zum Speichern des Gray-Codes

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	DINT	12345678	
2	VAR	output_value	DINT	0	result: here 14832105

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

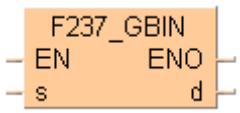
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F236_DGRY(input_value, output_value);
END_IF;
```

F237_GBIN

16-Bit-Gray-Code -> 16-Bit-BIN-Umwandlung

Erklärung Die Funktion wandelt einen Gray-Code-Wert am Eingang **s** in einen binären Wert um. Das Umwandlungsergebnis wird am Ausgang **d** zurückgegeben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F237_GBIN (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	Quellbereich des Gray-Codes
d		Ziel zum Speichern der konvertierten Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	output	BOOL	FALSE	activates the function
1	VAR	input_value	INT	28	
2	VAR	output_value	INT	0	result: here 23

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

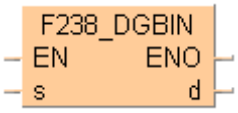
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F237_GBIN(input_value, output_value);
END_IF;
```

F238_DGBIN

32-Bit-Gray-Code -> 32-Bit-Wert Umwandlung

Erklärung Die Funktion wandelt einen Gray-Code-Wert am Eingang **s** in einen binären Wert um. Das Umwandlungsergebnis wird am Ausgang **d** zurückgegeben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F238_DGBIN (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s	ANY32	Quellbereich des Gray-Codes
d		Zielbereich zum Speichern konvertierter Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	DINT	14832105	
2	VAR	output_value	DINT	0	result: here 12345678

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

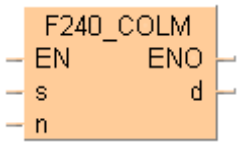
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F238_DGBIN(input_value, output_value);
END_IF;
```

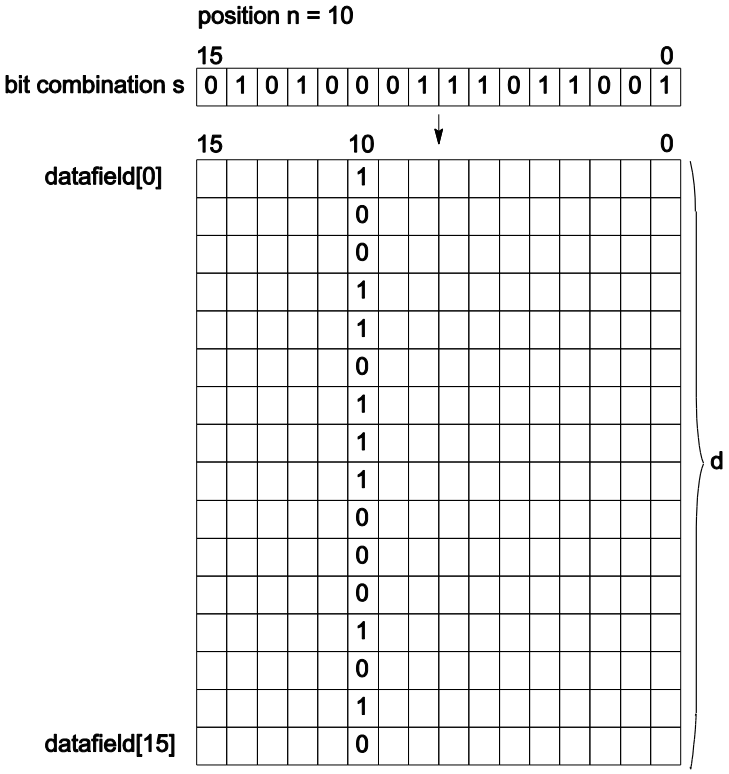
F240_COLM

Bit-Zeile -> Bit-Spalte Umwandlung

Erklärung Die Funktion erzeugt aus einem Wert am Eingang **s** eine Bitspalte, die am Ausgang **d** innerhalb eines ARRAY zurückgegeben wird. Die Position der Spalte im ARRAY wird dem Eingang **n** übergeben. Bei **n** können Sie einen Wert zwischen 0 und 15 eingeben.



Die Bits des ARRAY, die nicht vom Eingangswert (Eingang **s**) überschrieben werden, bleiben unverändert.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F240_COLM (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	Quelle
n		Spezifiziert die Bit-Position
d	ARRAY [0..15] OF ANY16 **oder** WORD	Zielbereich einer Bit-Spalte, der wiederbeschrieben wird

Operanden

Für	Merker				T/C		Register			Konstante
s, n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die am Eingang n angegebene Bit-Position nicht zwischen 0 und 15 liegt. ▪ das Umwandlungsergebnis zu einem Überlauf des am Ausgang d anliegenden Adressbereichs führt. ▪
R9008	%MX0.900.8	kurzzeitig	

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	bit_combination	WORD	16#FFFF	
2	VAR	position	INT	15	acceptable values 0..15
3	VAR	data_field	ARRAY [0..15] OF WORD	[16(0)]	result: bit 16 (highest-value bit) of the array's elements is set to 1 (TRUE)
4	VAR				

Hier wurden die Eingangsvariablen **bit_combination** und **position** deklariert. Statt dessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

ST

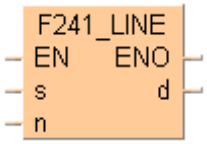
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

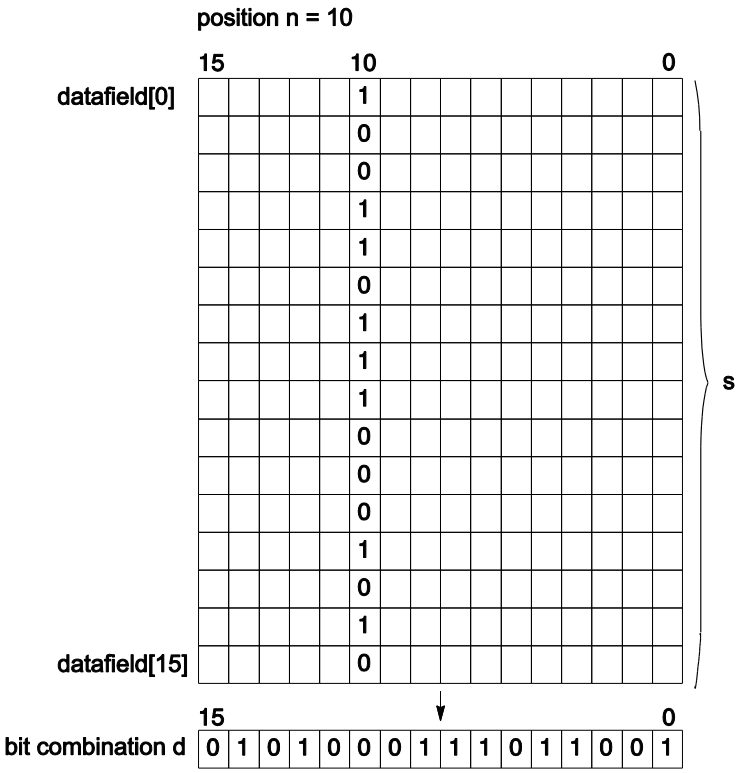
```
IF start THEN
    F240_COLM( s:= bit_combination,
          n:= position,
          d=> data_field );
END_IF;
```

F241_LINE

Bit-Spalte -> Bit-Zeile Umwandlung

Erklärung Die Funktion liest aus einem ARRAY am Eingang **s** die Bits einer bestimmten Spalte heraus und gibt sie am Ausgang **d** zurück. Die Position an der gelesen werden soll, wird dem Eingang **n** übergeben. Der Positionswert am Eingang **n** muss zwischen 0 und 15 liegen.





Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F241_LINE (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s	ARRAY [0..15] OF ANY16	Quellbereich, in dem die Bit-Spalte gelesen wird
n	ANY16	Spezifiziert die Bit-Position
d		Zielbereich zum Speichern konvertierter Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die am Eingang n angegebene Bit-Position nicht zwischen 0 und 15 liegt. ▪ ein Überlauf des am Eingang s anliegenden Adressbereichs vorliegt. ▪
R9008	%MX0.900.8	kurzzeitig	

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

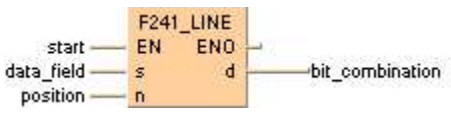
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	bit_combination	WORD	0	result: here 16#FFFF
2	VAR	position	INT	15	acceptable values 0.. 15
3	VAR	data_field	ARRAY [0..15] OF WORD	[16(16#8000)]	highest value bit of the array's elements is set to 1 (TRUE)
4	VAR				

Hier wurden die Eingangsvariablen **bit_combination** und **position** deklariert. Statt dessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST

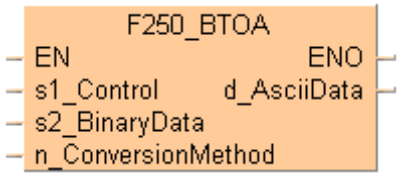
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F241_LINE( s:= data_field ,
          n:= position ,
          d=> bit_combination );
END_IF;
```

F250_BTOA

Binär -> ASCII Wandlung

Erklärung Wandelt die 16-Bit/32-Bit binärkodierten Werte aus dem Bereich in ASCII-Code um, der durch **s2_BinaryData** angegeben wird. Die Umwandlungsmethode wird durch **n_ConversionMethod** entsprechend den vier Steuerzeichen von **s1_Control** festgelegt. Das Umwandlungsergebnis wird in das Speicherregister **d_AsciiData** geschrieben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F250_BTOA (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1_Control	STRING	Steuerzeichen **16 – D** **D: converts to decimal ASCII data** **H: converts to hexadecimal ASCII data** **+ Normal direction** **- Reverse direction** **16: converts in 16-bit (1-word) units** **32: converts in 32-bit (2-word) units**
s2_BinaryData	ANY	Anfangsbereich zum Speichern der Binärwerte
n_Conversion Method	ANY16	Umwandlungsmethode **16# □ □ □ □** **Number of ASCII characters per conve** **Offset in ASCII character units (8-bit)** **Number of 16-bit (1-word) or 32-bit (2-w units conve** (nähere Erläuterungen finden Sie in den nachstehenden Tabellen)
d_AsciiData	ANY	Anfangsbereich zum Speichern der ASCII-Werte

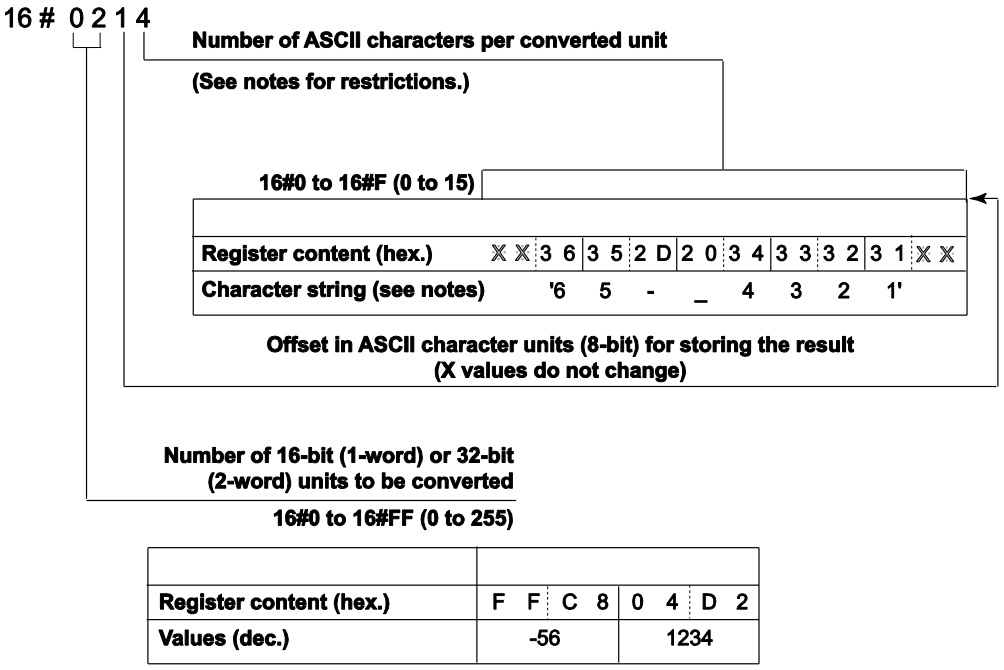
Operanden

Für	Merker				T/C		Register			Konstante
s1_Control	WX	WY	WR	WL	SV	EV	DT	LD	-	-
s2_BinaryData	WX	WY	WR	WL	SV	EV	DT	LD	-	-
n_Conversion Method	-	WY	WR	WL	SV	EV	DT	LD	-	dez., hex.
d_AsciiData	-	WY	WR	WL	SV	EV	DT	LD	-	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ das Steuerzeichen **s1_Control** fehlerhaft ist. ▪ die Standardrichtung (+) in **s1_Control** angegeben, und das Format dezimal ist. ▪ die Anzahl der ASCII-Zeichen pro umgewandelter Einheit, angegeben in **n_ConversionMethod**, für 16-Bitwerte den Wert 4 und für 32-Bitwerte den Wert 8 übersteigt und das hexadezimale Format durch **s1_Control** angegeben ist. ▪ 0 als Anzahl der 16- oder 32-Bit-Einheiten (1 oder 2 Worte) angegeben ist, die mit **n_ConversionMethod** umgewandelt werden sollen. ▪ die Anzahl der mit **n_ConversionMethod** umzuwandelnden 16- oder 32-Bit Dezimalzahlen den Speicherbereich der ASCII-Werte übersteigt. ▪ das Ergebnis der Umwandlung den Bereich überschreitet.
R9008	%MX0.900.8	kurzzeitig	

■ **Erläuterung der Umwandlungsmethode, z.B. n_ConversionMethod = 16#0214**



Register content (hex.)	FF C8	04 D2
Values (dec.)	-56	1234

☞ **Anzahl der ASCII-Zeichen (8-Bit) pro umgewandelter Einheit**

- Umwandlung von 16-Bit Binäreinheiten in hexadezimale ASCII-Werte:
- Bereich: 16#1 bis 16#4.
- Wenn ein Bereich unter 16#4 definiert ist, wird die angegebene Zeichenanzahl der niederwertigen Bytes gespeichert. Wenn die ursprünglichen Binärwerteinheiten nicht durch eine Einstellung unterhalb von 16#4 untergebracht werden können, tritt ein Fehler auf.
- Umwandlung von 32-Bit Binäreinheiten in hexadezimale ASCII-Werte:
- Bereich: 16#1 bis 16#8.
- Wenn ein Bereich unter 16#8 definiert ist, wird die angegebene Zeichenanzahl der niederwertigen Bytes gespeichert. Wenn die ursprünglichen Binärwerte nicht durch eine Einstellung unterhalb von 16#8 untergebracht werden können, tritt ein Fehler auf.
- Umwandlung von Binärwerteinheiten in dezimale ASCII-Werte:
- Bereich: 16#1 bis 16#F.
- Quelldaten werden als vorzeichenbehaftete Binärwerte behandelt. Bei einer negativen Zahl wird ein Minuszeichen "-" vorangestellt. Ein Leerzeichen "␣" wird als führendes Leerzeichen gespeichert, wenn der in d_AsciiData angegebene Bereich größer ist als die Anzahl der ASCII-Zeichen pro umgewandelter Einheit.

Umwandlungsbeispiele:

Binärwerte			**s1_Control**	**n_Conversion Method**	**Ergebnis der ASCII-Werte**				**Kommentar**
Datentyp	**Offs. in 16-Bit-Worteinheiten**	**Hex. Wert**			**D+3**	**D+2**	**D+1**	**D**	
	0	16#5678	16+H	16#204	21	43	65	87	Standardrichtung. 2 x 4 ASCII-Zeichen.
	1	16#1234							
INT, WORD	0	16#5678	16-H	16#204	43	21	87	65	Umgekehrte Richtung. 2 x 4 ASCII-Zeichen.
	1	16#1234							
INT, WORD	0	16#0456	16+H	16#203	XX	13	24	65	Standardrichtung. 2 x 3 ASCII-Zeichen.
	1	16#0123							
INT, WORD	0	16#0456	16-H	16#203	XX	32	16	54	Umgekehrte Richtung. 2 x 3 ASCII-Zeichen.
	1	16#0123							
DINT, DWORD	0	16#12345678	32+H	16#108	21	43	65	87	Standardrichtung. 1 x 8 ASCII-Zeichen.
DINT, DWORD	0	16#12345678	32-H	16#108	87	65	43	21	Umgekehrte Richtung. 1 x 8 ASCII-Zeichen.
DINT, DWORD	0	16#00012345	32+H	16#105	XX	X1	32	54	Standardrichtung. 1 x 5 ASCII-Zeichen.
DINT, DWORD	0	16#00012345	32-H	16#105	XX	X5	43	21	Umgekehrte Richtung. 1 x 5 ASCII-Zeichen.

'X' Werte ändern sich nicht.

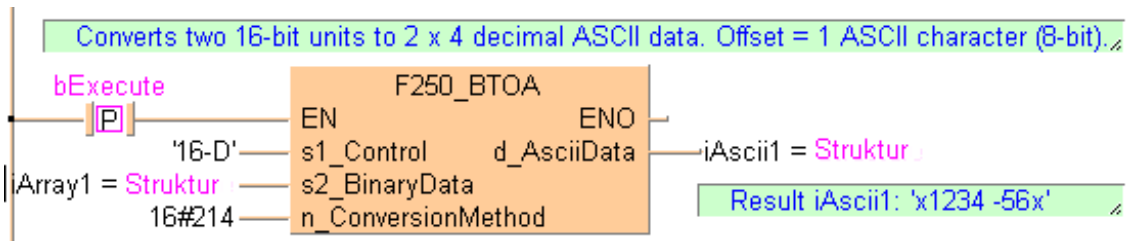
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bExecute	BOOL	FALSE
1	VAR	iArray1	ARRAY [0..1] OF INT	[1234,-56]
2	VAR	iAscii1	ARRAY [0..4] OF WORD	[5(16#FFFF)]

Rumpf Wenn **bExecute** auf TRUE gesetzt ist, wird der Befehl ausgeführt. Wandelt zwei 16-Bit-Einheiten in 2 x 4 dezimale ASCII-Werte um. Offset = 1 ASCII-Zeichen (8-Bit).

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

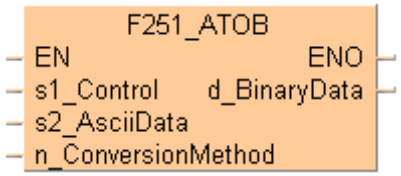
```
IF DF(bExecute) THEN
    F250_BTOA(s1_Control := '16-D',
        s2_BinaryData := iArray1,
        n_ConversionMethod := 16#214,
        d_AsciiData => iAscii1);

END_IF;
```

F251_ATOB

ASCII -> Binärumwandlung

Erklärung

Wandelt den ASCII-Code, der im Bereich **s2_AsciiData** festgelegt ist, in 16-Bit/32-Bit-Binärwerte um. Die Umwandlungsmethode wird durch **n_ConversionMethod** entsprechend den vier Steuerzeichen von **s1_Control** festgelegt. Das Umwandlungsergebnis wird in das Speicherregister **d_BinaryData** geschrieben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F251_ATOB (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1_Control	STRING	Steuerzeichen **D – 16** **16: converts the ASCII data to 16-bit to +32,767 (16#0 to 16#FFFF)** **32: converts the ASCII data to 32-bit -2,147,483,648 to +2,147,483,647 16#FFFFFFFF)** **+ Normal direction** **- Reverse direction** **D: converts decimal ASCII data** **H: converts hexadecimal ASCII data**
s2_AsciiData	ANY	Anfangsbereich zum Speichern der ASCII-Werte
n_ConversionMethod	ANY16	Umwandlungsmethode **16# □ □ □ □** **Number of ASCII characters per** **Offset in ASCII character units (8** **Number of 16-bit (1-word) or 32-bit units** (nähere Erläuterungen finden Sie in den nachstehenden Tabellen)
d_BinaryData	ANY	Anfangsbereich zum Speichern der Binärwerte

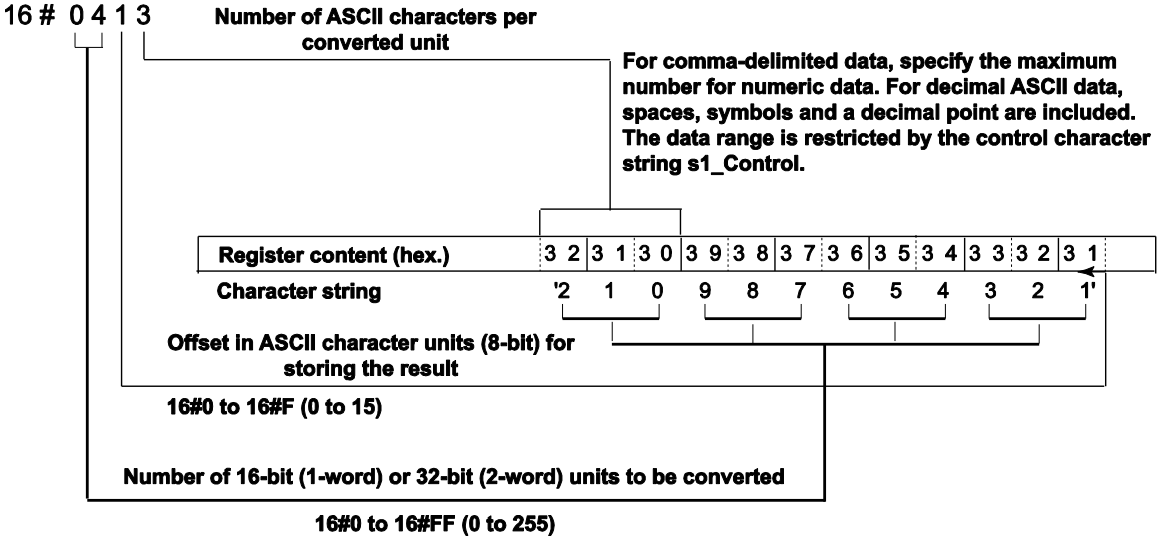
Operanden

Für	Merker				T/C		Register			Konstante
s1_Control	WX	WY	WR	WL	SV	EV	DT	LD	-	-
s2_AsciiData	WX	WY	WR	WL	SV	EV	DT	LD	-	-
n_Conversion Method	-	WY	WR	WL	SV	EV	DT	LD	-	dez., hex.
d_BinaryData	-	WY	WR	WL	SV	EV	DT	LD	-	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ das Steuerzeichen **s1_Control** fehlerhaft ist. ▪ die Standardrichtung (+) in **s1_Control** angegeben, und das Format dezimal ist. ▪ die Anzahl der ASCII-Zeichen pro umgewandelter Einheit, angegeben in **n_ConversionMethod**, für 16-Bitwerte den Wert 4 und für 32-Bitwerte den Wert 8 übersteigt und das hexadezimale Format durch **s1_Control** angegeben ist. ▪ 0 als Anzahl der 16- oder 32-Bit-Einheiten (1 oder 2 Worte) angegeben ist, die mit **n_ConversionMethod** umgewandelt werden sollen. ▪ die Anzahl der mit **n_ConversionMethod** umzuwandelnden 16- oder 32-Bit Dezimalzahlen den Speicherbereich der ASCII-Werte übersteigt. ▪ das Ergebnis der Umwandlung den Bereich überschreitet.
R9008	%MX0.900.8	kurzzeitig	

■ **Erläuterung der Umwandlungsmethode, z.B. n_ConversionMethod = 16#0413 für die ASCII-Werte '0123456789012'**



Umwandlungsbeispiele für die ASCII-Werte '0123456789ABCDEF'

n_Conversion method	s1_Control	ASCII-Werte	Binärwerte			Eräuterung
			Datentyp	Offs. in 16-Bit-Wort-einheiten	Hex. Wert	
H+16	16#404	0123 4567 89AB CDEF	INT, WORD	0	16#2301	Standardrichtung 4 x 4 ASCII-Zeichen
				1	16#6745	
				2	16#AB89	
				3	16#EFCD	
H+16	16#404		INT, WORD	0	6#0123	Umgekehrte Richtung 4 x 4 ASCII-Zeichen
				1	16#4567	
				2	16#89AB	
				3	16#CDEF	
H+16	16#403		INT, WORD	0	16#*201	Standardrichtung 3 x 4 ASCII-Zeichen
				1	16#*534	
				2	16#*867	
				3	16#*B9A	
H-16	16#403		INT, WORD	0	16#*012	Umgekehrte Richtung 3 x 4 ASCII-Zeichen
				1	16#*345	
				2	16#*678	
				3	6#*9AB	
H+32	16#208		DINT, DWORD	0	16#67452301	Standardrichtung 8 x 2 ASCII-Zeichen
				2	16#EFCDAB89	
H-32	16#208		DINT, DWORD	0	16#01234567	Umgekehrte Richtung 8 x 2 ASCII-Zeichen
				2	16#89ABCDEF	
H+32	16#205		DINT, DWORD	0	16#***42301	Umgekehrte Richtung 8 x 2 ASCII-Zeichen
				2	16#***97856	
H-32	16#205		DINT, DWORD	0	16#***01234	Umgekehrte Richtung 5 x 2 ASCII-Zeichen
				2	16#***56789	

*Die extra Zeichen werden '0'.

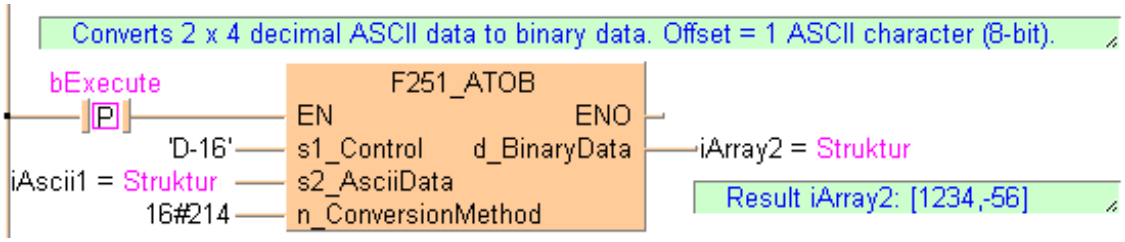
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bExecute	BOOL	FALSE
1	VAR	iArray1	ARRAY [0..1] OF INT	[2(0)]
2	VAR	iAscii1	ARRAY [0..4] OF WORD	[5(16#0)]

Rumpf Wenn **bExecute** auf TRUE gesetzt ist, wird der Befehl ausgeführt. Wandelt 2 x 4 dezimale ASCII-Zeichen in Binärwerte um. Offset = 1 ASCII-Zeichen (8-Bit)

LD



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

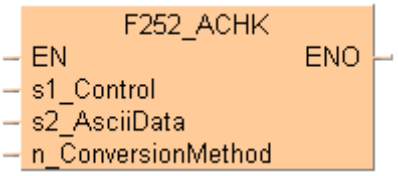
```
IF DF(bExecute) THEN
    F251_ATOB(s1_Control := 'D-16',
    s2_AsciiData := iAscii1,
    n_ConversionMethod := 16#214,
    d_BinaryData => iArray2);
END_IF;
```

F252_ACHK

ASCII-Datenprüfung

Erklärung

Prüft, ob die ASCII-Codes, die im Bereich **s2_AsciiData** gespeichert sind, sich korrekt mit der durch **n_ConversionMethod** festgelegten Umwandlungsmethode und den vier Steuerzeichen, definiert durch **s1_Control**, umwandeln lassen.



- Wenn die Ergebnisse korrekt sind, wird der Sondermerker (R900B) auf EIN gesetzt.
- Wenn die Ergebnisse nicht korrekt sind, wird der Sondermerker (R900B) auf AUS gesetzt.

Detaillierte Erläuterungen zu **s1_Control** und **n_ConversionMethod** finden Sie unter F251_ATOB.

SPS-Typen **Verfügbarkeit von F252_ACHK (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1_Control	STRING	Steuerzeichen **16 — D** **D: converts to decimal ASCII data** **H: converts to hexadecimal ASCII data** **+ Normal direction** **- Reverse direction** **16: converts in 16-bit (1-word) units** **32: converts in 32-bit (2-word) units**
s2_AsciiData	ANY	Anfangsbereich zum Speichern der ASCII-Werte
n_Conversion Method	ANY16	Konstante oder 16-Bit-Bereich zum Speichern der Umwandlungsmethode

Operanden

Für	Merker				T/C		Register			Konstante
s1_Control	WX	WY	WR	WL	SV	EV	DT	LD	-	-
s2_AsciiData	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
n_Conversion Method	WX	WY	WR	WL	SV	EV	DT	LD	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ das Steuerzeichen **s1_Control** fehlerhaft ist. ▪ die Standardrichtung (+) in **s1_Control** angegeben, und das Format dezimal ist. ▪ die Anzahl der ASCII-Zeichen pro umgewandelter Einheit, angegeben in **n_ConversionMethod**, für 16-Bitwerte den Wert 4 und für 32-Bitwerte den Wert 8 übersteigt und das hexadezimale Format durch **s1_Control** angegeben ist. ▪ 0 als Anzahl der 16- oder 32-Bit-Einheiten (1 oder 2 Worte) angegeben ist, die mit **n_ConversionMethod** umgewandelt werden sollen. ▪ die Anzahl der mit **n_ConversionMethod** umzuwandelnden 16- oder 32-Bit Dezimalzahlen den Speicherbereich der ASCII-Werte übersteigt. ▪ das Ergebnis der Umwandlung den Bereich überschreitet.
R9008	%MX0.900.8	kurzzeitig	

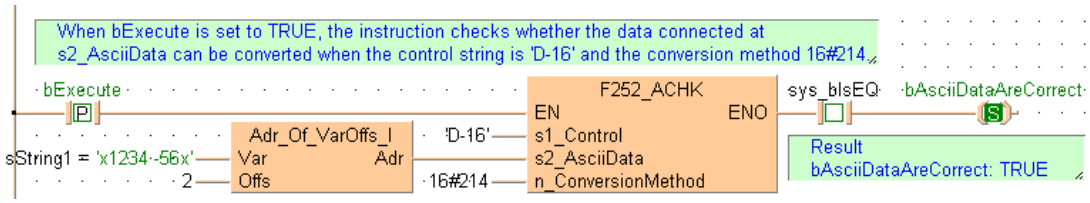
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bExecute	BOOL	FALSE
1	VAR	sString1	STRING[10]	'1234567890'
2	VAR	bAsciiDataAreCorrect	BOOL	FALSE

Rumpf Wenn **bExecute** auf TRUE gesetzt ist, prüft der Befehl, ob die mit **s2_AsciiData** verbundenen Werte sich mit dem Steuerzeichen 'D-16' und der Umwandlungmethode 16#214 umwandeln lassen.

LD



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(bExecute) THEN
    F252_ACHK(s1_Control := 'D-16', s2_AsciiData := Adr_Of_VarOffs(Var
:= sString1, Offs := 2), n_ConversionMethod := 16#214);
    IF (sys_bIsEQ) THEN
        bAsciiDataAreCorrect := TRUE;
    END_IF;
END_IF;
```

F325_FLT

16-Bit Integer Data to Floating Point Data Conversion

Erklärung Die Funktion wandelt den 16-Bit-Wert vom Typ INT, der durch **s** festgelegt wird, in einen Fließkommawert um. Die konvertierte Daten werden in Speicherregister **d** abgelegt.

- **Verwenden Sie F325_FLT nicht in Interrupt-Programmen.**
- **Anstelle von F325_FLT können Sie für den Datentyp REAL auch den vielseitigeren Befehl INT_TO_REAL (s. S. 193) verwenden.**

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F325_FLT (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	INT	16-Bit-Ganzzahl (Quellwert)
d	REAL	Fließkommawert (Zielwert)

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ das Ergebnis "0" ist.

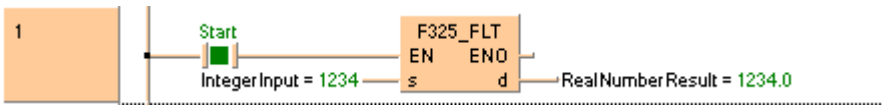
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Start	BOOL	FALSE
1	VAR	IntegerInput	INT	1234
2	VAR	RealNumberResult	REAL	0.0

Rumpf Wenn die Variable **Start** auf TRUE gesetzt wird, wandelt die Funktion die 16-Bit-Ganzzahl, die der Variablen **IntegerInput** zugewiesen wurde, in eine Fließkommazahl um. Das Ergebnis wird in den Adressbereich geschrieben, den der Compiler der Variablen **RealNumberResult** zugewiesen hat. Das Monitorwertsymbol ist sowohl für den KOP- als auch den AWL-Rumpf aktiviert.

KOP



F326_DFLT

32-Bit Integer Data to Floating Point Data Conversion

Erklärung Die Funktion wandelt den 32-Bit-Wert vom Typ INT, der durch **s** festgelegt wird, in einen Fließkommawert um. Die konvertierte Daten werden in Speicherregister **d** abgelegt.

- **Verwenden Sie F326_DFLT nicht in Interrupt-Programmen.**
- **Anstelle von F326_DFLT können Sie für den Datentyp REAL auch den vielseitigeren Befehl DINT_TO_REAL (s. S. 194) verwenden.**

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F326_DFLT (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	DINT	32-Bit-Ganzzahl (Quellwert)
d	REAL	Fließkommawert (Zielwert)

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ es zu viele signifikante Stellen in der Mantisse der konvertierten Fließkommazahl gibt.
R9009	%MX0.900.9	kurzzeitig	▪ das Ergebnis "0" ist.

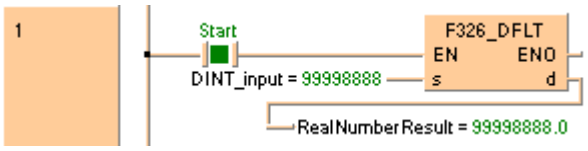
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Start	BOOL	FALSE
1	VAR	DINT_input	DINT	99998888
2	VAR	RealNumberResult	REAL	0.0

Rumpf Wenn die Variable **Start** auf TRUE gesetzt wird, wandelt die Funktion die 32-Bit-Ganzzahl, die der Variablen **DINT_input** zugewiesen wurde, in eine Fließkommazahl um. Das Ergebnis wird in den Adressbereich geschrieben, den der Compiler der Variablen **RealNumberResult** zugewiesen hat. Das Monitorwertsymbol ist sowohl für den KOP- als auch den AWL-Rumpf aktiviert.

KOP

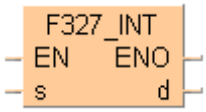

1
Start
DINT_input = 99998888
F326_DFLT
EN
ENO
s
d
RealNumberResult = 99998888.0

F327_INT

Fließkommawert zu 16-Bit-Ganzzahl-Wert (größter Ganzzahl-Wert überschreitet den Fließkommawert nicht)

Erklärung

Die Funktion konvertiert einen Fließkommawert am Eingang **s** im Bereich von -32767,99 bis 32767.99 in einen Ganzzahl-Wert (mit Vorzeichen). Das Funktionsergebnis wird am Ausgang **d** zurückgegeben.



Der konvertierte Ganzzahl-Wert am Ausgang **d** ist immer kleiner oder gleich groß wie der Fließkommawert am Eingang **s**:

- Wenn am Eingang ein positiver Fließkommawert steht, wird am Ausgang der positive Vorkomma-Wert zurückgegeben.
- Wenn am Eingang ein negativer Fließkommawert steht, wird am Ausgang der nächst kleinere Vorkomma-Wert zurückgegeben.
- Hat der negative Fließkommawert in den Nachkommastellen nur Nullen, so wird seine Vorkommastelle zurückgegeben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F327_INT (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	REAL	Quell-REAL-Zahl-Daten (2 Worte)
d	INT, WORD	Ziel zum Speichern der konvertierten Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert am Eingang **s** keine REAL-Zahl ist, oder das konvertierte Ergebnis den 16-Bit-Bereich am Ausgang **d** überschreitet.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	kurzzeitig	▪ das Funktionsergebnis Null ist.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

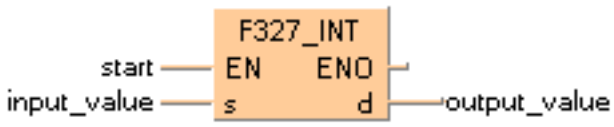
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	REAL	-1.234	
2	VAR	output_value	INT	0	result: here -2

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie konvertiert den Fließkommawert -1.234 in den Ganzzahlwert -2, der am Ausgang der Variablen **output_value** übergeben wird. Da die Ganzzahl den Fließkommawert nicht überschreiten darf, wird hier abgerundet.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F327_INT(input_value, output_value);
END_IF;
```

F328_DINT

Fließkommawert zu 32-Bit-Ganzzahl-Wert (größter Ganzzahl-Wert überschreitet den Fließkommawert nicht)

Erklärung

Die Funktion konvertiert einen Fließkommawert am Eingang **s** im Bereich von -2147483000 bis 32767.99 in einen Ganzzahl-Wert (mit Vorzeichen). Das Funktionsergebnis wird am Ausgang **d** zurückgegeben.

Der konvertierte Ganzzahl-Wert am Ausgang **d** ist immer kleiner oder gleich groß wie der Fließkommawert am Eingang **s**:

- Wenn am Eingang ein positiver Fließkommawert steht, wird am Ausgang der positive Vorkomma-Wert zurückgegeben.
- Wenn am Eingang ein negativer Fließkommawert steht, wird am Ausgang der nächst kleinere Vorkomma-Wert zurückgegeben.
- Hat der negative Fließkommawert in den Nachkommastellen nur Nullen, so wird seine Vorkommastelle zurückgegeben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F328_DINT (s. S. 1190)

Datentypen

Variable	Datentyp	Funktion
s	REAL	Quell-REAL-Zahl-Daten (2 Worte)
d	DINT, DWORD	Ziel zum Speichern der konvertierten Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert am Eingang **s** keine REAL-Zahl ist, oder das konvertierte Ergebnis den 32-Bit-Bereich am Ausgang **d** überschreitet.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	kurzzeitig	▪ das Funktionsergebnis Null ist.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

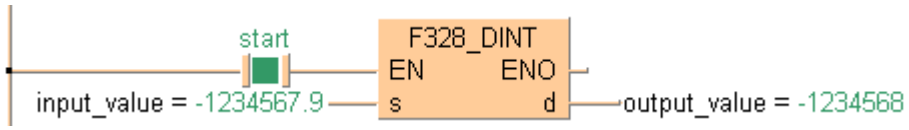
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	REAL	-1234567.89	
2	VAR	output_value	DINT	0	result: here -1234568

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie konvertiert den Fließkommawert -1234567.89 in den Ganzzahlwert -1234568, der am Ausgang der Variablen **output_value** übergeben wird. Da die Ganzzahl den Fließkommawert nicht überschreiten darf, wird hier abgerundet.

KOP



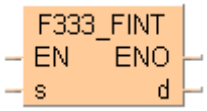
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F328_DINT(input_value, output_value);
END_IF;
```

F333_FINT

Fließkommawert - Erste Nachkommastelle wird abgerundet

Erklärung Die Funktion rundet einen Fließkommawert am Eingang s ab und gibt das Funktionsergebnis am Ausgang **d** zurück.



Der konvertierte Ganzzahl-Wert am Ausgang **d** ist immer kleiner oder gleich groß wie der Fließkommawert am Eingang **s**:

- Wenn am Eingang ein positiver Fließkommawert steht, wird am Ausgang der positive Vorkomma-Wert zurückgegeben.
- Wenn am Eingang ein negativer Fließkommawert steht, wird am Ausgang der nächste kleine Vorkomma-Wert zurückgegeben.
- Hat der negative Fließkommawert in den Nachkommastellen nur Nullen, so wird seine Vorkommastelle zurückgegeben.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F333_FINT (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	REAL	Quelle
d	REAL	Ziel

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert am Eingang **s** keine REAL-Zahl ist.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

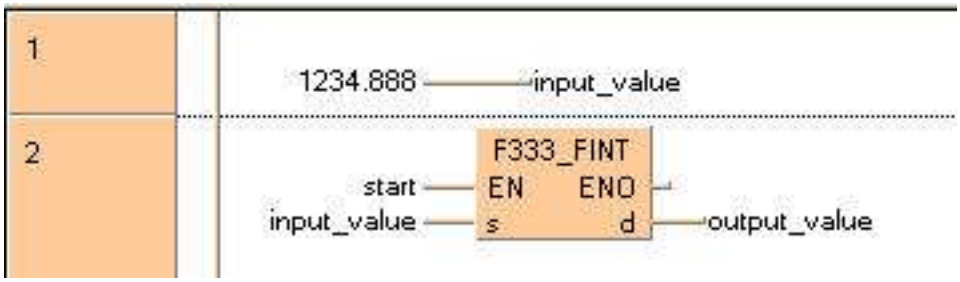
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	REAL	0.0	
2	VAR	output_value	REAL	0.0	result: here 1234.000

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Der Wert 1234.888 wird der Variablen **input_value** zugeordnet. Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie rundet den Wert der Variablen **input_value** nach der ersten Nachkommastelle ab und gibt das Funktionsergebnis (hier: 1234.000) an die Variable **output_value**.

KOP



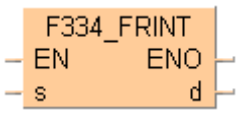
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
input_value:=1234.888;
IF start THEN
    F333_FINT(input_value, output_value);
END_IF;
```

F334_FRINT

Fließkommawert - Erste Nachkommastelle wird gerundet

Erklärung Die Funktion rundet einen Fließkommawert am Eingang s und gibt das Funktionsergebnis am Ausgang **d** zurück.



Wenn die erste Nachkommastelle zwischen 0..4 liegt, wird auf den nächstkleineren Vorkommawert abgerundet. Wenn die erste Nachkommastelle zwischen 5..9 liegt, wird auf den nächstgrößeren Vorkommawert aufgerundet.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F334_FRINT (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	REAL	Quelle
d	REAL	Ziel

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert am Eingang **s** keine REAL-Zahl ist. ▪
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

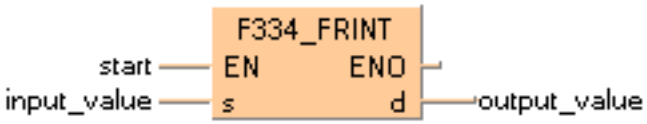
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	REAL	1234.567	
2	VAR	output_value	REAL	0.0	result: here 1235.000

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie rundet den Wert der Variablen **input_value** = 1234.567 nach der ersten Nachkommastelle auf und gibt das Funktionsergebnis (hier: 1235.000) an die Variable **output_value**.

KOP



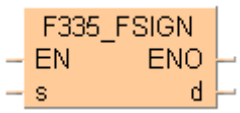
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F334_FRINT(input_value, output_value);
END_IF;
```

F335_FSIGN

Fließkommawert Vorzeichenwechsel

Erklärung Die Funktion wechselt das Vorzeichen von einem Fließkommawert am Eingang **s** und gibt das Funktionsergebnis am Ausgang **d** zurück.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F335_FSIGN (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	REAL	Quelle
d	REAL	Ziel

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert am Eingang **s** keine REAL-Zahl ist.
R9008	%MX0.900.8	kurzzeitig	
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

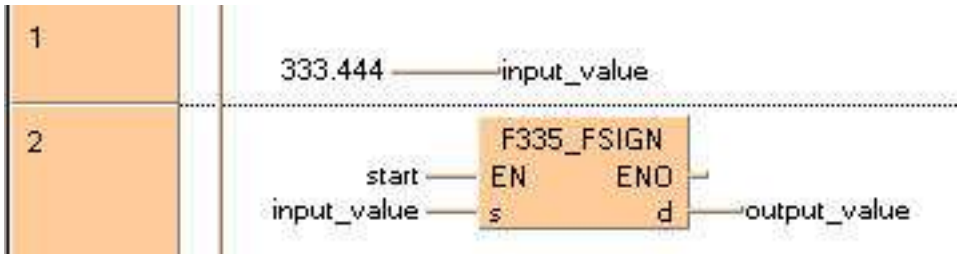
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	REAL	0.0	
2	VAR	output_value	REAL	0.0	result: here -333.444

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Der Wert 333,4 wird der Variablen **input_value** zugeordnet. Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Der Ausgangswert **output_value** ist dann -333.4.

KOP



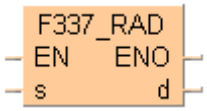
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
input_value:=333.444;
IF start THEN
    F335_FSIGN(input_value, output_value);
END_IF;
```

F337_RAD

Grad der Radiant Konvertierung

Erklärung Die Funktion konvertiert die Einheit eines Winkelwerts am Eingang **s** von Grad in Radiant und gibt das Funktionsergebnis am Ausgang **d** zurück.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F337_RAD (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	REAL	Quell-Winkel-Daten (Grad), 2 Worte
d	REAL	Ziel zum Speichern der konvertierten Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert am Eingang **s** keine REAL-Zahl ist.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

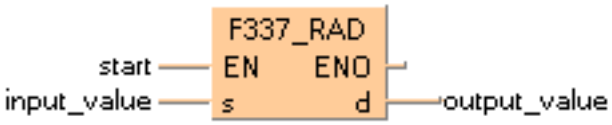
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	REAL	180.0	angle in °
2	VAR	output_value	REAL	0.0	angle in radians
3	VAR				result: here 3.14159

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



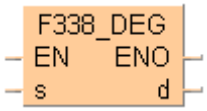
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F337_RAD(input_value, output_value);
END_IF;
```

F338_DEG

Grad der Radiant Konvertierung

Erklärung Die Funktion konvertiert die Einheit eines Winkelwerts am Eingang **s** von Radiant in Grad und gibt das Funktionsergebnis am Ausgang **d** zurück.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F338_DEG (s. S. 1190)**

Datentypen

Variable	Datentyp	Funktion
s	REAL	Quell-Winkel-Daten (Radiant), 2 Worte
d	REAL	Ziel zum Speichern der konvertierten Daten

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert am Eingang **s** keine REAL-Zahl ist.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	auf TRUE	▪ das Funktionsergebnis Null ist.
R9009	%MX0.900.9	kurzzeitig	▪ das Funktionsergebnis einen Überlauf verursacht.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

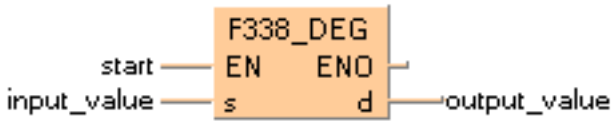
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	REAL	3.14159	angle in radians
2	VAR	output_value	REAL	0.0	angle in ° result: here 180.0
3	VAR				

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F338_DEG(input_value, output_value);
END_IF;
```

Kapitel 20

Datenübertragung über Kommunikationsschnittstellen

20.1 Einleitung

Die Panasonic-Steuerungen der FP-Serie bieten folgende Kommunikationsarten für die Datenübertragung über die Kommunikationsschnittstellen:

- Programmgesteuerter Modus: für Datenübertragung zwischen SPS und anderen Geräten
- Master/Slave-Modus (MEWTOCOL-COM/Modbus-RTU): für die Kommunikation mit einem Computer oder einem Bediengerät
- SPS-Kopplung: (s. S. 742) zur gemeinsamen Datenhaltung in mehreren SPSen der FP-Serie

Bevor Daten übertragen werden können, müssen Sie die Kommunikationsparameter (s. S. 683) in der SPS einstellen. Es ist möglich, diese Einstellungen im RUN-Modus zu prüfen (s. S. 690).

20.2 Kommunikationsparameter einstellen

Die Kommunikationsart und andere Kommunikationsparameter werden in den Systemregistern der SPS eingestellt.

Für die einzelnen Kommunikationsarten sind unterschiedliche Einstellungen erforderlich

Verwenden Sie die Programmiersoftware oder DIP-Schalter an der SPS, um Einstellungen im PROG-Modus vorzunehmen. Oder verwenden Sie FP-Befehle, um die Parameter im RUN-Modus zu ändern.

Wählen Sie eine geeigneten Einstellungsmöglichkeiten für CPU oder MCU:

Parameter	PROG-Modus	RUN-Modus	CPU	MCU
Kommunikationsart (s. S. 681)	Systemregister (s. S. 683)	–	●	●
	–	SetCommunicationMode (s. S. 683)	●	●
	–	F159_MTRN (n_Number=16#8000) (s. S. 685)	●	●
Sonstige Parameter (z.B. Baudrate, Teilnehmeradresse, Start- und Endezeichen, Empfangspuffer)	Systemregister (s. S. 683) oder DIP-Schalter	–	●	
	–	(nur SYS1 (s. S. 1004), SYS2 (s. S. 1016) (FPΣ, FP-X)	●	
	MCU-Dialogfeld oder DIP-Schalter (s. S. 689)	–		●
	–	F159_MWRT_PARA (s. S. 687)		●

20.2.1 Kommunikationsparameter in den Systemregistern einstellen

◆Vorgehensweise

1. **Im Navigator auf "SPS" doppelklicken**
2. **Auf “Systemregister” doppelklicken**
3. **Auf "COM-Schnittstelle" doppelklicken**

   Da die Kommunikationsschnittstellen unterschiedliche Bitpositionen desselben Systemregisters belegen, sind individuelle Einstellungen für jede Schnittstelle möglich.

   Die Einstellungen für die TOOL-Schnittstelle werden in den Systemregistern unter "TOOL-Schnittstelle" vorgenommen.

   Die Nummern der Systemregister sind nicht für alle SPS-Typen gleich.

4. **Nehmen Sie die Einstellungen für die Kommunikationsart, das Kommunikationsformat, die Baudrate, die Teilnehmernummer und den Empfangspuffer vor, wenn erforderlich.**

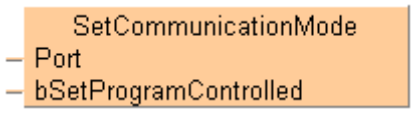
SetCommunication Mode

Kommunikationsart einstellen

Erklärung

Je nachdem, welcher Wert an **bSetProgramControlled** anliegt, stellt dieser Befehl die Kommunikationsart "programmgesteuert" oder "MEWTOCOL-COM Slave" ein.

- TRUE: Programmgesteuerter Modus
- FALSE: MEWTOCOL-COM Slave [PC-Kopplung]



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

- **Beim Einschalten der Steuerung wird die in den Systemregistern festgelegte Kommunikationsart eingestellt.**
- **Eine Änderung der Kommunikationsarten Modbus RTU oder SPS-Kopplung ist im RUN-Modus nicht möglich.**

SPS Typen

siehe S. 1196

Datentypen

Variable	Datentyp	Beschreibung
Port	INT	Kommunikationsschnittstellen Für **FP-X**, **FP$_{\Sigma}$** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** — **MCU:** SYS_COM1_PORT — 16#xx01 (COM1) SYS_COM2_PORT — 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)
bSetProgramControlled		Stellt die Kommunikationsart ein: ▪ TRUE: Programmgesteuerter Modus ▪ FALSE: MEWTOCOL-COM Slave [PC-Kopplung]

Beispiel

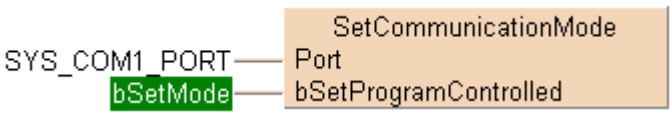
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bSetMode	BOOL	FALSE	*If TRUE, communicatio mode es set to 'Program controlled'*

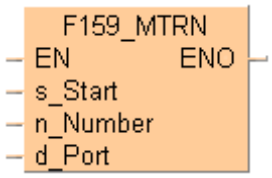
KOP



ST
```
SetCommunicationMode(Port := SYS_COM1_PORT,
                     bSetProgramControlled := bSetMode);
```

F159_MTRN Kommunikationsart wechseln

Erklärung Die Kommunikationsart der CPU-Schnittstellen kann im RUN-Modus geändert werden. Um zwischen programmgesteuerter Kommunikation und MEWTOCOL-COM-Modus umzuschalten, führen Sie F159_MTRN (s. S. 703) aus und setzen die Variable **n_Number** (Anzahl der zu sendenden Bytes) auf 16#8000.



SPS Typen siehe (s. S. 1187)

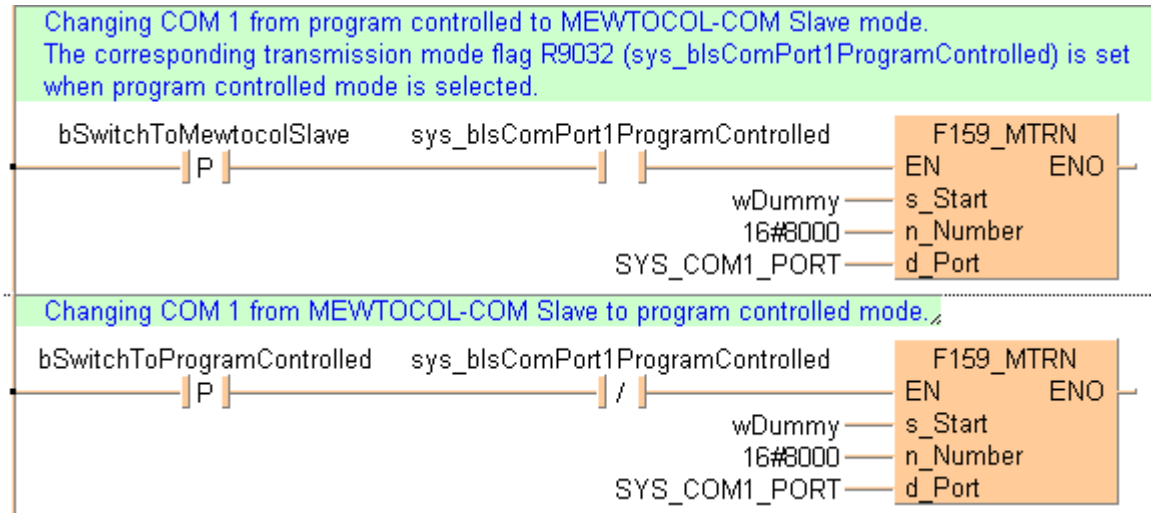
- **Beim Einschalten der Steuerung wird die in den Systemregistern festgelegte Kommunikationsart eingestellt.**
- **Eine Änderung der Kommunikationsarten Modbus RTU oder SPS-Kopplung ist im RUN-Modus nicht möglich.**

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bSwitchtoMewtocolSlave	BOOL	FALSE
1	VAR	wDummy	WORD	0
2	VAR	bSwitchtoProgramControlled	BOOL	FALSE

KOP

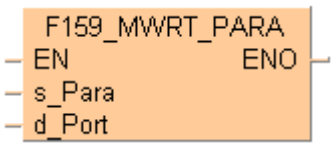


Der Merker "Kommunikationsart" wird auf TRUE gesetzt, wenn der Modus "programmgesteuerte Kommunikation" aktiv ist. Er wird auf FALSE gesetzt, wenn der Modus "MEWTOCOL-COM" eingestellt ist. Der Merker kann mit der Systemvariablen sys_bIsComPort1ProgramControlled, sys_bIsComPort2ProgramControlled oder sys_bIsToolPortProgramControlled ausgewertet werden.

ST
```
(* Changing COM 1 from program controlled to MEWTOCOL-COM Slave mode.
   The corresponding transmission mode flag R9032
(sys_bIsComPort1ProgramControlled) is set
   when program controlled mode is selected. *)
if (DF(bSwitchToMewtocolSlave) AND sys_bIsComPort1ProgramControlled) then
   F159_MTRN(s_Start := wDummy, n_Number := 16#8000, d_Port :=
SYS_COM1_PORT);
end_if;

(* Changing COM 1 from MEWTOCOL-COM Slave to program controlled *)
if (DF(bSwitchToProgramControlled) AND NOT
sys_bIsComPort1ProgramControlled) then
   F159_MTRN(s_Start := wDummy, n_Number := 16#8000, d_Port :=
SYS_COM1_PORT);
end_if;
```

F159_MTRN — Kommunikationsparameter im RUN-Modus setzen

Erklärung Die Kommunikationsparameter im vordefinierten SDT MCU_PARA_DUT werden in die angegebene Schnittstelle eines MCU-Moduls geschrieben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

☞ **Die Einstellungen im MCU-Konfigurator sind nur aktiv, wenn die DIP-Schalter 3 und 4 (für COM1) bzw. 7 und 8 (für COM2) des MCU-Moduls auf ON stehen.**

SPS Typen siehe (s. S. 1187)

Datentypen

Variable	Datentyp	Beschreibung
s_Para	MCU_PARA_DUT	Im vordefinierten SDT festgelegte Kommunikationsparameter
d_Port	ANY16	Steckplatznummer (höherwertiges Byte) und Schnittstellennummer (niederwertiges Byte) des MCU-Moduls, an das die Daten übertragen werden. 16#xx01: COM1 an MCU-Modul in Steckpatz 16#xx 16#xx02: COM2 an MCU-Modul in Steckpatz 16#xx

Operanden

Für	Merker				T/C		Register			Konstante
s_Para	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
d_Port	-	WY	WR	WL	SV	EV	DT	LD	FL	dez. oder hex.

Fehlermerker

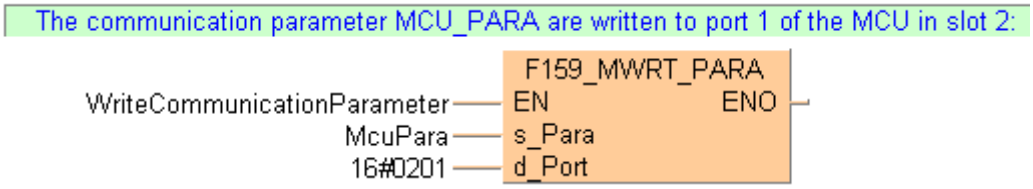
Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	dauerhaft	das MCU-Modul sich nicht im angegebenen Steckplatz befindet
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bWriteCommunicationPara...	BOOL	FALSE
1	VAR	McuPara	MCU_PARA_DUT	

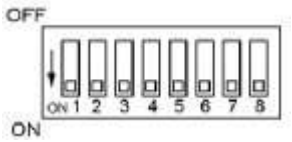
KOP



```
ST (*The 10 bytes beginning with Data are sent from port 1 of the MCU in slot
   2:*)
   if WriteCommunicationParameter then
      F159_MWRT_PARA(s_Para := McuPara, d_Port := 16#0201);
   end_if;
```

20.2.2 Einstellen der Kommunikationsart über die DIP-Schalter (FP2/FP2SH)

Die DIP-Schalter zur Festlegung der Kommunikationsart und der Baudrate befinden sich auf der Rückseite des Gehäuses.



	Schnittstelle	COM 1				COM 2			
	Schalternr.	SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8
Kommunikationsart	(nicht belegt)	AUS	AUS	-	-	AUS	AUS	-	-
	SPS-Kopplung	EIN	AUS	-	-	EIN	AUS	-	-
	Programmge-steuerter Modus	AUS	EIN	-	-	AUS	EIN	-	-
	MEWTOCOL-COM-Slave-Modus	EIN	EIN	-	-	EIN	EIN	-	-
Baudrate	115200 bit/s	-	-	AUS	AUS	-	-	AUS	AUS
	19200 bit/s	-	-	EIN	AUS	-	-	EIN	AUS
	9600 bit/s	-	-	AUS	EIN	-	-	AUS	EIN
	Wert aus MCU-Konfigurator	-	-	EIN	EIN	-	-	EIN	EIN

◆HINWEIS

Sämtliche DIP-Schalter stehen ab Werk auf "EIN".

20.3 Kommunikationsparameter lesen

Verwenden Sie FP-Befehle oder Systemvariablen, um die Kommunikationseinstellungen der SPS oder des MCU-Moduls im RUN-Modus zu prüfen.

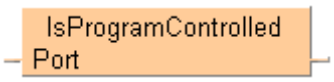
Wählen Sie eine der folgenden Möglichkeit zum Lesen der Parameter im RUN-Modus:

Parameter		Methode	CPU	MCU
Kommunikationsart (s. S. 681)	Programmgesteuerter Modus MEWTOCOL-COM-Master/Slave	IsProgramControlled (s. S. 691)	●	●
		sys_bIsToolPortProgramControlled	●	
		sys_bIsComPort1ProgramControlled	●	
		sys_bIsComPort2ProgramControlled	●	
		F161_MRD_STATUS (s. S. 695)		●
	SPS-Kopplung	IsPlcLink (s. S. 692)	●	●
		sys_bIsComPort1PlcLink	●	
		F161_MRD_STATUS (s. S. 695)		●
Sonstige Parameter (z.B. Baudrate, Teilnehmeradresse, Start- und Endezeichen)		F161_MRD_PARA (s. S. 693)		●
		F161_MRD_STATUS (s. S. 695)		●

IsProgramControlled

Liefert den Wert des Merkers "COM-Schnittstelle im Modus Programmgesteuerte Kommunikation"

Erklärung Dieser Befehl liefert den Wert des Merkers "Kommunikationsart". Der Merker "Kommunikationsart" ist TRUE, wenn für die Kommunikationsschnittstelle der SPS der Modus programmgesteuerte Kommunikation gewählt wurde. Der Merker ist FALSE, wenn "MEWTOCOL-COM Master/Slave" eingestellt wurde.



SPS Typen siehe (s. S. 1194)

Datentypen

Variable	Datentyp	Funktion
Port	ANY16	Kommunikationsschnittstellen Für **FP-X**, **FP$_\Sigma$** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iPort	INT	0
1	VAR	bProgramControlled	BOOL	FALSE

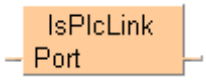
KOP

ST
```
bProgramControlled:=IsProgramControlled(Port := iPort);
```

IsPlcLink

Liefert den Wert des Merkers "COM-Schnittstelle im Modus SPS-Kopplung"

Erklärung Dieser Befehl liefert den Wert des Merkers "SPS-Kopplung". Der Merker ist TRUE, wenn für die Kommunikationsschnittstelle der SPS der Modus SPS-Kopplung gewählt wurde.



SPS Typen siehe (s. S. 1194)

Datentypen

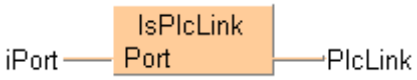
Variable	Datentyp	Funktion
Port	ANY16	Kommunikationsschnittstellen Für **FP-X**, **FP$_{\Sigma}$** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iPort	INT	0
1	VAR	PlcLink	BOOL	FALSE

KOP



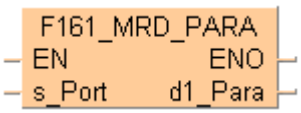
ST
```
PlcLink:=IsPlcLink(Port := iPort);
```

F161_MRD_PARA

Kommunikationsparameter im RUN-Modus von den MCU-Schnittstellen lesen

Erklärung

Die im strukturierten Datentyp MCU_PARA_DUT festgelegten Kommunikationsparameter werden von einer Schnittstelle des MCU-Moduls im angegebenen Steckplatz gelesen.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F161_MRD_PARA (s. S. 1188)

Datentypen

Variable	Datentyp	Beschreibung
s_Port	ANY16	Steckplatznummer (höherwertiges Byte) und Schnittstellennummer (niederwertiges Byte) des MCU-Moduls, an das die Daten übertragen werden. 16#xx01: COM1 an MCU-Modul in Steckplatz 16#xx 16#xx02: COM2 an MCU-Modul in Steckplatz 16#xx
d1_Para	MCU_PARA_DUT	Im vordefinierten SDT festgelegte Kommunikationsparameter

Operanden

Für	Merker				T/C		Register			Konstante
s_Port	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez. oder hex.
d1_Para	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	dauerhaft	▪ die mit Indexmodifizierern definierte Adresse den zulässigen Bereich überschreitet ▪ das MCU-Modul sich nicht im angegebenen Steckplatz befindet ▪ die angegebene Kommunikationsschnittstelle nicht vorhanden ist
R9008	%MX0.900.8	kurzzeitig	

Beispiel

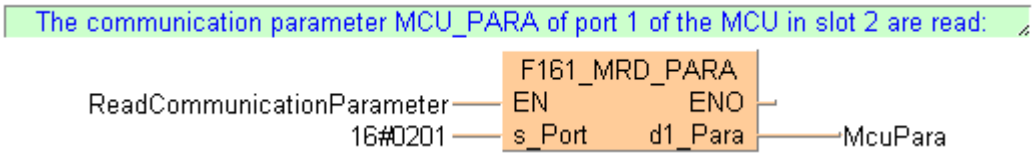
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	ReadCommunicationParameter	BOOL	FALSE
1	VAR	McuPara	MCU_PARA_DUT	

KOP

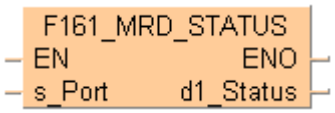


ST
```
(*The communication parameter MCU_PARA of port 1 of the MCU in slot 2 are read:*)
if (ReadCommunicationParameter) then
    F161_MRD_PARA(s_Port := 16#0201, d1_Para => McuPara);
end_if;
```

F161_MRD_STATUS

Statusdaten im RUN-Modus von den MCU-Schnittstellen lesen

Erklärung Statusdaten der angegebenen COM-Schnittstelle eines MCU-Moduls werden gelesen.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS Typen siehe (s. S. 1188)

Datentypen

Variable	Datentyp	Beschreibung
s_Port	ANY16	Steckplatznummer (höherwertiges Byte) und Schnittstellennummer (niederwertiges Byte) des MCU-Moduls, an das die Daten übertragen werden. 16#xx01: COM1 an MCU-Modul in Steckpatz 16#xx 16#xx02: COM2 an MCU-Modul in Steckpatz 16#xx
d1_Status	MCU_STATUS_DUT	Im vordefinierten SDT festgelegte Kommunikationsparameter

Operanden

Für	Merker				T/C		Register			Konstante
s_Port	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez. oder hex.
d1_Status	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermer-ker

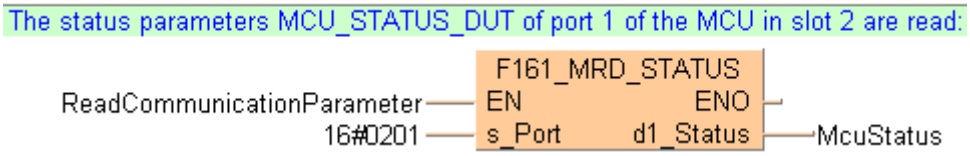
Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	dauerhaft	▪ die mit Indexmodifizierern definierte Adresse den zulässigen Bereich überschreitet ▪ das MCU-Modul sich nicht im angegebenen Steckplatz befindet ▪ die angegebene Kommunikationsschnittstelle nicht vorhanden ist
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	ReadCommunicationParameter	BOOL	FALSE
1	VAR	McuStatus	MCU_STATUS_DUT	

KOP

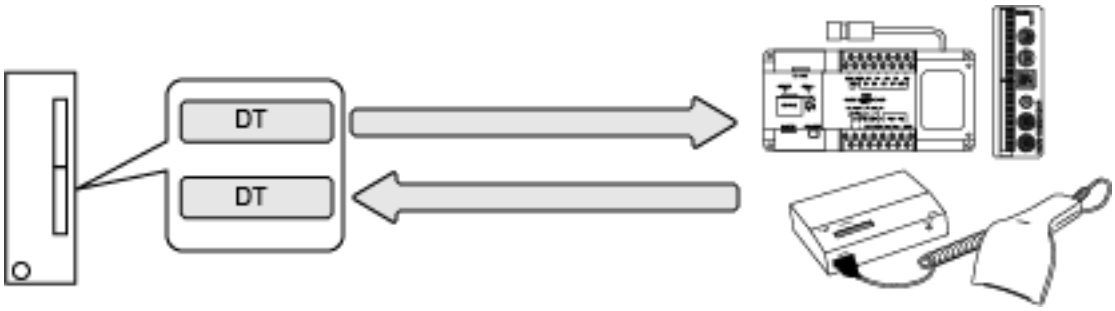


ST
```
(*The status parameters MCU_STATUS_DUT of port 1 of the MCU in slot 2 are read:*)
if (ReadCommunicationParameter) then
    F161_MRD_STATUS(s_Port := 16#0201, d1_Status => McuStatus);
end_if;
```

20.4 Programmgesteuerter Modus

Im programmgesteuerten Modus erstellt der Benutzer ein Programm, das die Datenübertragung zwischen einer SPS und einer oder mehreren an die COM-Schnittstelle angeschlossenen externen Geräten, z.B. einem Bildverarbeitungsgerät oder Strichcodeleser, steuert. Auf diese Weise lassen sich sowohl Standardprotokolle als auch benutzerdefinierte Protokolle programmieren.

Ein solches benutzerdefiniertes Programm umfasst in der Regel das Senden und Empfangen der Daten. Die sendebereiten Daten und die empfangenen Daten werden in jenen Datenregisterbereichen (DT) gespeichert, die als Sende- und Empfangspuffer definiert wurden.



20.4.1 Daten senden

Beim Senden werden die Daten für den Sendepuffer generiert und mit den Befehlen SendCharacters (s. S. 698), SendCharactersAndClearString (s. S. 700) oder F159_MTRN (s. S. 703) gesendet. Die im Systemregister angegebenen Start- und Endezeichen werden automatisch an die gesendeten Daten gehängt. Die maximale Anzahl von Bytes, die übertragen werden können, ist 2048.

Ablauf des Sendevorgangs:

- **Schritt 1:** Kommunikationsparameter einstellen (s. S. 683)

  Erforderliche Einstellungen: Kommunikationsart (programmgesteuert), Baudrate, Kommunikationsformat
- **Schritt 2:** Sendepuffer generieren (s. S. 698)

  Nicht notwendig, wenn SendCharacters (s. S. 698) oder SendCharactersAndClearString (s. S. 700) verwendet wird.
- **Schritt 3:** Sendebefehl ausführen

  Verwenden Sie einen der folgenden Befehle:

Befehl	Beschreibung
SendCharacters (s. S. 698)	geeignet für 90% aller Applikationen, möglicheweise speicherintensiver
SendCharactersAndClearString (s. S. 700)	Wie SendCharacters, aber funktioniert ohne Sendepuffer, jedoch ohne Sendepuffer und weniger speicherintensiv
F159_MTRN (s. S. 703)	Original FP-Befehl mit allen Parametern, Transferbefehl zum Schreiben der Daten in den Sendepuffer erforderlich

- **Schritt 4 (optional):** Merker "Senden beendet" auswerten
  Wählen Sie eine der folgenden Möglichkeiten:

Methode	Beschreibung
IsTransmissionDone (s. S. 707)	Gibt den Wert des Merkers "Senden beendet" zurück. Schaltet auf TRUE, wenn die angegebene Anzahl von Bytes gesendet wurde.
sys_bIsComPort1TransmissionDone sys_bIsComPort2TransmissionDone sys_bIsToolPortTransmissionDone	Diese Systemvariablen schalten auf TRUE, wenn die angegebene Anzahl von Bytes gesendet wurde.
Eingangsmerker X4 und X5 (nur MCU)	Diese Merker zeigen das Ende einer Datenübertragung am MCU-Modul an.

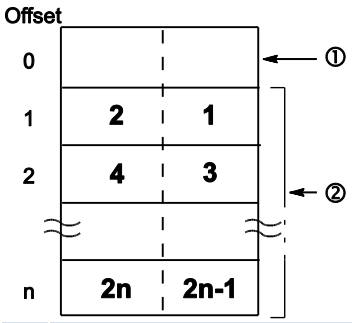
◆HINWEIS

- **Wenn die angegebene Anzahl von Bytes gesendet wurde, wird der Merker "Senden beendet" auf TRUE gesetzt. Die Auswertung des Merkers "Empfangen beendet" kann in Fällen sinnvoll sein, wo keine Antwort erwartet wird, z.B. bei Broadcast-Meldungen.**
- **Daten können nur gesendet werden, wenn das CS-Signal der COM-Schnittstelle (RS232C) gesetzt ist. Unterstützt die Gegenstation das CTS-Signal nicht (Dreidraht-Schnittstelle), müssen CS und RS der COM-Schnittstelle überbrückt werden.**
- **SPSen mit nur einer Schnittstelle kompilieren F159_MTRN in F144_TRNS (die Variable d_Port wird ignoriert).**

20.4.1.1 Sendepuffer generieren

Die Befehle SendCharacters (s. S. 698) und SendCharactersAndClearString (s. S. 700) erzeugen die Daten im Sendepuffer automatisch.

Aufbau des Sendepuffers



1	Speicherbereich für die Anzahl der zu sendenden Bytes
2	Speicherbereich für die zu sendenden Daten

Die eingekreisten Zahlen geben die Übertragungsreihenfolge an. Der Speicherbereich für die zu sendenden Daten beginnt mit dem zweiten Wort des Sendepuffers (Offset 1). Offset 0 enthält die Anzahl der zu sendenden Bytes. Die maximale Anzahl von Bytes, die übertragen werden können, ist 2048.

Wenn F159_MTRN (s. S. 703) für die Datenübertragung verwendet wird, müssen die Daten mit einem Transferbefehl in den Sendepuffer kopiert werden, z.B. F10_BKMV (s. S. 775).

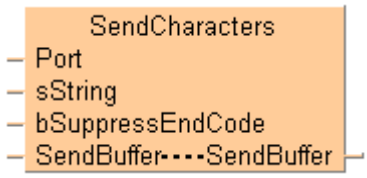
SendCharacters

Zeichen an CPU- oder MCU-Schnittstelle senden

Erklärung

Dieser Befehl füllt den Sendepuffer (s. S. 698) und führt F159_MTRN (s. S. 703) aus, um die Daten zu senden. Die zu sendenden Daten werden **sString.** zugewiesen. Der Sendepuffer ist eine VAR_INOUT-Variable, die **SendBuffer** zugewiesen wird. Wenn **bSuppressEndCode** auf TRUE gesetzt wird, wird das in den Systemregistern ausgewählte Endezeichen nicht an die zu übertragende Zeichenfolge angehängt.

Im Gegensatz zum Befehl **SendCharactersAndClearString** (s. S. 700) bleibt hier die **sString** zugewiesene Zeichenfolgenvariable unverändert.



Wenn die angegebene Anzahl von Bytes gesendet wurde, wird der Merker "Senden beendet" auf TRUE gesetzt. Die Auswertung des Merkers "Empfangen beendet" kann in Fällen sinnvoll sein, wo keine Antwort erwartet wird, z.B. bei Broadcast-Meldungen.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1196

Datentypen

Variable	Datentyp	Beschreibung
Port	ANY16	Kommunikationsschnittstellen Für **FP-X**, **FP$_{\Sigma}$** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)
sString	STRING	Speichert die zu übertragende Zeichenfolge
bSuppressEndCode	BOOL	Das in den Systemregistern gewählte Endezeichen wird nicht an die zu übertragende Zeichenfolge angehängt.
Ein-/Ausgangsvariable		
SendBuffer	ANY	Speichert die zu übertragende Zeichenfolge

Fehlermerker

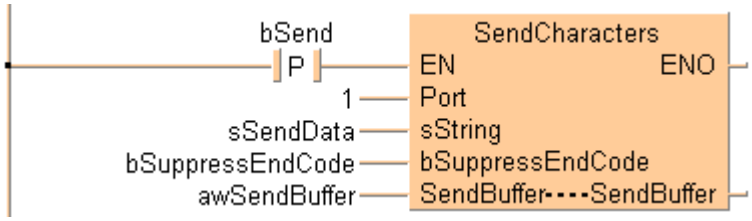
Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	dauerhaft	▪ das MCU-Modul sich nicht im angegebenen Steckplatz befindet ▪ 16#8000 wird im MEWTOCOL-COM Master/Slave-Modus festgelegt
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bSend	BOOL	FALSE	activates function
1	VAR	sSendData	STRING[30]	'ABCDEFGH'	up to 30 chars
2	VAR	awSendBuffer	ARRAY [0..15] OF WORD	[16(0)]	for 30 chars + 1 word
3	VAR	bDoNotAppendEndCode	BOOL	FALSE	

KOP Wenn **bSend** von FALSE auf TRUE wechselt, sendet der Befehl die Zeichen von **sSendData** an die MCU-Schnittstelle 1. Die Zeichen werden in den Array **awSendBuffer** kopiert. **awSendBuffer[0]** ist für die Länge der zu sendenden Zeichenfolge reserviert.



ST
```
if (DF(bSend)) then
    sResult:=SendCharacters(Port := 1, sString := sSendData,
bSuppressEndCode := bSuppressEndCode, SendBuffer := awSendBuffer);
end_if;
```

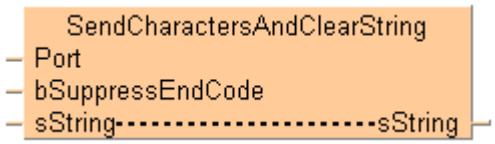
SendCharactersAndClearString

Zeichen senden und Zeichenfolge löschen

Erklärung

Dieser Befehl führt den Befehl F159_MTRN (s. S. 703) zum Senden der Daten aus. Die Daten sind in der zu übertragenden Zeichenfolge **sString**, einer VAR_INOUT-Variable, enthalten. Wenn **bSuppressEndCode** auf TRUE gesetzt wird, wird das in den Systemregistern ausgewählte Endezeichen nicht an die zu übertragende Zeichenfolge angehängt.

Im Gegensatz zu SendCharacters (s. S. 698) sind hier keine zusätzlichen Sendepuffer erforderlich. Die **sString** zugewiesene Variable wird nach der Ausführung gelöscht.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

☞ **Wenn die angegebene Anzahl von Bytes gesendet wurde, wird der Merker "Senden beendet" auf TRUE gesetzt. Die Auswertung des Merkers "Empfangen beendet" kann in Fällen sinnvoll sein, wo keine Antwort erwartet wird, z.B. bei Broadcast-Meldungen.**

SPS Typen s. S. 1196

Datentypen

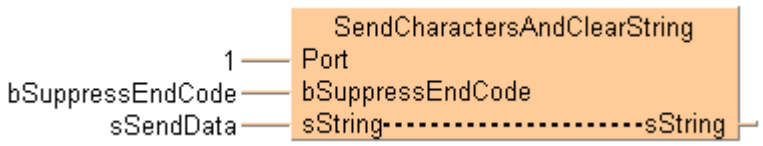
Variable	Datentyp	Beschreibung
Port	INT	Kommunikationsschnittstellen Für **FP-X**, **FP$_{\Sigma}$** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)
bSuppressEndCode	BOOL	Das in den Systemregistern gewählte Endezeichen wird nicht an die zu übertragende Zeichenfolge angehängt.
Ein-/Ausgangsvariable		
sString	STRING	Speichert die zu übertragende Zeichenfolge und löscht diese nach der Befehlsausführung

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Class	Identifier	Type	Initial	Comment
0	VAR	sSendData	STRING[30]	'ABCDEFGH'	up to 30 chars
1	VAR	bSuppressEndCode	BOOL	FALSE	

KOP



ST

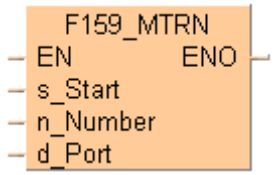
```
SendCharactersAndClearString(Port := 1,
                             bSuppressEndCode := bSuppressEndCode,
                             sString := sSendData);
```

F159_MTRN

Daten an CPU- oder MCU-Schnittstelle senden

Erklärung

Dieser Befehl sendet die Daten über einen Sendepuffer (s. S. 698) an externe Geräte (Computer, Messgeräte, Strichcodeleser usw.), die an der angegebenen COM-Schnittstelle angeschlossen sind. Wenn Sie diesen Befehl auf die COM-Schnittstellen der CPUs anwenden, wird außerdem der Empfangspuffer gelöscht (s. S. 720) und der Merker "Empfangen beendet" zurückgesetzt, so dass weiterer Datenempfang möglich ist.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

☞

- **Wenn die angegebene Anzahl von Bytes gesendet wurde, wird der Merker "Senden beendet" auf TRUE gesetzt. Weitere Daten können gesendet oder empfangen werden. Jeder Sendebefehl setzt den Merker "Senden beendet" auf FALSE; es ist dann kein Datenempfang mehr möglich. Die Auswertung des Merkers "Empfangen beendet" kann in Fällen sinnvoll sein, wo keine Antwort erwartet wird, z.B. bei Broadcast-Meldungen.**
- **SPSen mit nur einer Schnittstelle kompilieren F159_MTRN in F144_TRNS (die Variable d_Port wird ignoriert).**
- **F159_MTRN ist in den folgenden Befehlen gekapselt:**
  - **SendCharacters (s. S. 698)**
  - **SendCharactersAndClearString (s. S. 700)**
  - **ClearReceiveBuffer (s. S. 718)**
  - **SetCommunicationMode (s. S. 683)**

SPS Typen siehe (s. S. 1187)

Datentypen

Variable	Datentyp	Beschreibung
s_Start	ANY16	Sendepuffer
n_Number		▪ Anzahl der zu sendenden Bytes: – Negativer Wert: Das in den Systemregistern gewählte Endezeichen wird nicht an die zu übertragende Zeichenfolge angehängt. – 0 (Null Byte): System für weiteren Datenempfang vorbereiten (s. S. 720) – 16#8000: Umstellen der Kommunikationsart (s. S. 685)
d_Port		Kommunikationsschnittstellen Für **FP-X**, **FP$_\Sigma$** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20) **Andere SPSen:** SPSen mit nur einer Schnittstelle kompilieren F159_MTRN in F144_TRNS (die Variable **d_Port** wird ignoriert).

Operanden

Für	Merker				T/C		Register			Konstante
s_Start	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
n_Number	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez. oder hex.
d_Port	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	dauerhaft	▪ die mit Indexmodifizierern definierte Adresse den zulässigen Bereich überschreitet ▪ die durch '**n_Number**' angegebene Anzahl zu sendender Bytes liegt außerhalb des festgelegten Bereichs. **Merker für MCU-Modul:** ▪ das MCU-Modul sich nicht im angegebenen Steckplatz befindet ▪ 16#8000 wird im MEWTOCOL-COM Master/Slave-Modus festgelegt
R9008	%MX0.900.8	kurzzeitig	

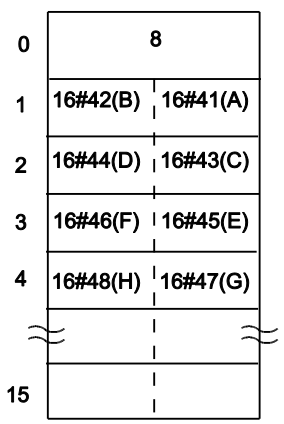
Beispiel

In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

In diesem Beispiel werden die Zeichen der Zeichenfolge **sSendData** übertragen.

Definieren Sie einen Sendepuffer für 30 Byte (ARRAY [0...15] OF WORD) und kopieren Sie 8 Zeichen einer Zeichenfolge ("ABCDEFGH") in den Puffer.

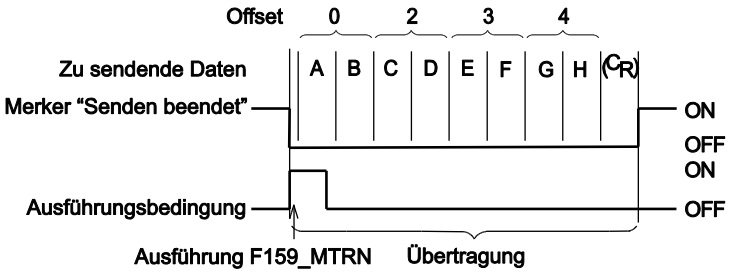
Aufbau des Sendepuffers:



Das erste Wort des Sendepuffers (Offset 0) ist für die Anzahl der zu sendenden Bytes reserviert. Kopieren Sie also die Daten in Offset 1 (SendBuffer[1]).

Sobald der Sendevorgang beginnt (d.h. die Ausführungsbedingung des Sendebefehls auf TRUE steht), wird der Wert von Offset 0 auf 8 gesetzt. Am Ende der Übertragung wird der Wert von Offset 0 automatisch wieder auf 0 zurückgesetzt.

Übertragen Sie die Zeichen "ABCDEFGH" an das externe Gerät, das mit der Schnittstelle COM1 verbunden ist. Start- und Endezeichen haben die Standardwerte "Kein STX" und "CR".



1	Merker "Senden beendet"	3	Ausführung des Sendebefehls
2	Ausführungsbedingung	4	Senden

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bSend	BOOL	FALSE	activates function
1	VAR	sSendData	STRING[30]	'ABCDEFGH'	up to 30 chars
2	VAR	awSendBuffer	ARRAY [0..15] OF WORD	[16(0)]	for 30 chars + 1 word

Rumpf

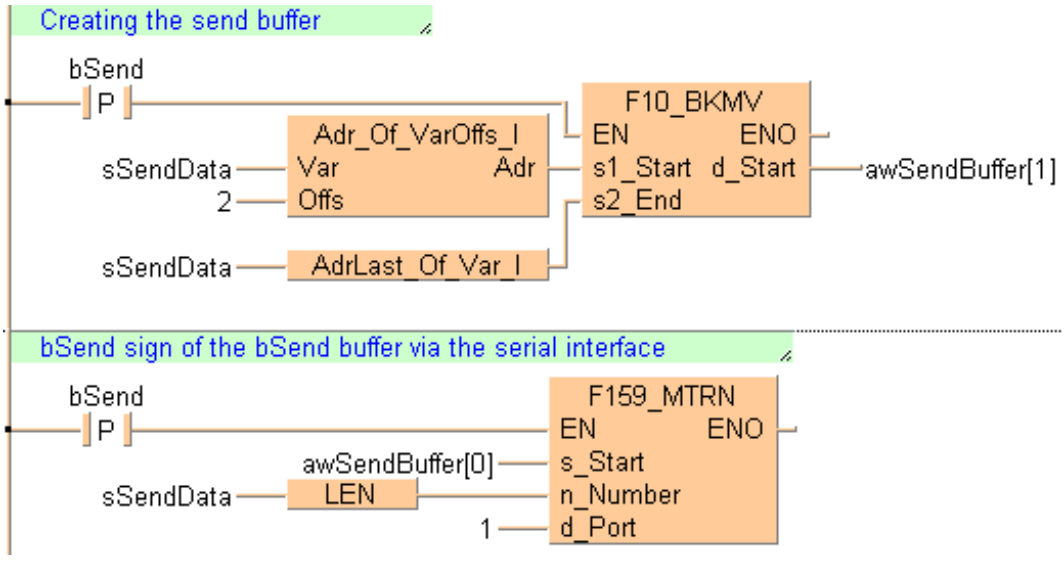
Wenn **bSend** TRUE ist, kopiert F10_BKMV die Zeichen der Zeichenfolge aus **sSendData** in den Puffer **awSendBuffer**, beginnend bei **awSendBuffer[1]**.

Die ersten beiden Worte einer Zeichenfolge enthalten die Kopfinformationen (maximale und aktuelle Zeichenanzahl). Die Kopfinformationen dürfen nicht in den Puffer kopiert werden. Fügen Sie deshalb der Startadresse der Zeichenfolge einen Offset von 2 hinzu, ehe Sie die Daten kopieren.

Der Sendepuffer muss ausreichend groß bemessen sein. Jedes Element des Arrays an **SendBuffer** kann zwei Zeichen der Zeichenfolge an **SendString** enthalten. **SendBuffer[0]** ist für die Anzahl der zu sendenden Bytes reserviert.

Der Befehl F159_MTRN sendet die Daten aus dem ersten Element des Sendepuffers (**awSendBuffer[0]**), das mit **s_Start** angegeben wird. Die Länge der zu sendenden Zeichenfolge (8 Byte) wird mit **n_Number** festgelegt. Verwenden Sie die Funktion LEN, um die Anzahl der Bytes zu berechnen. Die Daten werden an COM 1 ausgegeben, wie durch **d_Port** festgelegt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if (DF(bSend)) then
    (* Copy all characters of the SendString to the SendBuffer from position
1 *)
    F10_BKMV(s1_Start :=Adr_Of_VarOffs(Var :=sSendData, Offs := 2),
          s2_End :=AdrLast_Of_Var(sSendData),
          d_Start =>awSendBuffer[1]);
    (* Send the data of the SendBuffer via the COM Port 2 of the MCU unit in
slot 3 *)
    (* In SendBuffer[0] the number of bytes not yet transmitted is stored *)

    F159_MTRN(s_Start :=SendBuffer[0], n_Number :=LEN(sSendData), d_Port :=
16#0302);
end_if;
```

IsTransmissionDone

Merker "Senden beendet" auswerten

Erklärung

Dieser Befehl gibt den Wert des Merkers "Senden beendet" (s. S. 697) zurück. Der Merker "Senden beendet" wird auf TRUE gesetzt, wenn von der festgelegten COM-Schnittstelle der SPS die angegebene Anzahl von Bytes gesendet wurde. Weitere Daten können gesendet oder empfangen werden. Jeder Sendebefehl setzt den Merker "Senden beendet" auf FALSE; es ist dann kein Datenempfang mehr möglich. Die Auswertung des Merkers "Empfangen beendet" kann in Fällen sinnvoll sein, wo keine Antwort erwartet wird, z.B. bei Broadcast-Meldungen.

SPS Typen siehe (s. S. 1194)

Datentypen

Eingangsvariable	Datentyp	Beschreibung
Port	ANY16	Kommunikationsschnittstellen Für **FP-X**, **FP$_{\Sigma}$** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** — **MCU:** SYS_COM1_PORT — 16#xx01 (COM1) SYS_COM2_PORT — 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)
Ausgangsvariable		
IsDone	BOOL	TRUE, wenn das Endezeichen empfangen wurde. Das Endezeichen wird in den Systemregistern festgelegt.

Beispiel

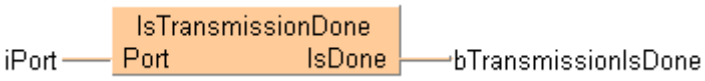
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iPort	INT	0
1	VAR	bTransmissionIsDone	BOOL	FALSE

KOP



ST

```
bTransmissionIsDone:=IsTransmissionDone(Port := iPort);
```

20.4.2 Daten empfangen

Die Daten können von einem externen Gerät empfangen werden, wenn der Merker "Empfangen beendet" auf FALSE steht.(Der Merker "Empfangen beendet" wird beim Umschalten in den RUN-Modus auf FALSE gesetzt.) Die Daten werden automatisch im Empfangspuffer (s. S. 709) der CPU oder des MCU-Moduls empfangen. Bei einer CPU muss der Empfangspuffer in den Systemregistern definiert werden. Wenn das Empfangsende ermittelt wurde, können die Daten in einen festgelegten Zielbereich der CPU kopiert werden.

Wenn ein Endezeichen empfangen wird, wird der Merker "Empfangen beendet" auf TRUE gesetzt. Weiterer Datenempfang ist unmöglich. Maximal 4094 Byte können empfangen werden. Die gespeicherten Daten enthalten keine Endezeichen.

Ablauf des Datenempfangs:

- **Schritt 1:** Kommunikationsparameter (s. S. 683) und Empfangspuffer (s. S. 709) einstellen

Erforderliche Einstellungen: Kommunikationsart (programmgesteuert), Baudrate, Kommunikationsformat, Empfangspuffer (nur CPU)

- **Schritt 2:**Daten empfangen

Eingehende Daten werden automatisch im Empfangspuffer empfangen.

- **Schritt 3:** Empfangsende feststellen

Verwenden Sie eine der folgenden Möglichkeiten:

Methode	Beschreibung
IsReceptionDone (s. S. 712)	Gibt den Wert des Merkers "Empfangen beendet" zurück. Wird auf TRUE gesetzt, wenn das Endezeichen empfangen wurde.
IsReceptionDoneByTimeOut (s. S. 713)	Bestimmt das Empfangsende durch eine Zeitsteuerung, z.B. bei Binärdaten ohne Endezeichen.
sys_bIsComPort1ReceptionDone sys_bIsComPort2ReceptionDone sys_bIsToolPortReceptionDone (nur CPU)	Diese Systemvariablen werden auf TRUE gesetzt, wenn das Endezeichen empfangen wurde.
Eingangs(X)-Merker X0 und X2 (nur MCU)	Zeigen das Ende einer Datenübertragung am MCU-Modul an.
Direkte Auswertung des Empfangspuffers.	

Schritt 4: Daten im Empfangspuffer verarbeiten

Verwenden Sie einen der folgenden Befehle:

Befehl	Beschreibung
ReceiveData (s. S. 714)	Kopiert die über eine CPU oder ein MCU-Modul empfangenen Daten automatisch in die festgelegte Variable.
ReceiveCharacters (s. S. 715)	Kopiert die über eine CPU oder ein MCU-Modul empfangenen Zeichen automatisch in die festgelegte Zeichenfolgenvariable.
F10_BKMV (s. S. 775)	Überträgt die Daten vom Empfangspuffer in einen Zielbereich. Nicht erforderlich bei ReceiveData oder ReceiveCharacters.
F161_MRCV (s. S. 717)	Kopiert über ein MCU-Modul erhaltene Daten in den Empfangspuffer einer CPU. Nicht erforderlich bei ReceiveData oder ReceiveCharacters.

- **Schritt 5:** CPU oder MCU-Modul für nächsten Datenempfang vorbereiten

Verwenden Sie einen der folgenden Befehle:

Befehl	Beschreibung
ClearReceiveBuffer (s. S. 718)	Der Empfangspuffer wird beim Senden der nächsten Daten automatisch zurückgesetzt. Um den Empfangspuffer ohne das Senden von Daten zurückzusetzen, verwenden Sie einen dieser Befehle.
F159_MTRN (n_Number=0) (s. S. 720)	

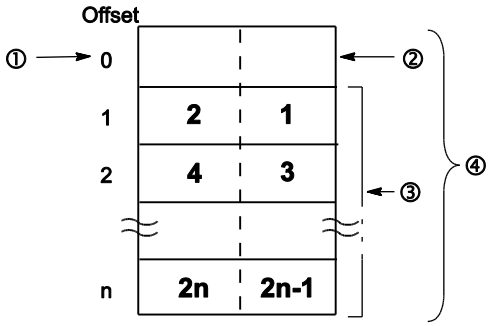
20.4.2.1 Empfangspuffer der CPU definieren

Für die programmgesteuerte Kommunikation muss im DT-Speicherbereich ein Empfangspuffer definiert werden. Er kann maximal 2048 Worte speichern.

Machen Sie folgende Einstellungen:

1. Anfangsadresse
2. Größe des Empfangspuffers (Anzahl der Worte)

Aufbau des Empfangspuffers



Die eingekreisten Zahlen geben die Schreibreihenfolge an.

1	Anfangsadresse	3	Speicherbereich für die empfangenen Daten
2	Speicherbereich für die Anzahl der empfangenen Bytes	4	Speichergröße

Eingehende Daten werden im Empfangspuffer gespeichert. Anfangs- und Endezeichen werden nicht im Empfangspuffer gespeichert. Der Speicherbereich für die empfangenen Daten beginnt mit dem zweiten Wort des Empfangspuffers (Offset 1). Offset 0 enthält die Anzahl der empfangenen Bytes. Der Anfangswert in Offset 0 ist 0.

◆Vorgehensweise

1. **Im Navigator auf "SPS" doppelklicken**
2. **Auf “Systemregister” doppelklicken**
3. **Auf "COM-Schnittstelle" doppelklicken**

   Da die Kommunikationsschnittstellen unterschiedliche Bitpositionen desselben Systemregisters belegen, sind individuelle Einstellungen für jede Schnittstelle möglich. Die Einstellungen für die TOOL-Schnittstelle werden in den Systemregistern unter "TOOL-Schnittstelle" vorgenommen.

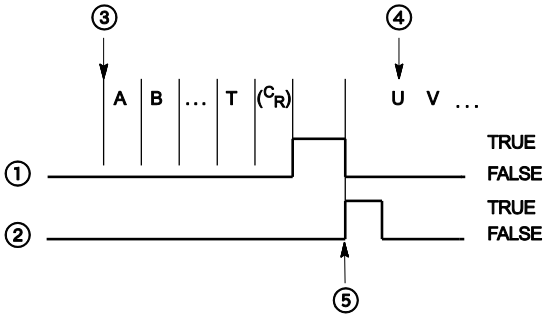
   Die Nummern der Systemregister sind nicht für alle SPS-Typen gleich.

◆HINWEIS

Um die Daten im Empfangspuffer zu verwenden, definieren Sie eine globale Variable mit derselben Anfangsadresse und Speichergröße.

Empfangene Daten verarbeiten und weiteren Datenempfang vorbereiten

Eine Zeichenfolge von 8 Bytes mit den Zeichen "ABCDEFGH" wird über COM1 empfangen. Diese Zeichen werden im Format ASCII HEX ohne Start- und Endezeichen gespeichert.



1	Merker "Empfangen beendet"	4	Empfang wird fortgesetzt
2	Ausführungsbedingung	5	Ausführung von F159_MTRN (n_Number=0) (s. S. 720)
3	Empfang beginnt		

Aufbau des Empfangspuffers:

Offset	8	
0		
1	16#42(B)	16#41(A)
2	16#44(D)	16#43(C)
3	16#46(F)	16#45(E)
4	16#48(H)	16#47(G)

Zu Beginn des Datenempfangs steht in Offset 0 der Wert 8. Am Ende der Übertragung wird der Wert in Offset 0 auf 0 gesetzt. Die Daten in Offset 1 bis Offset 4 werden nacheinander, beginnend mit dem niederwertigen Byte, empfangen.

Systemregistereinstellungen

Nr.	Bezeichnung	Daten
412	COM1-Schnittstelle: Kommunikationsart	Programmgesteuert [Andere Geräte]
410	COM1-Schnittstelle: Teilnehmeradresse	1
415	COM1-Schnittstelle: Baudrate	9600
413	COM1-Schnittstelle: Datenlänge	8 Bits
413	COM1-Schnittstelle: Art der Paritätsprüfung	Ungerade
413	COM1-Schnittstelle: Stoppbit	1 Bit
413	COM1-Schnittstelle: Startzeichen	Kein STX
413	COM1-Schnittstelle: Endezeichen/Bedingung für Merker "Em...	CR
416	COM1-Schnittstelle: Anfangsadresse Empfangspuffer	0
417	COM1-Schnittstelle: Größe Empfangspuffer	0

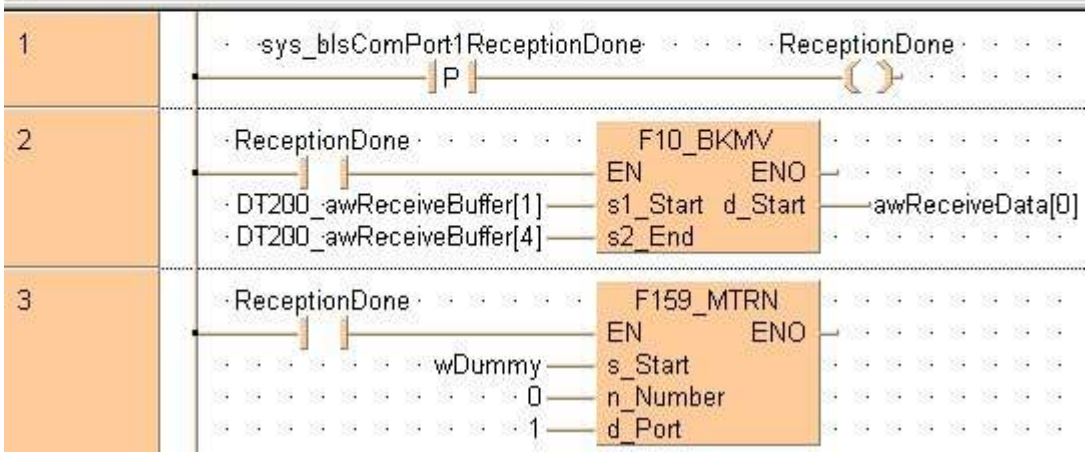
Um die Daten im Empfangspuffer zu verwenden, definieren Sie eine globale Variable mit derselben Anfangsadresse und Speichergröße. In diesem Beispiel ist die Anfangsadresse 200 (VAR_GLOBAL ReceivedData) und die Größe des Empfangspuffers beträgt 5 (ARRAY [0..4] OF WORD).

GVL

	Klasse	Bezeichner	FP-A...	IEC-Adre...	Typ	Initial
0	VAR_GLOBAL	DT200_awReceiveBuffer	DT200	%MW5.200	ARRAY [0..4] OF WORD	[5(0)]

POE und KOP

	Klasse	Bezeichner	Typ	Initial	Kor
0	VAR	wDummy	WORD	0	
1	VAR	ReceptionDone	BOOL	FALSE	
2	VAR_EXTERNAL	DT200_awReceiveBuffer	ARRAY [0..4] OF WORD	[5(0)]	
3	VAR	awReceiveData	ARRAY[0..3] OF WORD	[4(0)]	



ST

```
if (sys_bIsComPort1ReceptionDone) then
    F10_BKMV(s1_Start := DT200_awReceiveBuffer[1], s2_End := DT200_awReceiveBuffer[4],
    d_Start => awReceiveData[0]);
    F159_MTRN(s_Start := wDummy, n_Number := 0, d_Port := 1);
end_if;
```

Die Daten können von einem externen Gerät empfangen werden, wenn der Merker "Empfangen beendet" auf FALSE steht. Die Auswertung des Merkers "Empfangen beendet" erfolgt mit der Systemvariable sys_bIsComPort1ReceptionDone. Wenn der Empfang der Daten abgeschlossen ist (nach Empfang des Endezeichens) wird der Merker "Empfangen beendet" auf TRUE gesetzt, und es können zunächst keine Daten mehr empfangen werden. Soll das System in Empfangsbereitschaft versetzt werden, ohne erneut Daten zu senden, muss der Empfangspuffer mit dem Befehl F159_MTRN und **n_Number** = 0 zurückgesetzt werden.

◆HINWEIS

- **Der Status des Merkers "Empfangen beendet" kann sich während eines Zyklus verändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.**
- **Mit dem Startzeichen "STX" wird der Empfangspuffer zurückgesetzt. Beim Zurücksetzen des Empfangspuffers wird die Anzahl der empfangenen Bytes in Offset 0 auf Null und der Zeiger zurück auf Offset 1 gesetzt. Die nächsten Daten werden im Empfangspuffer beginnend bei Offset 1 gespeichert.**

IsReceptionDone

Liefert den Wert des Merkers "Empfangen beendet"

Erklärung

Dieser Befehl gibt den Wert des Merkers "reception done" zurück. Er ist TRUE, wenn das Endezeichen an der Kommunikationsschnittstelle empfangen wurde, die durch **Port** festgelegt wurde.

Auswertung des Merkers "Empfang beendet"

Wenn ein Endezeichen empfangen wird, wird der Merker "Empfangen beendet" auf TRUE gesetzt. Weiterer Datenempfang ist unmöglich. F159_MTRN (s. S. 703) setzt den Merker "Empfangen beendet" auf FALSE. Der Merker "Empfang beendet" kann mit den folgenden Befehlen oder Systemvariablen ausgewertet werden:

- IsReceptionDone
- IsReceptionDoneByTimeOut (s. S. 713)
- sys_bIsComPort1ReceptionDone, sys_bIsComPort2ReceptionDone, sys_bIsToolPortReceptionDone (je nach Port)

Das Empfangsende lässt sich durch die Auswertung des Empfangspufferinhalts ermitteln. Der Status des Merkers "Empfangen beendet" kann sich während eines Zyklus verändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

Datentypen

Eingangsvariable	Datentyp	Beschreibung
Port	ANY16	Kommunikationsschnittstellen Für **FP-X**, **FPΣ** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)
Ausgangsvariable		
IsDone	BOOL	TRUE, wenn das Endezeichen empfangen wurde. Das Endezeichen wird im betreffenden Systemregister unter den Einstellungen für die COM-Schnittstelle festgelegt.

Beispiel

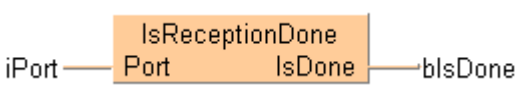
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iPort	INT	0
1	VAR	bIsDone	BOOL	FALSE

KOP



ST

```
bIsDone:=IsReceptionDone(Port := iPort);
```

IsReceptionDonebyTimeOut

Auswerten von "Empfang beendet" durch Zeitüberschreitung

Erklärung Je nach SPS-Typ und dem Eingangsparameter Port, wertet diese Funktion die Bedingung von "Empfang beendet" aus, wenn kein Endezeichen im Datenstrom erwartet wird, z.B. bei der Übertragung von Binärdaten.

Symbol:

Dieser Funktionsbaustein wertet die Time-out-Bedingung aus, um das Empfangsende eines Datenstroms zu ermitteln, der kein Endezeichen enthält, z.B. beim Übertragen von Binärdaten.

Bei einer CPU wird der Ausgang **IsDone** auf TRUE gesetzt, wenn die Anzahl der empfangenen Bytes sich nicht innerhalb der Zeit ändert, die unter **TimeOutForCPU** angegeben ist. Um dies zu prüfen, verbinden Sie das erste Wort des Empfangspuffers mit **NoOfBytesReceived** (Anzahl der empfangenen Bytes).

Bei einem MCU-Modul kann die Zeitüberschreitung im PROG-Modus im Dialogfeld Dialogfeld "MCU Einstellung" eingestellt werden, oder im RUN-Modus mit F159_MWRT_PARA.

Auswertung des Merkers "Empfang beendet"

Wenn ein Endezeichen empfangen wird, wird der Merker "Empfangen beendet" auf TRUE gesetzt. Weiterer Datenempfang ist unmöglich. F159_MTRN setzt den Merker "Empfangen beendet" auf FALSE. Der Merker "Empfang beendet" kann mit einem der nachstehenden Befehle oder Systemvariablen ausgewertet werden:

- IsReceptionDone
- IsReceptionDoneByTimeOut
- sys_bIsComPort1ReceptionDone, sys_bIsComPort2ReceptionDone, sys_bIsToolPortReceptionDone (je nach Port)

Das Empfangsende lässt sich auch durch die Prüfung der Inhalte des Empfangspuffers ermitteln. Der Status des Merkers "Empfangen beendet" kann sich während eines Zyklus verändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

Datentypen

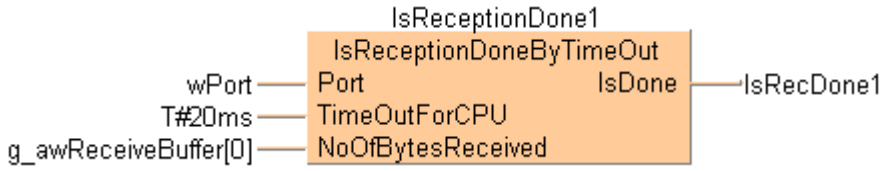
Eingangsvariable	Datentyp	Beschreibung
Port	ANY16	Kommunikationsschnittstellen Für FP-X, FP□ und FP2, FP2SH (V1.4 oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)
TimeOutForCPU	TIME	Legt die Zeitspanne fest. Werden in dieser Zeit keine weiteren Daten empfangen, ist der Empfang beendet und **IsDone** wird auf TRUE gesetzt.
NoOfBytesReceived	ANY16	Legen Sie hier die Anfangsadresse des Empfangspuffers an. Diese Adresse enthält die Anzahl der empfangenen Bytes. (Nur CPU)
Ausgangsvariable		
IsDone	BOOL	TRUE, wenn ein oder mehrere Bytes empfangen wurden, und sich die Anzahl der empfangenen Bytes innerhalb der mit **TimeOutForCPU** oder im "Dialogfeld MCU Einstellung" festgelegten Zeitspanne nicht geändert hat.

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	IsReceptionDone1	IsReceptionDoneByTimeOut	
1	VAR_EXTERNAL	g_awReceiveBuffer	ARRAY [0..10] OF WORD	
2	VAR	bIsRecDone1	BOOL	FALSE
3	VAR	iPort	WORD	0

KOP



ST

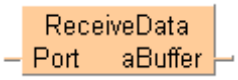
```
IsReceptionDone1(Port := wPort,
                 TimeOutForCPU := T#20ms,
                 NoOfBytesReceived := g_awReceiveBuffer[0],
                 IsDone => bIsRecDone1);
```

ReceiveData

Daten von CPU- oder MCU-Schnittstelle empfangen

Erklärung

Dieser Befehl kopiert die Daten, die an der durch **Port** festgelegten Schnittstelle empfangen wurden, in die **aBuffer** zugewiesene Variable.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1196

Datentypen

Eingangsvariable	Datentyp	Beschreibung
Port	ANY16	Kommunikationsschnittstellen Für **FP-X**, **FP$_\Sigma$** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)
Ausgangsvariable		
aBuffer	ANY	Speichert empfangene Daten

Fehlermerker

Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	dauerhaft	▪ das MCU-Modul ist nicht an den durch '**Port**' festgelegten Steckplatz angeschlossen.
R9008	%MX0.900.8	kurzzeitig	

Beispiel

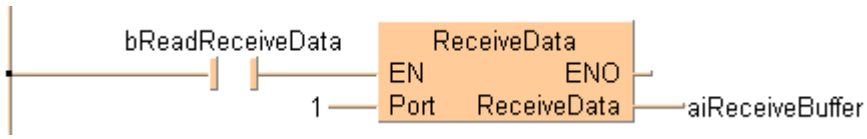
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bReceive	BOOL	FALSE	activates function
1	VAR	sString	String	''	

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

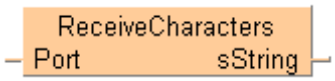
```
if (bReadReceiveData) then
    aiReceiveBuffer:=ReceiveData(1);
end_if;
```

ReceiveCharacters

Zeichen von CPU- oder MCU-Schnittstelle empfangen

Erklärung

Dieser Befehl kopiert die Zeichen, die an der durch **Port** festgelegten Schnittstelle empfangen wurden, in die **sString** zugewiesene Variable.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen

s. S. 1196

Datentypen

Eingangsvariable	Datentyp	Beschreibung
Port	ANY16	Kommunikationsschnittstellen Für **FP-X**, **FP$_{\Sigma}$** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)
Ausgangsvariable		
sString	STRING	Speichert empfangene Zeichen

Fehlermerker

Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	dauerhaft	▪ das MCU-Modul ist nicht an den durch '**Port**' festgelegten Steckplatz angeschlossen.
R9008	%MX0.900.8	kurzzeitig	

Beispiel

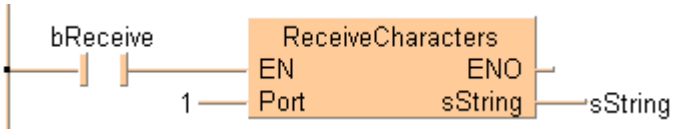
In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bReceive	BOOL	FALSE	activates function
1	VAR	sString	String	''	

KOP



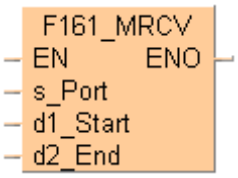
ST

```
If (bReceive) Then
    sString:=ReceiveCharacters(1);
End_if;
```

F161_MRCV

Serielle Daten von MCU-Schnittstelle lesen

ErklÃ¤rung Mit diesem Befehl kopieren Sie die Daten, die das MCU-Modul vom externen Gerät empfangen hat, in den Empfangspuffer der CPU. Die Schnittstelle des MCU-Moduls wird durch **s_Port** festgelegt. Der Empfangspuffer wird durch **d1_Start** und **d2_End** festgelegt.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Führen Sie F161_MRCV nicht aus, bevor der Empfang durch die Auswertung des Merkers "Empfangen beendet" bestätigt wird. Eine zyklische Abfrage mit dem Befehl F161_MRCV ist nicht möglich! Der Merker "Empfangen beendet" lässt sich mit den Funktionen IsReceptionDone (s. S. 712) und IsReceptionDoneByTimeOut (s. S. 713) auswerten oder anhand der Eingangsmerker X0 und X2.

Die Anzahl der empfangenen Bytes wird in der mit **d1_Start** festgelegten Anfangsadresse des Empfangspuffers gespeichert. Überschreiten die empfangenen Daten die mit **b2_End** festgelegte Endadresse, wird ein Operationsfehler ausgegeben. Die Daten, die bis zum Erreichen von **d2_End** empfangen wurden, werden gespeichert. F161_MRCV löscht auch den Empfangspuffer (s. S. 720) und setzt den Merker "Empfangen beendet" zurück, damit weitere Daten empfangen werden können.

☞ **F161_MRCV wird von allen SPS-Typen unterstützt: Wenn anstelle von Merkern passende Funktionen verwendet werden, lassen sich SPS-unabhängige Programme erstellen, die die Kommunikation zwischen den COM-Schnittstellen sowohl mit CPU- als auch mit MCU-Schnittstellen regeln. SPS-Typen ohne MCU-Schnittstellen übersetzen den Befehl F161_MRCV einfach nicht.**

SPS Typen siehe (s. S. 1187)

Datentypen

Variable	Datentyp	Beschreibung
s_Port	ANY16	Steckplatznummer (höherwertiges Byte) und Schnittstellennummer (niederwertiges Byte) des MCU-Moduls, an das die Daten übertragen werden. 16#xx01: COM1 an MCU-Modul in Steckpatz 16#xx 16#xx02: COM2 an MCU-Modul in Steckpatz 16#xx
d1_Start		Anfangsadresse des Empfangspuffers
d2_End		Endadresse des Empfangspuffers

Operanden

Für	Merker				T/C		Register			Konstante
s_Port	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez. oder hex.
d1_Start	-	WY	WR	WL	SV	EV	DT	LD	FL	-
d2_End	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	dauerhaft	▪ die mit Indexmodifizierern definierte Adresse den zulässigen Bereich überschreitet ▪ das MCU-Modul sich nicht im angegebenen Steckplatz befindet ▪ die angegebene Kommunikationsschnittstelle nicht vorhanden ist
R9008	%MX0.900.8	kurzzeitig	

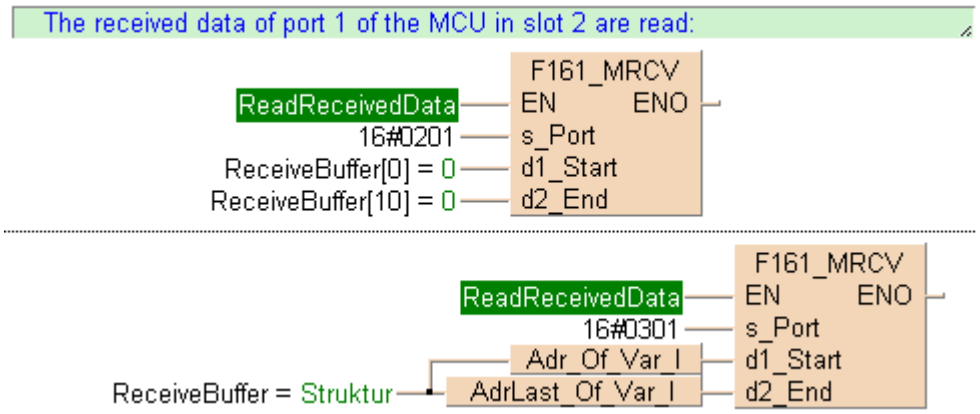
Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	ReadReceivedData	BOOL	FALSE	
1	VAR	ReceiveBuffer	ARRAY [0..10] OF INT	[11(0)]	

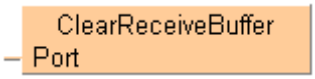
Rumpf

KOP



ClearReceiveBuffer Empfangspuffer zurÃ¼cksetzen

Erklärung Dieser Befehl setzt den Empfangspuffer an der durch **Port** festgelegten Kommunikationsschnittstelle zurück.



Bei der Ausführung eines Sendebefehls wird der Empfangspuffer automatisch zurückgesetzt. Wenn Sie den Empfangspuffer zurücksetzen möchten, ohne weitere Daten zu versenden, führen Sie diesen Befehl aus. Alternativ können Sie auch F159_MTRN (s. S. 703) mit **n_Number = 0** verwenden. Beim Zurücksetzen des Empfangspuffers stellt die Anzahl der empfangenen Bytes in Offset 0 auf Null und der Zeiger zurück auf Offset 1 gesetzt. Die nächsten Daten werden beim Start an Offset 1 gespeichert, vorhandene Daten werden überschrieben. Der Merker "Empfangen beendet" wird auf FALSE gesetzt.

SPS Typen s. S. 1184

Datentypen

Variable	Datentyp	Beschreibung
Port	ANY16	Kommunikationsschnittstellen Für **FP-X**, **FPΣ** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)

Operanden

Für	Merker				T/C		Register			Konstante
Port	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez. oder hex.

Fehlermerker

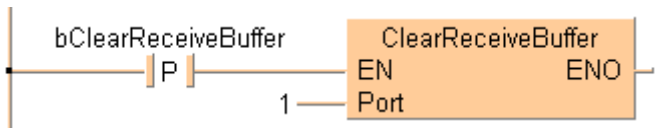
Nr.	IEC-Adresse	Setzen	Wenn
R900B	%MX0.900.11	kurzzeitig	▪ die durch **Port** angegebene Kommunikationsschnittstelle nicht vorhanden ist
R9009	%MX0.900.9	kurzzeitig	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bClearReceiveBuffer	BOOL	FALSE

KOP

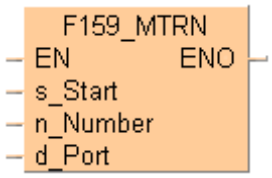


ST
```
if (DF(bClearReceiveBuffer)) then
    ClearReceiveBuffer(1);
end_if;
```

F159_MTRN

System fÃ¼r weiteren Datenempfang vorbereiten

Erklärung Bei der Ausführung eines Sendebefehls wird der Empfangspuffer automatisch zurückgesetzt. Wenn Sie den Empfangspuffer zurücksetzen möchten, ohne weitere Daten zu versenden, führen Sie diesen Befehl aus. Alternativ können Sie ClearReceiveBuffer (s. S. 718) verwenden. Beim Zurücksetzen des Empfangspuffers stellt die Anzahl der empfangenen Bytes in Offset 0 auf Null und der Zeiger zurück auf Offset 1 gesetzt. Die nächsten Daten werden beim Start an Offset 1 gespeichert, vorhandene Daten werden überschrieben.



SPS Typen siehe (s. S. 1187)

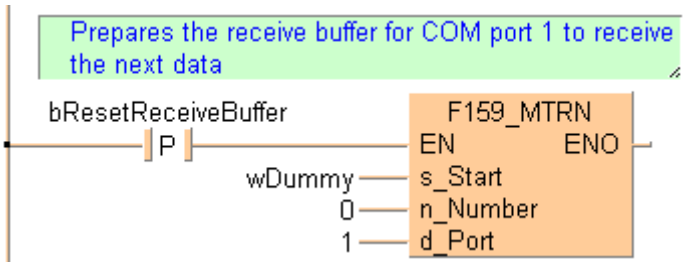
☞ **Wenn die Daten von einem MCU-Modul mit F161_MRCV (s. S. 717) empfangen werden, wird der Empfangsbereich gelöscht und der Merker "Empfangen beendet" zurückgesetzt. Es können weitere Daten über die COM-Schnittstelle empfangen werden. Um den Empfangspuffer ohne F161_MRCV zurückzusetzen, müssen Sie ClearReceiveBuffer (s. S. 718) verwenden. F159_MTRN zusammen mit n_Number=0 erzeugt einen Operationsfehler.**

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bResetReceiveBuffer	BOOL	FALSE
1	VAR	wDummy	WORD	0

KOP



ST

```
if (DF (ClearTheReceiveBuffer)) then
    (* Clears the receive buffer of the COM1 port of the FP-SIGMA *)
    F159_MTRN (s_Start := wDummy, n_Number := 0, d_Port := 1);
end_if;
```

20.4.3 Kommunikationsfehler

Wenn während der Kommunikation ein Fehler auftritt, wird der Merker "Kommunikationsfehler" auf TRUE gesetzt.

Wählen Sie eine der folgenden Möglichkeiten zum Auswerten des Merkers "Kommunikationsfehler":

Methode	CPU	MCU
IsCommunicationError (s. S. 722)	●	
sys_bIsComPort1CommunicationError sys_bIsComPort2CommunicationError sys_bIsToolPortCommunicationError	●	
Eingangsmerker X6 und X7		●

☞ **Wird der Merker "Kommunikationsfehler" während des Empfangs auf TRUE gesetzt, wird der Empfang fortgesetzt. Führen Sie einen Sendebefehl aus, um den Merker "Kommunikationsfehler" auf FALSE zu setzen und den Zeiger zurück auf Offset 1 zu stellen.**

IsCommunicationError

Liefert den Wert des Merkers "Kommunikationsfehler"

Erklärung Dieser Befehl gibt den Wert des Merkers "Kommunikationsfehler" zurück. Der Merker "Kommunikationsfehler" wird auf TRUE gesetzt, wenn an der durch **Port** festgelegten Schnittstelle ein Fehler in der seriellen Kommunikation aufgetreten ist.

SPS Typen siehe (s. S. 1193)

Datentypen

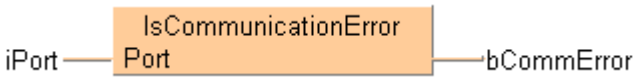
Variable	Datentyp	Funktion
Port	ANY16	Kommunikationsschnittstellen Für **FP-X**, **FP$_{\Sigma}$** und **FP2**, **FP2SH** (**V1.4** oder neuere Versionen): **SPS:** **MCU:** SYS_COM1_PORT 16#xx01 (COM1) SYS_COM2_PORT 16#xx02 (COM2) SYS_TOOL_PORT xx = Steckplatznummer (hexadezimal) der MCU (z.B. 16#0001: COM1 in Steckplatz 0, 16#0A02: COM2 in Steckplatz 10, 16#1401: COM1 in Steckplatz 20)

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iPort	INT	0
1	VAR	bCommError	BOOL	FALSE

KOP



ST
```
bCommError:=IsCommunicationError(Port := iPort);
```

20.5 Datenübertragung mit MEWTOCOL oder dem Modus Modbus-RTU-Master/Slave

20.5.1 Allgemeine Informationen zum Programmieren mit F145 und F146

Vorsichtsmaßnahmen beim Programmieren mit F145 und F146

- Es können nicht gleichzeitig mehrere F145_WRITE_DATA (s. S. 723) und F146_READ_DATA-Befehle für dieselbe COM-Schnittstelle ausgeführt werden. Achten Sie deshalb bei der Programmierung darauf, dass diese Befehle erst ausgeführt werden, wenn die Merker für Sende-/Empfangsbereitschaft im Netzwerk R9044 (COM1) bzw. R904A (COM2) gesetzt sind.

COM1	**sys_blsComPort1F145F146NotActive**	**R9044**	▪ 0: Befehl kann ausgeführt werden, da bereits ein WRITE– oder READ–Befehl ausgeführt wird. ▪ 1: Befehl kann ausgeführt werden.
COM2	**sys_blsComPort2F145F146NotActive**	**R904A**	

- Der Sendebefehl (z.B F145_WRITE_DATA) stellt lediglich eine Aufforderung zum Senden dar, die eigentliche Übertragung beginnt erst, wenn die ED-Anweisung ausgeführt wird. Verwenden Sie die Merker "Senden/Empfangen im Netzwerk abgebrochen" (R9045: COM1, R904B: COM2), um festzustellen, ob die Übertragung erfolgreich war.

COM1	**sys_blsComPort1F145F146Error**	**R9045**	▪ 0: Kein Fehler ▪ 1: Fehler (der Fehlercode wird in DT90045 gespeichert.)
COM1	**sys_wComPort1F145F146ErrorCode**	**DT90124**	Speichert den Fehlercode, wenn bei der Übertragung ein Fehler aufgetreten ist (R9045: 1).
COM2	**sys_blsComPort2F145F146Error**	**R904B**	▪ 0: Kein Fehler ▪ 1: Fehler (der Fehlercode wird in DT90125 gespeichert)
COM2	**sys_wComPort2F145F146ErrorCode**	**DT90125**	Speichert den Fehlercode, wenn bei der Übertragung ein Fehler aufgetreten ist (R904B: 1).

- Für detaillierte Informationen, siehe Fehlercodes. Fehlercode 16#73 bedeutet, dass die Kommunikation nach Überschreiten der Wartezeit abgebrochen wurde. Die Wartezeit kann 10,0ms bis 81,9s betragen (in Schritten von 10ms) und wird in Systemregister 32 eingestellt. Der Standardwert beträgt 10 Sekunden.

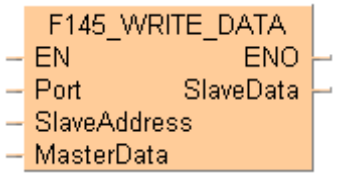
Fehlercode	Erklärung
16#73	Wartezeit überschritten: keine Antwort

- Wird als Slave-Adresse 16#00 angegeben, wird ein Broadcast-Befehl an alle Teilnehmer gesendet. Achten Sie bei der Programmierung darauf, dass der Befehl erst abgesetzt wird, nachdem die maximale Zykluszeit verstrichen ist.
- Zum Lesen oder Schreiben von internen Sondermerkern (R9000 ff.) und Sonderdatenregistern (DT90000 ff.) können die Befehle F145 und F146 nicht verwendet werden.

F145_WRITE_DATA Daten in einen Slave schreiben

Erklärung

Mit diesem Befehl schreiben Sie mit dem Protokoll MEWTOCOL oder Modbus-RTU Daten von einem Master über eine serielle Schnittstelle (COM1 oder COM2), wie in den Systemregistereinstellungen der verwendeten Schnittstelle definiert, in einen Slave (siehe Kommunikationsmodus). Sowohl Master als auch Slave müssen dasselbe Protokoll verwenden. Der Master muss im Modus Master/Slave konfiguriert sein. Der Slave kann entweder im Modus Master/Slave oder ausschließlich im Slave-Modus konfiguriert sein.



Die Daten werden aus dem Speicherbereich des Masters, der mit **MasterWordAddress** festgelegt wird, in den Speicherbereich des Slaves geschrieben, der mit **SlaveWordAddress** festgelegt wird. Die Variable **SlaveAddress** gibt die Teilnehmeradresse und die COM-Schnittstelle (1 oder 2) des Slaves an.

Beachten Sie auch die Informationen im Abschnitt Allgemeine Informationen zum Programmieren mit F145 und F146 (s. S. 723).

SPS-Typen **Verfügbarkeit von F145_WRITE_DATA (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
Port	ANY16	Gibt den COM-Port (1 oder 2) des Slaves über die Systemvariable an: SYS_COM1_PORT SYS_COM2_PORT
SlaveAddress	ANY16	Adresse der Station (1-99).
MasterData	ANY	Die Daten des Masters, die in den Slave geschrieben werden.
SlaveData	ANY	Die Daten des Slaves, in den die Daten geschrieben werden.

Operanden

Für	Merker				T/C		Register			Konstante
Port	WX	WY	WR	WL	-	-	DT	LD	FL	-
Slave Address	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
Master Data	WX	WY	WR	WL	-	-	DT	LD	FL	-
Slave Data	WX	WY	WR	WL	-	-	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Port 0 (Übertragung an alle Teilnehmer) keine Antwort von COM1 oder COM2 erhält ▪ Slave-Daten oder Master-Daten den verfügbaren Adressbereich überschreiten ▪ Als Kommunikationsart (s. S. 681) **nicht** MEWTOCOL-COM Master/Slave oder Modbus RTU Master/Slave eingestellt ist ▪ Der ausgewählte COM-Port ein Schnittstellenmodul erfordert, das nicht installiert ist
R9008	%MX0.900.8	permanent	

Beispiel In diesem Beispiel wird die Funktion F145 im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

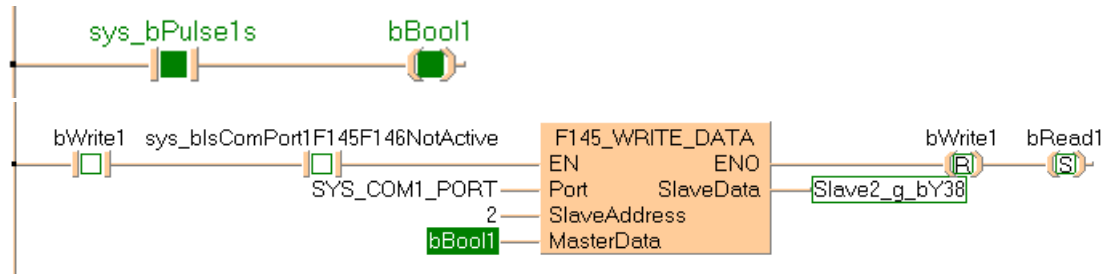
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial	Kommentar
0	VAR_GLOBAL	Slave2_g_bY38	Y38	%QX3.8	BOOL	FALSE	

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bBool1	BOOL	FALSE
1	VAR	bRead1	BOOL	FALSE
2	VAR	bWrite1	BOOL	FALSE
3	VAR_EXTERNAL	Slave2_g_bY38	BOOL	FALSE

Rumpf Die Systemvariable **sys_bPulse1s** wird in **bBool1** kopiert. Wenn **bWrite1** und **sys_bIsComPort1F145F146NotActive** auf TRUE gesetzt sind, wird **bBool1** in den Ausgang Y38 von Slave 2 geschrieben.

LD



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

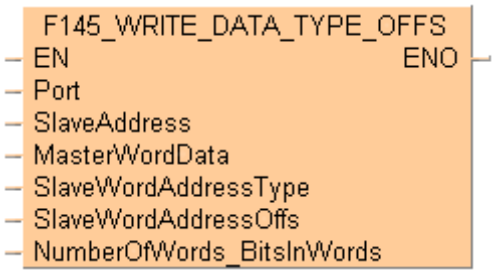
```
bBool1 := sys_bPulse1s;

if (bWrite1 and sys_bIsComPort1F145F146NotActive) then
      F145_WRITE_DATA(Port := SYS_COM1_PORT,
                      SlaveAddress := 2,
                      MasterData := bBool1,
                      SlaveData => Slave2_g_bR15);

      bRead1 := true;
      bWrite1 := false;
end_if;
```

F145_WRITE_DATA_TYPE_OFFS

Daten mit Type und Offset in Slave schreiben

Erklärung

Mit diesem Befehl schreiben Sie mit dem Protokoll MEWTOCOL oder Modbus-RTU Daten von einem Master über eine serielle Schnittstelle (COM1 oder COM2), wie in den Systemregistereinstellungen der verwendeten Schnittstelle definiert, in einen Slave (siehe Kommunikationsmodus). Sowohl Master als auch Slave müssen dasselbe Protokoll verwenden. Der Master muss im Modus Master/Slave konfiguriert sein. Der Slave kann entweder im Modus Master/Slave oder ausschließlich im Slave-Modus konfiguriert sein.



Die Daten werden aus dem Speicherbereich des Masters, der mit **MasterWordData** und NumberOfWords_ BitsInWords festgelegt wird, in den Speicherbereich des Slaves geschrieben, der mit **SlaveWordAddressType** und **SlaveAddressOffs** festgelegt wird.

Beachten Sie auch die Informationen im Abschnitt Allgemeine Informationen zum Programmieren mit F145 und F146 (s. S. 723).

SPS-Typen **Verfügbarkeit von F145_WRITE_DATA_TYPE_OFFS (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
Port	ANY16	Gibt den COM-Port (1 oder 2) des Slaves über die Systemvariable an: SYS_COM1_PORT SYS_COM2_PORT
SlaveAddress		Adresse der Station (1-99).
MasterWordData	ANY	Die Daten des Masters, die in den Slave geschrieben werden.
SlaveWordAddressType	ANY16	Adresstyp des Speicherbereichs im Slave, in den die Daten geschrieben werden. Der Offset muss Null, z.B. DT0, WL0...
SlaveWordAddressOffs		Offset für die Anfangsadresse des Speicherbereichs im Slave, in den die Daten geschrieben werden und dessen Typ mit **SlaveWordAddressType** festgelegt wird.

NumberOfWords_ BitsInWords		Anzahl der Wort-Einheiten, die an den Master gesendet werden (wenn das höchste Bit nicht gesetzt ist), oder Anzahl der Bits in den Worten (wenn das höchste Bit gesetzt ist). Ist identisch mit dem niederwertigen Wort des Steuerworts.

s1_ControlData

	Higher word				Lower word		
Hex	1	0 fixed	0	7	8	1	0 fixed
	COM port (16#1 or 16#2)		Unit No. (16#00 to 16#63) (0 to 99)		Bit unit transmission	Bit No. of Slave (16#0 to 16#F)	

- To generate function code 05, bit unit transmission (16#8) must be specified.

s1_ControlData: 16#10078100
s2_MasterStartAddr: 16#0000
d_SlaveStartAddrType: WY0
d_SlaveStartAddrOffs: 1

Command conversion

	Modbus command
1	Slave address
2	Function code (16#05)
3	Coil No. (H)
4	Coil No. (L)
5	Setting status (H)
6	Setting status (L)
7	CRC16 (H)
8	CRC16 (L)

- After the ON or OFF value of bit 0 of s2_MasterStartAddr has been read in the master, this
  slave (ON=FF00, OFF=0000).

Operanden

Für	Merker				T/C		Register			Konstante
Port	WX	WY	WR	WL	-	-	DT	LD	FL	-
Slave Address	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
Master WordData	WX	WY	WR	WL	-	-	DT	LD	FL	-
SlaveWord Address Type	WX	WY	WR	WL	-	-	DT	LD	FL	-
SlaveWord AddressOffs	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
NumberOfWords_ BitsInWords	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Port 0 (Übertragung an alle Teilnehmer) keine Antwort von COM1 oder COM2 erhält ▪ Slave-Daten oder Master-Daten den verfügbaren Adressbereich überschreiten ▪ SlaveWordAddressType: Offset ≠ 0 ▪ Als Kommunikationsart (s. S. 681) **nicht** MEWTOCOL-COM Master/Slave oder Modbus RTU Master/Slave eingestellt ist ▪ Der ausgewählte COM-Port ein Schnittstellenmodul erfordert, das nicht installiert ist
R9008	%MX0.900.8	permanent	

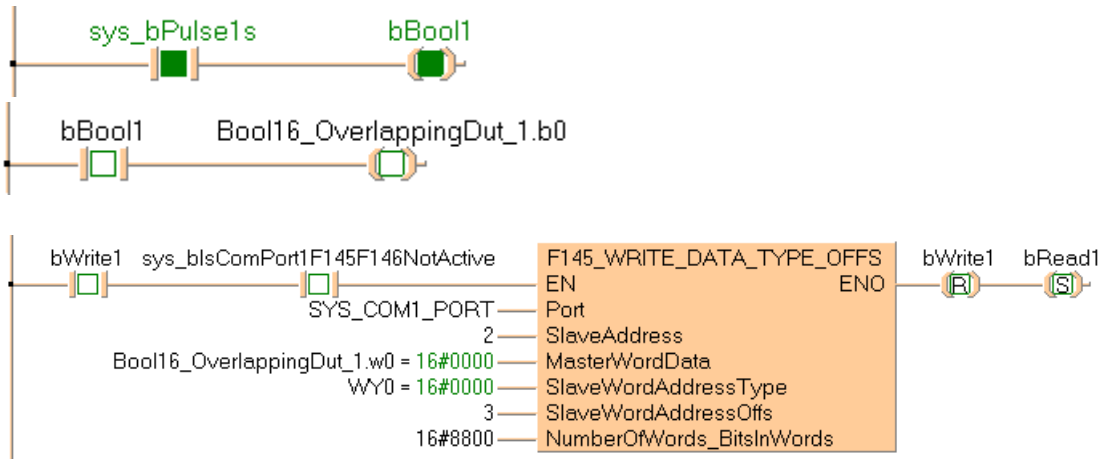
Beispiel In diesem Beispiel wird die Funktion F145 im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bBool1	BOOL	FALSE
1	VAR	bRead1	BOOL	FALSE
2	VAR	bWrite1	BOOL	FALSE
3	VAR	Bool16_OverlappingDut_1	BOOL16_OVERLAPPING_DUT	
4	VAR_EXTERNAL	Printer	WORD	0

Rumpf Die Systemvariable **sys_bPulse1s** wird in **bBool1** und **Bool16_OverlappingDut_1.b0**kopiert. Wenn **bWrite1** und **sys_bIsComPort1F145F146NotActive** auf TRUE gesetzt sind, wird **bBool1** über **Bool16_OverlappingDut_1.b0** in den Ausgang Y38 von Slave 2 geschrieben.

LD



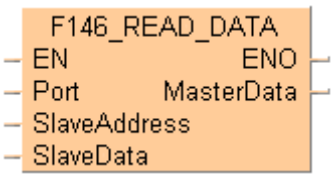
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bBool1 := sys_bPulse1s;
Bool16_OverlappingDut_1.b0 := bBool1;

if (bWrite1 and sys_bIsComPort1F145F146NotActive) then
    F145_WRITE_DATA_TYPE_OFFS(Port := SYS_COM1_PORT,
                              SlaveAddress := 2,
                              MasterWordData :=
Bool16_OverlappingDut_1.w0,
                              SlaveWordAddressType := WY0,
                              SlaveWordAddressOffs := 3,
                              NumberOfWords_BitsInWords := 16#8800);
    bRead1 := true;
    bWrite1 := false;
end_if;
```

F146_READ_DATA Daten von Slave lesen

Erklärung

Mit diesem Befehl fordern Sie mit dem Protokoll MEWTOCOL oder Modbus-RTU Daten von einem Slave (siehe Kommunikationsmodus) über eine serielle Schnittstelle (COM1 oder COM2) an, wie in den Systemregistereinstellungen der verwendeten Schnittstelle definiert. Sowohl Master als auch Slave müssen dasselbe Protokoll verwenden. Der Master muss im Modus Master/Slave konfiguriert sein. Der Slave kann entweder im Modus Master/Slave oder ausschließlich im Slave-Modus konfiguriert sein.



Die Daten werden aus dem Speicherbereich des Masters, der mit **MasterWordAddress** festgelegt wird, vom Speicherbereich des Slaves angefordert, der mit **SlaveWordAddress** festgelegt wird. Die Variable **SlaveAddress** gibt die Teilnehmeradresse und die COM-Schnittstelle (1 oder 2) des Slaves an.

Beachten Sie auch die Informationen im Abschnitt Allgemeine Informationen zum Programmieren mit F145 und F146 (s. S. 723).

SPS-Typen **Verfügbarkeit von F146_READ_DATA (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
Port	ANY16	Gibt den COM-Port (1 oder 2) des Slaves über die Systemvariable an: SYS_COM1_PORT SYS_COM2_PORT
SlaveAddress		Adresse der Station (1-99).
SlaveData	ANY	Die Daten des Slaves, in den die Daten geschrieben werden.
MasterData	ANY	Die Daten des Masters, in den die Daten (gelesen vom Slave) geschrieben werden.

Operanden

Für	Merker				T/C		Register			Konstante
Port	WX	WY	WR	WL	-	-	DT	LD	FL	-
Slave Address	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
Master Data	WX	WY	WR	WL	-	-	DT	LD	FL	-
Slave Data	WX	WY	WR	WL	-	-	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Port 0 (Übertragung an alle Teilnehmer) keine Antwort von COM1 oder COM2 erhält ▪ Slave-Daten oder Master-Daten den verfügbaren Adressbereich überschreiten ▪ Als Kommunikationsart (s. S. 681) **nicht** MEWTOCOL-COM Master/Slave oder Modbus RTU Master/Slave eingestellt ist ▪ Der ausgewählte COM-Port ein Schnittstellenmodul erfordert, das nicht installiert ist
R9008	%MX0.900.8	permanent	

Beispiel In diesem Beispiel wird die Funktion F146 im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

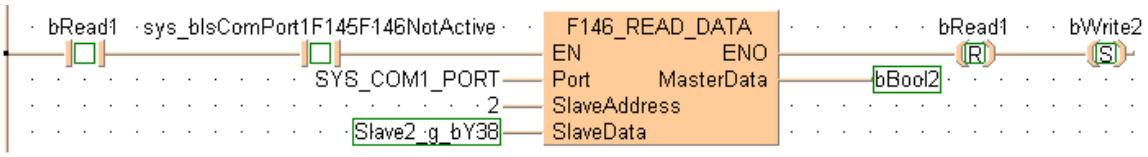
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial	Kommentar
0	VAR_GLOBAL	Slave2_g_bY38	Y38	%QX3.8	BOOL	FALSE	

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bBool2	BOOL	FALSE
1	VAR	bRead1	BOOL	FALSE
2	VAR	bWrite2	BOOL	FALSE
3	VAR_EXTERNAL	Slave2_g_bY38	BOOL	FALSE

Rumpf Wenn **bRead1** und **sys_bIsComPort1F145F146NotActive** auf TRUE gesetzt sind, wird die globale Variable **Slave2_g_bY38**, die Y38 von Slave 2 zugewiesen ist, in **bBool2** gelesen und gespeichert.

LD



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

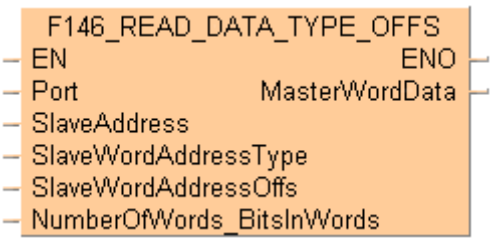
```
if (bRead1 and sys_bIsComPort1F145F146NotActive) then
        F146_READ_DATA(Port := SYS_COM1_PORT,
                       SlaveAddress := 2,
                       SlaveData := Slave2_g_bY38,
                       MasterData => bBool2);
    bRead1 := false;
    bWrite1 := true;
end_if;
```

F146_READ_DATA_TYPE_OFFS

Daten mit Type und Offset von Slave lesen

Erklärung

Mit diesem Befehl fordern Sie mit dem Protokoll MEWTOCOL oder Modbus-RTU Daten von einem Slave (siehe Kommunikationsmodus) über eine serielle Schnittstelle (COM1 oder COM2) an, wie in den Systemregistereinstellungen der verwendeten Schnittstelle definiert. Sowohl Master als auch Slave müssen dasselbe Protokoll verwenden. Der Master muss im Modus Master/Slave konfiguriert sein. Der Slave kann entweder im Modus Master/Slave oder ausschließlich im Slave-Modus konfiguriert sein.



Die Daten werden aus dem Speicherbereich des Slaves gelesen, der mit **SlaveAddressType** und **SlaveAddressOffs** festgelegt wird. Gespeichert werden die Daten im Speicherbereich des Masters, der mit **MasterWordAddress** festgelegt wird.

Beachten Sie auch die Informationen im Abschnitt Allgemeine Informationen zum Programmieren mit F145 und F146 (s. S. 723).

SPS-Typen **Verfügbarkeit von F146_READ_DATA_TYPE_OFFS (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
Port	ANY16	Gibt den COM-Port (1 oder 2) des Slaves über die Systemvariable an: SYS_COM1_PORT SYS_COM2_PORT
SlaveAddress		Adresse der Station (1-99).
SlaveWordAddressType		Adresstyp des Speicherbereichs im Slave, von dem die Daten gelesen werden.
SlaveWordAddressOffs		Offset für die Anfangsadresse des Speicherbereichs im Slave, in den die Daten geschrieben werden und dessen Typ mit **SlaveWordAddressType** festgelegt wird.
NumberOfWords_ BitsInWords		Anzahl der Wort-Einheiten, die vom Master gelesen werden (wenn das höchste Bit nicht gesetzt ist), oder Anzahl der Bits in den Worten (wenn das höchste Bit gesetzt ist). Ist identisch mit dem niederwertigen Wort des Steuerworts.
MasterWordData	ANY	Die Daten des Masters, die in den Slave geschrieben werden.

s1_ControlData

	Higher word				Lower word			
Hex	1	0 fixed	1	1	8	5	0	7

COM port (16#1 or 16#2)	Unit No. (16#00 to 16#63) (0 to 99)	Bit unit transmission	Bit No. of Master (16#0 to 16#F)	Bit No. of Slave (16#0 to 16#F)

- To generate function code 01, bit unit transmission (16#8) must be specified.

s1_ControlData: 16#10118507
s2_SlaveStartAddrType: WY0
s2_SlaveStartAddrOffs: 1
d_MasterStartAddr: 16#0000

Command conversion

Modbus command

1	Slave address	11
2	Function code (16#01)	01
3	Starting No. (H)	00
4	Starting No. (L)	17
5	No. of coils to read (H)	00
6	No. of coils to read (L)	01
7	CRC16 (H)	DC
8	CRC16 (L)	59

- Starting No. is the first coil to read in the slave (here: Y17)
- The No. of coils to read must be 1

Operanden

Für	Merker				T/C		Register			Konstante
Port	WX	WY	WR	WL			DT	LD	FL	-
Slave Address	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
SlaveWord Address Type	WX	WY	WR	WL	-	-	DT	LD	FL	-
SlaveWord AddressOffs	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
NumberOfWords_ BitsInWords	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
Master WordData	WX	WY	WR	WL	-	-	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Port 0 (Übertragung an alle Teilnehmer) keine Antwort von COM1 oder COM2 erhält ▪ Slave-Daten oder Master-Daten den verfügbaren Adressbereich überschreiten ▪ SlaveWordAddressType: Offset ≠ 0 ▪ Als Kommunikationsart (s. S. 681) **nicht** MEWTOCOL-COM Master/Slave oder Modbus RTU Master/Slave eingestellt ist ▪ Der ausgewählte COM-Port ein Schnittstellenmodul erfordert, das nicht installiert ist
R9008	%MX0.900.8	permanent	

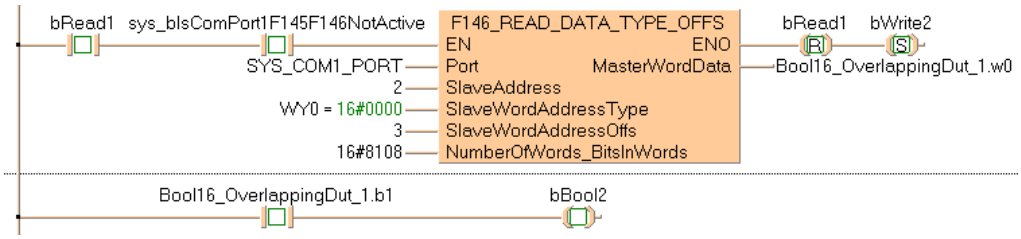
Beispiel In diesem Beispiel wird die Funktion F146 im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bBool2	BOOL	FALSE
1	VAR	bRead1	BOOL	FALSE
2	VAR	bWrite2	BOOL	FALSE
3	VAR	Bool16_OverlappingDut_1	BOOL16_OVERLAPPING_DUT	

Rumpf Wenn **bRead1** und **sys_bIsComPort1F145F146NotActive** auf TRUE gesetzt sind, wird der Ausgang Y38 von Slave 2 gelesen und in Bit 1 von **Bool16_OverlappingDut_1.w0.** geschrieben. Auf dieses Bit kann von **Bool16_OverlappingDut_1.b1** zugegriffen werden. Es wird in **bBool2** kopiert.

LD



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if (bRead1 and sys_bIsComPort1F145F146NotActive) then
    F146_READ_DATA_TYPE_OFFS(Port := SYS_COM1_PORT,
                             SlaveAddress := 2,
                             SlaveWordAddressType := WY0,
                             SlaveWordAddressOffs := 3,
                             NumberOfWords_BitsInWords := 16#8108,
                             MasterWordData =>
Bool16_OverlappingDut_1.w0);
end_if;

bBool2 := Bool16_OverlappingDut_1.b1;
```

F145F146_MODBUS_COMMAND

Schreibt Daten in Slave

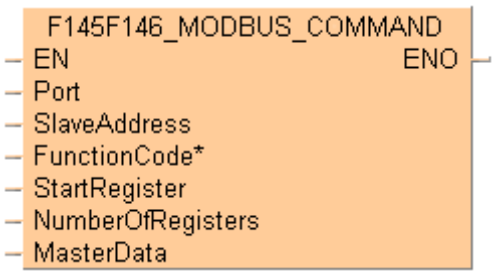
Erläuterung

Verfügbare Funktionscodes (Programmierbeispiele, siehe Online-Hilfe)

SYS_MODBUS_01_READ_COIL
SYS_MODBUS_02_READ_INPUT
SYS_MODBUS_03_READ_HOLDING_REGISTERS
SYS_MODBUS_04_READ_INPUT_REGISTERS
SYS_MODBUS_05_FORCE_COIL
SYS_MODBUS_06_PRESET_REGISTER
SYS_MODBUS_15_FORCE_COILS
SYS_MODBUS_16_PRESET_REGISTERS

Schreiben der Daten von einem Master auf einen Slave oder Lesen der Daten von einem Slave über eine serielle Schnittstelle (COM1 oder COM2), abhängig vom Funktionscode. Das Modbus-RTU-Protokoll (siehe Kommunikationsart) muss in den Systemregistern eingestellt sein. Wählen Sie für die gewünschte Schnittstelle "Modbus-RTU-Master/Slave".

Für Slave-Adressen, die über 99 liegen, oder für Anfangsregisteradressen, die außerhalb des zulässigen Bereichs liegen, sollten Sie den Befehl F145F146_MODBUS_MASTER (s. S. 736) verwenden.



Im Gegensatz zu den anderen F145- oder F146-Funktionen, kann der erforderliche Modbus-Befehl direkt durch den Parameter **FunctionCode*** angegeben werden.

Beachten Sie auch die Informationen im Abschnitt Allgemeine Informationen zum Programmieren mit F145 und F146 (s. S. 723).

Vom Master unterstützte Befehle:

Funktionscode	Systemkonstante	Anfangsadresse	Anzahl der Register	Referenznummern
01	SYS_MODBUS_01_READ_COIL	0–9998	1 oder ein Vielfaches von 16	000001–009999
02	SYS_MODBUS_02_READ_INPUT	0–9998	1	100001–109999
03	SYS_MODBUS_03_READ_HOLDING_REGISTERS	0–32764	≥1	400001–432765
04	SYS_MODBUS_04_READ_INPUT_REGISTERS	0–127 2000–2255	≥1	300001–300128 302001–302256
05	SYS_MODBUS_05_FORCE_COIL	0–9998	1	000001–009999
06	SYS_MODBUS_06_PRESET_REGISTER	0–32764	1	400001–432765
15	SYS_MODBUS_15_FORCE_COILS	0–9998	Vielfaches von 16	000001–009999
16	SYS_MODBUS_16_PRESET_REGISTERS	0–32764	≥1	400001–432765

Modbus-Spezifikationen für Panasonic-Steuerungen:

Referenznummern	Adressbereich der Panasonic-Steuerungen
Von 000001	Von Y0
Von 002049	Von R0
Von 100001	Von X0
Von 400001	Von DT0
Von 300001	Von WL0
Von 302001	Von LD0

Weitere Informationen zu den Referenznummern und eingeschränkten Adressbereichen der Panasonic-Steuerungen finden Sie im Benutzerhandbuch der SPS. Wenn die Referenznummer außerhalb des unterstützten Bereichs liegt, wird ein Fehler zurückgegeben.

SPS-Typen **Verfügbarkeit von F145F146_MODBUS_COMMAND (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
Port	ANY16	Gibt den COM-Port (1 oder 2) des Slaves über die Systemvariable an: SYS_COM1_PORT SYS_COM2_PORT
SlaveAddress		Adresse der Station (1-99).
FunctionCode*		SYS_MODBUS_01_READ_COIL SYS_MODBUS_02_READ_INPUT SYS_MODBUS_03_READ_HOLDING_REGISTERS SYS_MODBUS_04_READ_INPUT_REGISTERS SYS_MODBUS_05_FORCE_COIL SYS_MODBUS_06_PRESET_REGISTER SYS_MODBUS_15_FORCE_COILS SYS_MODBUS_16_PRESET_REGISTERS
StartRegister		Anfangsadresse Der Adresstyp hängt vom Befehl ab, der durch **FunctionCode*** angegeben wird.
NumberOfRegisters*		Anzahl der Übertragungs-Bits oder -Worte.
MasterData	ANY	Die Daten des Masters, die in den Slave geschrieben werden.

Operanden

Für	Relais				T/C		Register			Konstante
Port	WX	WY	WR	WL	-	-	DT	LD	FL	-
Slave Address	WX	WY	WR	WL	-	-	DT	LD	FL	-
Function Code*	-	-	-	-	-	-	-	-	-	vom System definiert
Start Register	WX	WY	WR	WL	-	-	DT	LD	FL	-
NumberOf Registers*	-	-	-	-	-	-	-	-	-	dez. oder hex.
Master Data	WX	WY	WR	WL	-	-	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Port 0 (Übertragung an alle Teilnehmer) keine Antwort von COM1 oder COM2 erhält ▪ Slave-Daten oder Master-Daten den verfügbaren Adressbereich überschreiten ▪ Als Kommunikationsart (s. S. 681) **nicht** MEWTOCOL-COM Master/Slave oder Modbus RTU Master/Slave eingestellt ist ▪ Der ausgewählte COM-Port ein Schnittstellenmodul erfordert, das nicht installiert ist
R9008	%MX0.900.8	permanent	

F145F146_MODBUS_MASTER

Daten in Slave schreiben oder Daten von Slave lesen

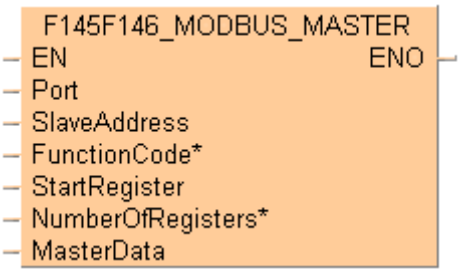
Erklärung

Verfügbare Funktionscodes (Programmierbeispiele, siehe Online-Hilfe)

SYS_MODBUS_01_READ_COIL
SYS_MODBUS_02_READ_INPUT
SYS_MODBUS_03_READ_HOLDING_REGISTER
SYS_MODBUS_04_READ_INPUT_REGISTERS
SYS_MODBUS_05_FORCE_COIL
SYS_MODBUS_06_PRESET_REGISTER
SYS_MODBUS_15_FORCE_COILS
SYS_MODBUS_16_PRESET_REGISTERS

Schreiben der Daten von einem Master auf einen Slave oder Lesen der Daten von einem Slave über eine serielle Schnittstelle (COM1 oder COM2), abhängig vom Funktionscode. Das Modbus-RTU-Protokoll (siehe Kommunikationsart) muss in den Systemregistern eingestellt sein. Wählen Sie für die gewünschte Schnittstelle "Modbus-RTU-Master/Slave".

Dieser Befehl ist identisch mit dem Befehl F145F146_MODBUS_COMMAND (s. S. 734); er unterstützt jedoch auch Slave-Adressen über 99 und einen größeren Anfangsadressbereich.



Im Gegensatz zu den anderen F145- oder F146-Funktionen, kann der erforderliche Modbus-Befehl direkt durch den Parameter **FunctionCode*** angegeben werden.

Beachten Sie auch die Informationen im Abschnitt Allgemeine Informationen zum Programmieren mit F145 und F146 (s. S. 723).

Vom Master unterstützte Befehle:

Funktionscode	Systemkonstante	Anfangsadresse	Anzahl der Register	Referenznummern (abhängig vom Modbus-Slave)
01	SYS_MODBUS_01_READ_COIL	0–65535	1–2040	000001–065536
02	SYS_MODBUS_02_READ_INPUT	0–65535	1–2040	100001–165536
03	SYS_MODBUS_03_READ_HOLDING_REGISTER	0–65535	1–127	400001–465536
04	SYS_MODBUS_04_READ_INPUT_REGISTERS	0–65535	1–127	300001–365536
5	SYS_MODBUS_05_FORCE_COIL	0–65535	1	000001–065536
6	SYS_MODBUS_06_PRESET_REGISTER	0–65535	1	400001–465536
15	SYS_MODBUS_15_FORCE_COILS	0–65535	2–2040	000001–065536
16	SYS_MODBUS_16_PRESET_REGISTERS	0–65535	2–127	400001–465536

Modbus-Spezifikationen für Panasonic-Steuerungen:

Referenznummern	Adressbereich der Panasonic-Steuerungen
Von 000001	Von Y0
Von 002049	Von R0
Von 100001	Von X0
Von 400001	Von DT0
Von 300001	Von WL0
Von 302001	Von LD0

Weitere Informationen zu den Referenznummern und eingeschränkten Adressbereichen der Panasonic-Steuerungen finden Sie im Benutzerhandbuch der SPS. Wenn die Referenznummer außerhalb des unterstützten Bereichs liegt, wird ein Fehler zurückgegeben.

SPS-Typen **Verfügbarkeit von F145F146_MODBUS_MASTER (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
Port	ANY16	Gibt den COM-Port (1 oder 2) des Slaves über die Systemvariable an: SYS_COM1_PORT SYS_COM2_PORT
SlaveAddress		Addresse der Gegenstelle (0–255).
FunctionCode*		SYS_MODBUS_01_READ_COIL SYS_MODBUS_02_READ_INPUT SYS_MODBUS_03_READ_HOLDING_REGISTER SYS_MODBUS_04_READ_INPUT_REGISTERS SYS_MODBUS_05_FORCE_COIL SYS_MODBUS_06_PRESET_REGISTER SYS_MODBUS_15_FORCE_COILS SYS_MODBUS_16_PRESET_REGISTERS
StartRegister		Anfangsadresse (0–65535). Der Adresstyp hängt vom Befehl ab, der durch **FunctionCode*** angegeben wird.
NumberOfRegisters*		Anzahl der Übertragungs-Bits oder -Worte. 1–2040 für Funktionscodes 01, 02 2–2040 für Funktionscode 15 1–127für Funktionscodes 03, 04 2–127 für Funktionscode 16
MasterData	ANY	Die Daten des Masters, die in den Slave geschrieben werden.

Operanden

Für	Relais				T/C		Register			Konstante
Port	WX	WY	WR	WL			DT	LD	FL	-
Slave Address	WX	WY	WR	WL	-	-	DT	LD	FL	-
Function Code*	-	-	-	-	-	-	-	-	-	vom System definiert
Start Register	WX	WY	WR	WL	-	-	DT	LD	FL	-
NumberOf Registers*	-	-	-	-	-	-	-	-	-	dez. oder hex.
Master Data	WX	WY	WR	WL	-	-	DT	LD	FL	-

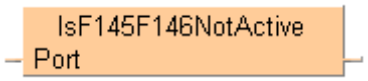
Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Port 0 (Übertragung an alle Teilnehmer) keine Antwort von COM1 oder COM2 erhält ▪ Slave-Daten oder Master-Daten den verfügbaren Adressbereich überschreiten ▪ Als Kommunikationsart (s. S. 681) **nicht** MEWTOCOL-COM Master/Slave oder Modbus RTU Master/Slave eingestellt ist ▪ Der ausgewählte COM-Port ein Schnittstellenmodul erfordert, das nicht installiert ist
R9008	%MX0.900.8	permanent	

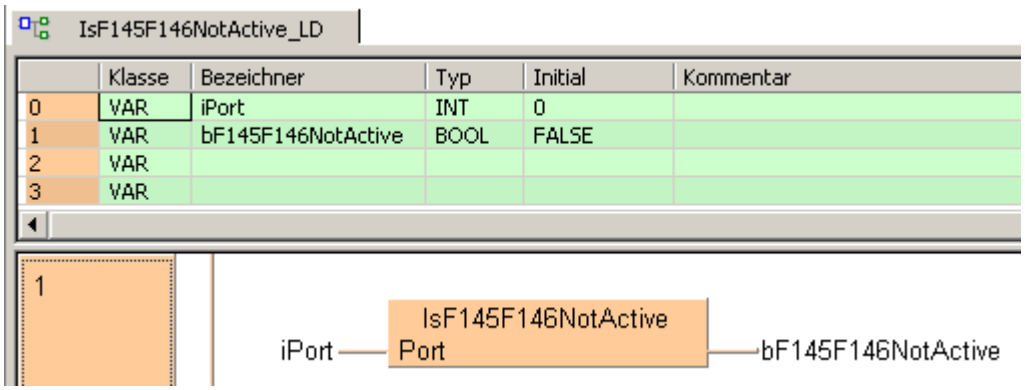
Is145F146NotActive

Liefert den Wert des Merkers "F145F146 Not Active"

Erklärung Diese Funktion liefert den Wert des Merkers "F145F146 Not Active", der anzeigt, ob an einer COM-Schnittstelle der SPS bereits ein Modbus-Befehl ausgeführt wird.



Beispiel:



Der Merker ist abhängig von der Steuerung:

SPS	Nummer der Schnitt-stelle	Schnittstelle	Merker	Systemvariable
FP-Sigma, FP-X	0	TOOL-Schnittstelle (nicht bei FP-Sigma, 12k)	gibt immer TRUE zurück	-
	1	COM1-Schnittstelle	R9044	sys_bIsComPort1F145F146_NotActive
	2	COM2-Schnittstelle	R904A	sys_bIsComPort2F145F146_NotActive
FP0, FP-e	-	-	gibt immer TRUE zurück	-
FP2, FP2SH	0	COM-Schnittstelle an CPU	gibt immer TRUE zurück	-
	16#xx01	COM1 an MCU-Modul in Steckplatz xx	gibt immer TRUE zurück	-
	16#xx02	COM2 an MCU-Modul in Steckplatz xx		

Detaillierte Informationen zur Benutzung von Systemvariablen finden Sie unter Datenübertragung von/zu Sonderdatenregistern (s. S. 817).

IsReceptionDone für bestimmte COM-Schnittstelle mit entsprechender Systemvariable

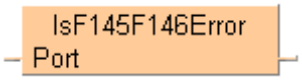
Sie können die folgenden Systemvariablen dazu verwenden, den Merker IsF145F146NotActive für eine bestimmte COM-Schnittstelle auszuwerten:

- sys_bIsComPort1F145F146NotActive
- sys_bIsComPort2F145F146NotActive

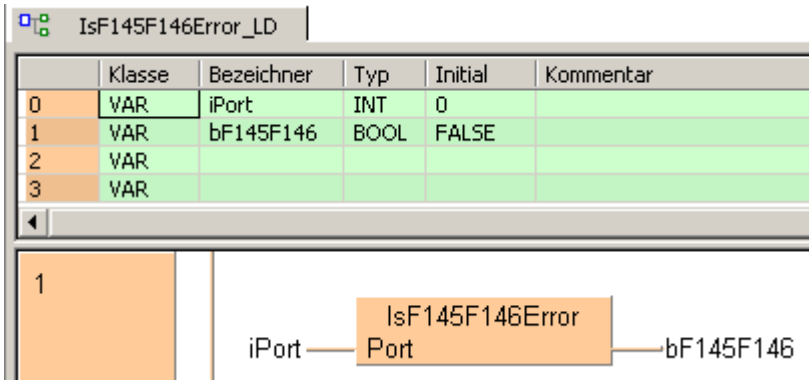
IsF145F146Error

Liefert den Wert des Merkers "Modbus Error"

Erklärung Diese Funktion liefert den Wert des Merkers "F145F146 Error", der anzeigt, ob der Befehl erfolgreich an der COM-Schnittstelle ausgeführt wurde.



Beispiel:



Der Merker ist abhängig von der Steuerung:

SPS	Nummer der Schnittstelle	Schnittstelle	Merker	Systemvariable
FP-Sigma, FP-X	0	TOOL-Schnittstelle (nicht bei FP-Sigma, 12k)	gibt immer FALSE zurück	-
	1	COM1-Schnittstelle	R9045	sys_bIsComPort1F145F146Error
	2	COM2-Schnittstelle	R904B	sys_bIsComPort2F145F146Error
FP0R	0	TOOL-Schnittstelle	gibt immer FALSE zurück	-
	1	TOOL-Schnittstelle	R9045	sys_bIsComPort1F145F146Error
FP0, FP-e	-	-	gibt immer FALSE zurück	-
FP2, FP2SH	0	COM-Schnittstelle an CPU	gibt immer FALSE zurück	-
	16#xx01	COM1 an MCU-Modul in Steckplatz xx	gibt immer FALSE zurück	-
	16#xx02	COM2 an MCU-Modul in Steckplatz xx		

Detaillierte Informationen zur Benutzung von Systemvariablen finden Sie unter Datenübertragung von/zu Sonderdatenregistern (s. S. 817).

20.6 SPS-Kopplung

Die SPS-Kopplung ist eine einfache Möglichkeit, mehrere Steuerungen über eine verdrillte Zweidrahtleitung und das MEWNET-Protokoll zu verbinden. Bei der SPS-Kopplung werden die Daten in allen miteinander vernetzten Steuerungen über interne Merker, sogenannte Koppelmerker (L), und Datenregister, sogenannte Koppeldatenregister (LD), gemeinsam gehalten. Ändert sich der Zustand eines Koppelmerkers oder der Inhalt eines Koppeldatenregisters in einer SPS, wird diese Änderung automatisch an die anderen Steuerungen im Verbund weitergegeben. Die Koppelmerker und -datenregister der Steuerungen enthalten Bereiche zum Senden und Bereiche zum Empfangen von Daten. Teilnehmeradressen und Koppelbereiche werden über die Systemregister festgelegt.

Detaillierte Informationen zum Einstellen der Kommunikationsparameter und des Koppelbereichs finden Sie in den Hardware-Handbüchern der jeweiligen Module.

◆REFERENZ

Nähere Informationen finden Sie in den zugehörigen Hardware-Handbüchern.

Kapitel 21

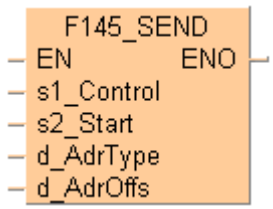
Datenübertragung per Netzwerk

21.1 Senden und Empfangen über MEWNET

In diesem Abschnitt:

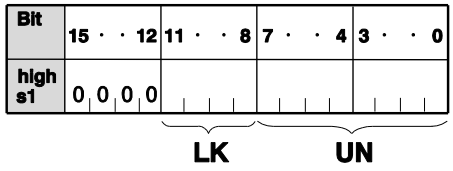
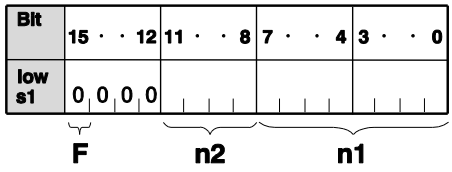
F145_SEND Senden über MEWNET

Erklärung Die Daten des Speicherbereichs mit der Anfangsadresse **s2** des sendenden CPU-Moduls werden in den Speicherbereich mit der Anfangsadresse **d+n*** des gekoppelten CPU-Moduls über Lichtwellenleiter übertragen.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Spezifikationen von s1:

s1 höherwertiges Byte		**s1 niederwertiges Byte**	
			
1. Koppelnummer (LK: 1 bis 3)		**1. Worttransfer**	
Bis zu 3 Koppelmodule können mit einer CPU verbunden werden.		F = 0	Worttransfer
		n2 = 0	Das Wort wird transferiert wenn "0" gesetzt ist.
Dieses Koppelmodul (LK) wählt das Quellkoppelmodul der 3 aus.		n1 = 11 bis 16	Anzahl der zu sendenden Worte
2. Adresse des empfangenden Koppelmoduls (UN: 1 bis 63)		**2. Bittransfer**	
Bis zu 63 Adressen können mit einem Koppelmodul verbunden werden.		F = 1	Bittransfer
Diese (UN) legt die Koppelstellenadresse des empfangenden Moduls fest.		n2 = 0 bis 15	Bitnummer im Zielbereich
		n1 = 0 bis 15	Bitnummer im Quellbereich

REFERENZ

Weitere Informationen und Programmbeispiele entnehmen Sie bitte der technischen Spezifikation zum jeweiligen Spezialmodul.

SPS-Typen **Verfügbarkeit von F145_SEND (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s1	DWORD	32-Bit-Speicherregister für die Steuerdaten
s2	ANY16	16-Bit-Speicherregister für die Anfangsadresse der zu sendenden Daten
d		16-Bit-Speicherregister für die Anfangsadresse des Zielbereichs Vergewissern Sie sich diesen Bereich zu wählen, indem sie die Adresse auf 0 setzen (z.B. DT0 oder WR0, ...) (Zieldatenregister an einer anderen Station).
n		Offset für die Anfangsadresse des Zielkoppelbereichs **d** (Zieldatenregister an einer anderen Station)

Die Variablen **s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
d	-	WY	WR	WL	SV	EV	DT	LD	FL	
n	-	-	-	-	-	-	-	-	-	dez., hex.

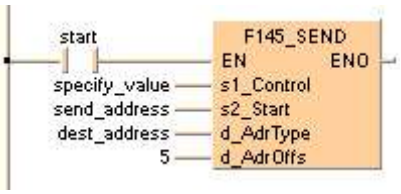
Beispiel In diesem Beispiel wird die Funktion F145_SEND im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	specify_value	DWORD	0	stores the control data
2	VAR	send_address	WORD	0	Starting 16-bit area for
3	VAR	dest_address	WORD	0	Type of destination
4	VAR	n	INT	0	operands for storing data
5	VAR				in the destination station

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

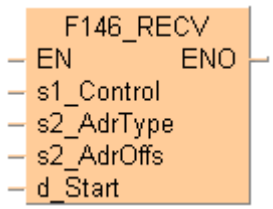


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F145_SEND( specify_value, send_address, dest_address, 5);
END_IF;
```

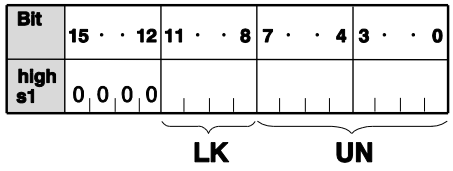
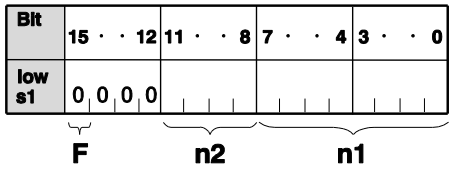
F146_RECV Empfangen über MEWNET

Erklärung Die Daten des Speicherbereichs mit der Anfangsadresse **s2+n*** des sendenden CPU-Moduls werden in den Speicherbereich mit der Anfangsadresse **d** des gekoppelten CPU-Moduls über Lichtwellenleiter übertragen.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Spezifikationen von s1:

s1 höherwertiges Byte		**s1 niederwertiges Byte**	
			
1. Koppelnummer (LK: 1 bis 3)		**1. Worttransfer**	
Bis zu 3 Koppelmodule können mit einer CPU verbunden werden.		F = 0	Worttransfer
		n2 = 0	Das Wort wird transferiert wenn "0" gesetzt ist.
Dieses Koppelmodul (LK) wählt das Quellkoppelmodul der 3 aus.		n1 = 11 bis 16	Anzahl der zu sendenden Worte
2. Adresse des empfangenden Koppelmoduls (UN: 1 bis 63)		**2. Bittransfer**	
Bis zu 63 Adressen können mit einem Koppelmodul verbunden werden.		F = 1	Bittransfer
Diese (UN) legt die Koppelstellenadresse des empfangenden Moduls fest.		n2 = 0 bis 15	Bitnummer im Zielbereich
		n1 = 0 bis 15	Bitnummer im Quellbereich

REFERENZ

Weitere Informationen und Programmbeispiele entnehmen Sie bitte der technischen Spezifikation zum jeweiligen Spezialmodul.

SPS-Typen **Verfügbarkeit von F146_RECV (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s1	DWORD	32-Bit-Speicherregister für die Steuerdaten
s2	ANY16	Speicherregister in der Zielstation für Arten von Zieloperanden. Vergewissern Sie sich diesen Bereich zu wählen, indem sie die Adresse auf 0 setzen (z.B. DT0 oder WR0, ...). (Zieldatenregister an einer anderen Station)
d		16-Bit-Anfangsadresse für die Quelloperanden spezifiziert in s2 (Quelldatenbereich in einer anderen Station)
n		16-Bit-Speicherregister für die Anfangsadresse der zu empfangenden Daten (Zieldatenbereich in einer anderen Quellstation)

Die Variablen **s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
d	-	WY	WR	WL	SV	EV	DT	LD	FL	
n	-	-	-	-	-	-	-	-	-	dez., hex.

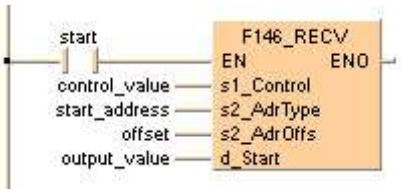
Beispiel In diesem Beispiel wird die Funktion F146_RECV im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	control_value	DWORD	0	32-bit area for storing control data
2	VAR	start_address	WORD	0	Starting 16-bit area address for
3	VAR	output_value	WORD	0	Starting 16-bit area address for storing data received
4	VAR				

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F146_RECV( s1_Control:= control_value, s2_AdrType:= start_address,
n_AdrOffs:= offset,
               d_Start:= output_value);
END_IF;
```

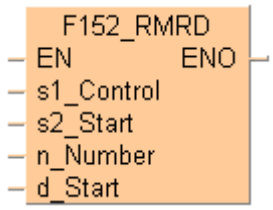
21.2 Datenaustausch über den gemeinsam genutzten Datenspeicher einer MEWNET-F-Slave Station

In diesem Abschnitt:

F152_RMRD

Lesen vom dezentralen Spezialmodul

Erklärung Von einem Spezialmodul (Intelligentes Modul mit eigenem Speicher zur Realisierung spezieller E/A-Funktionen) werden **n** Worte ab der Anfangsadresse **s2** gelesen und in der CPU im Speicherbereich mit der Anfangsadresse **d** gespeichert.

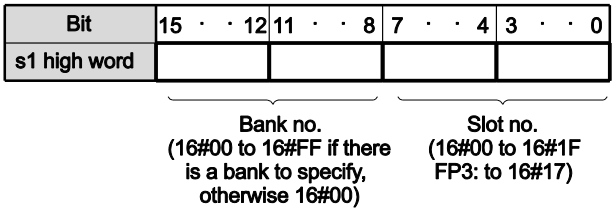


s1 speichert die Kontrolldaten für die Konfiguration der Master- und Slavemodule im Netzwerk. **n** Worte werden ab der gemeinsamen Speicheradressnummer **s2** im Spezialmodul gelesen. Das Ergebnis wird in **d** gespeichert.

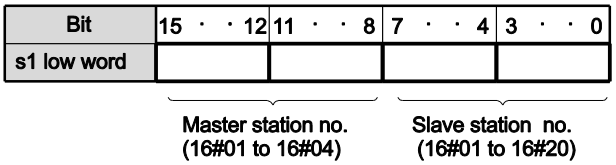
Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Spezifikationen von s1:

s1 higher word



s1 lower word



Hinweis: Intelligentes Modul mit Speicherbank

Bezeichnung	Bestellnummer
FP3 Datenspeichermodul	AFP32091 AFP32092

SPS-Typen **Verfügbarkeit von F152_RMRD (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s1	DWORD	speichert die Steuerdaten der Master/Slave-Konfiguration
s2	ANY16	Anfangsadresse für die Anzahl der Worte, die gelesen werden sollen
n	INT	Anzahl der Worte, die gelesen werden sollen (max. 32 Worte)

d	ANY16	Anfangsadresse (16 Bit), in der die gelesenen Worte gespeichert werden (siehe F153 (s. S. 752))

Die Variablen **s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
s2, n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Steuerdaten **s1** die Grenze des festgelegten Bereichs überschreiten ▪ kein MEWNET-F Master_Modul gefunden wird ▪ die gelesenen Daten den Bereich von **d** überschreiten
R9008	%MX0.900.8	kurzzeitig	

■ Hinweise zur Programmierung

Es ist nicht möglich mehrere **F152_RMRD** und **F153_RMWT** Befehle gleichzeitig auszuführen. Das Programm sollte so aufgesetzt sein, dass diese Befehle ausgeführt werden, wenn der Freigabe-Merker (R9035) des Befehls **F152_RMRD/F153_RMWT** auf EIN gesetzt ist.

R9035
- 0: Befehl kann nicht ausgeführt werden, weil bereits ein RMRD– oder RMWT–Befehl ausgeführt wird

1: Befehl kann ausgeführt werden

Der Befehl **F152_RMRD** kann nur eine Anfrage annehmen. Der tatsächliche Verarbeitung wird am Ende der Prüfung ausgeführt. Der Bearbeitungsende-Merker (R9036) des Befehls **F152_RMRD/F153_RMWT** kann zur Bestätigung dafür verwendet werden, ob der Befehl abgearbeitet wurde oder nicht.

R9036	0: Kein Fehler 1: Fehler (der Fehlercode wird in DT9036/DT90036 gespeichert)
DT9036 (DT90036)	Speichert den Fehlercode, wenn bei der Übertragung ein Fehler aufgetreten ist (R9036: 1).

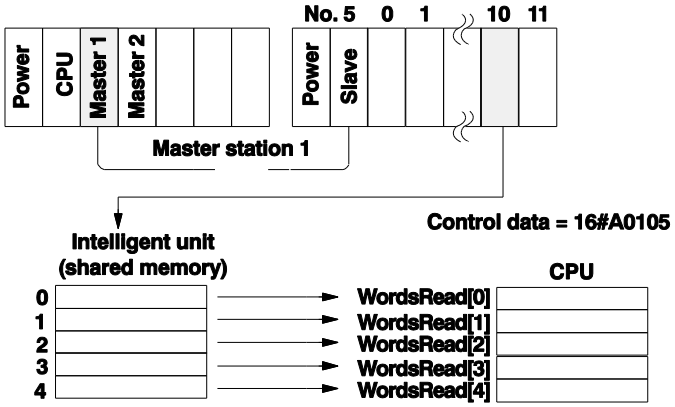
Hinweis: Die Fehlercodes werden in DT9036/DT90036 gespeichert

Fehlercode (HEX)	Beschreibung
16#5B	Wartezeitfehler (am angegebenen Ort wurde kein Spezialmodul gefunden)
16#68	Kein Speicherfehler (an der angegebenen Adresse gibt es keinen Speicher)
16#71	Antwort auf Wartezeitfehler senden
16#72	Wartezeitfehler "Datenpuffer voll" senden
16#73	Wartezeitfehler "Ansprechzeit"

Die Ausgabe der Fehlercodes 16#71 bis 16#73 bedeutet, dass die Kommunikation nach Überschreiten der Wartezeit abgebrochen wurde. Die Wartezeit (Timeout) kann in den Einstellungen des Systemregisters 32, in einem Bereich von 10,0ms bis 81,9s (in 10ms Einheiten), eingestellt werden. Der Wert 2s ist voreingestellt.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

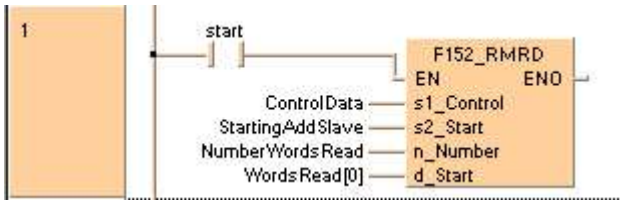
Es werden 5 Worte ab Adresse 0 bis 4 im gemeinsam genutzten Speicher des Spezialmoduls des Slave gelesen. Die gelesenen Daten werden im ARRAY **WordsRead** der Master "CPU" gespeichert, sobald **Start** gesetzt ist.



POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Start	BOOL	FALSE	
1	VAR	Control Data	DWORD	16#000A0105	No bank, slot no. 10, Master station 1, Slave station 5
2	VAR	StartingAddSlave	WORD	0	
3	VAR	NumberWordsRead	INT	5	
4	VAR	WordsRead	ARRAY [0..4] OF WORD		

KOP

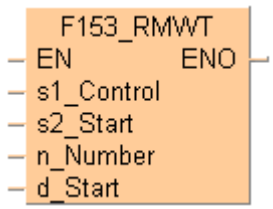


F153_RMWT

Schreiben zum dezentralen Spezialmodul

Erklärung

Der Befehl F/P153 wird zum Schreiben von Daten zu einem Spezialmodul benutzt, das über ein E/A-Master-Koppelmodul (im Baugruppenträger der lesenden CPU) und ein E/A-Slave-Koppelmodul (im Baugruppenträger des Spezialmoduls) dezentral betrieben wird.



s1 speichert die Steuerdaten der Master/Slave-Konfiguration. In ein Spezialmodul (Intelligentes Modul mit eigenem Speicher zur Realisierung spezieller E/A-Funktionen) werden **n** Worte ab der Anfangsadresse **d** geschrieben. Der Quellspeicher der CPU wird mit der Anfangsadresse **s2** spezifiziert.

Spezifikationen von s1:

s1 higher word

Bit	15 · · 12	11 · · 8	7 · · 4	3 · · 0
s1 high word				

Bank no. (16#00 to 16#FF if there is a bank to specify, otherwise 16#00)

Slot no. (16#00 to 16#1F FP3: to 16#17)

s1 lower word

Bit	15 · · 12	11 · · 8	7 · · 4	3 · · 0
s1 low word				

Master station no. (16#01 to 16#04)

Slave station no. (16#01 to 16#20)

Hinweis: Intelligentes Modul mit Speicherbank

Bezeichnung	Bestellnummer
FP3 Datenspeichermodul	AFP32091 AFP32092

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F153_RMWT (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s1	DWORD	speichert die Steuerdaten der Master/Slave-Konfiguration
s2	ANY16	Anfangsadresse (16 Bit) der CPU, von der die Wörter gelesen werden
n	INT	Anzahl der Wörter, die gelesen werden und dann zur Empfangsstation geschrieben werden (max. 32 Wörter)
d	ANY16	Anfangsspeicher-Nr. des intelligenten Speichermoduls zu der die Wörter geschrieben werden

Die Variablen **s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
n, d	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Fehlermerker

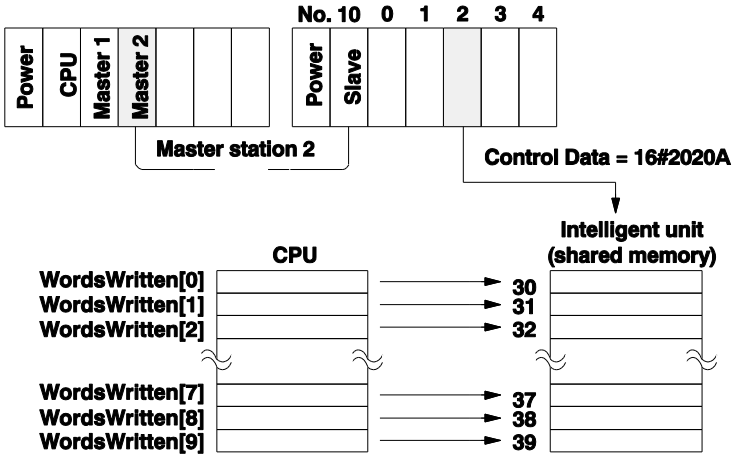
Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Steuerdaten **s1** die Grenze des festgelegten Bereichs überschreiten ▪ kein MEWNET-F Master_Modul gefunden wird ▪ die gelesenen Daten den Bereich von **s2** überschreiten
R9008	%MX0.900.8	kurzzeitig	

Hinweise zur Programmierung: siehe F152_RMRD (s. S. 749)

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

Es werden 20 Worte, die im ARRAY **WordsWritten[0]..[9]** der Master "CPU" gespeichert sind, ab Adresse 30 bis 39 in den gemeinsam genutzten Speicher des Spezialmoduls des Slave geschrieben, sobald **Start** gesetzt ist.

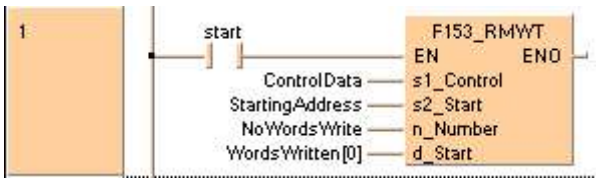


POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Start	BOOL	FALSE	
1	VAR	Control Data	DWORD	16#2020A	No bank, slot no. 10, Master station 1, Slave station 5
2	VAR	StartingAddress	WORD	30	
3	VAR	NoWordsWrite	INT	10	
4	VAR	WordsWritten	ARRAY [0..14] OF WORD	[15(0)]	

KOP


1
start
F153_RMWT
EN
ENO
ControlData
s1_Control
StartingAddress
s2_Start
NoWordsWrite
n_Number
WordsWritten[0]
d_Start

21.3 Datenaustausch mit flexiblem Netzwerk

In diesem Abschnitt:

FNS_InitConfigDataTable

Funktion

Erklärung Die Funktion FNS_InitConfigDataTable erstellt ConfigDataTable aus der Variablen **ProcessDataTable**, bei der es sich um einen Datentyp aus einem oder mehreren Elementen handeln kann. ConfigDataTable ist notwendig, um den FP-FNS-Baustein mit den Funktionsbausteinen FNS-Profibus, FNS-DeviceNet und FNS-CanOpen zu konfigurieren.

Stellen Sie sicher, dass die Größe der Variablen **ConfigDataTable** der Struktur von **ProcessDataTable** entspricht, d.h. wenn **ProcessDataTable** aus drei Einträgen besteht, muss die Variable **ConfigDataTable** ein "Array[0..2] of WORD" sein, da dessen Größe der Anzahl der Einträge entspricht. Wenn die Variable **ProcessDataTable** nur einen Eintrag besitzt (z.B. WORD), muss die Variable **ConfigDataTable** ein "Array[0..0] of WORD" (mit der Größe 1) sein.

Für die Eingabe in **FNS_InitConfigDataTable** zulässige Datentypen sind alle Variablen mit 16-Bit (INT, WORD), 32-Bit (DINT, DWORD, TIME (32 Bit), REAL) und 64-Bit oder Arrays von diesen. 64-Bit-Variablen sind als zweidimensionale Arrays definiert, z.B. "Array[0..0,0..3] of INT" ist eine 64-Bit-Variable, während "Array[0..3] of INT" einen Array mit vier Elementen der 16-Bit-Variablen darstellt.

Die Datentypen BOOL, STRING und Arrays dieser Typen sind bei der Eingabe der Funktion FNS_InitConfigDataTable NICHT zulässig.

Die Ausgabe **ConfigDataTable** der Funktion muss ein Array von WORD sein.

SPS-Typen verfügbar für FP2/FP2SH und FPΣ.

Datentypen

Variable	Datentypen	Funktion
ProcessDataTable	INT, WORD ,DINT, DWORD, REAL, TIME und ARRAYS für diese Typen	{0>Eingangs- und Ausgangsprozessdatenvariablen
ConfigDataTable	ARRAY of WORD	Konfigurationsdaten für FP-FNS-Funktionsbausteine. Die Arraygröße der Variablen ConfigDataTable muss der Anzahl der Elemente der Variablen ProcessDataTable entsprechen.

ProcessDataTable

Die folgende Syntaxtabelle zeigt, wie die 16-Bit-, 32-Bit- und 64-Bit-Variablen und die Arrays davon deklariert werden, wenn sie als ProcessDataTable-Eingang für die Funktion FNS_InitConfigDataTable verwendet werden sollen.

Eingangsdatentyp	Eingangsgröße	Anmerkung
INT, WORD	16-Bit	
DINT, WORD, REAL, TIME	32-Bit	
Array[0..0,0..3] of INT **Array[0..0,0..3] of WORD**	64-Bit	zweidimensionaler Array; Größe der zweiten Dimension = 4
Array[a ..b] of INT/Array[a ..b] of WORD	Array of 16-Bit Größe = b-a+1	eindimensionaler Array
Array[a ..b] of DINT/Array[a ..b] of DWORD/ **Array[a ..b] of REAL/Array[a ..b] of TIME**	Array of 32-Bit Größe = b-a+1	eindimensionaler Array
Array[0..x,0..3] of INT **Array[0..x,0..3] of WORD**	Array of 64-Bit Größe = x	zweidimensionaler Array; Größe der zweiten Dimension = 4

Operanden

Für	Merker				T/C		Register			Konstante
ProcessDataTable	-	-	-	-	-	-	DT	FL	-	-
ConfigDataTable	-	-	-	-	-	-	DT	FL	-	-

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet.

GVL

In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projektes verwendet werden können.

Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial	Autoextern
0	VAR_GLOBAL	ProcessData			ProcessDataStructure		☑

POE Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR_EXTERNAL	ProcessData		
1	VAR	ConfigData	ARRAY [0..2] OF WORD	[3(0)]

Die Arraygröße der Variablen **ConfigDataTable** muss der Anzahl der Elemente der Variablen **ProcessData** entsprechen.

In diesem Beispiel ist die Variable **ProcessData** ein SDT des Typs ProcessDataStructure mit der folgenden Struktur:

ProcessDataStructure [SDT]

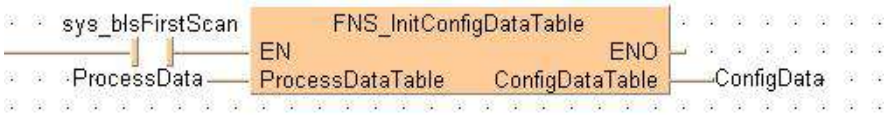
	Bezeichner	Typ	Initial
0	Data1_16bits	INT	0
1	Data2_32bits	DWORD	0
2	Data3_arraysize2_64bits	ARRAY [0..1,0..3] OF INT	[8(0)]

Da der SDT aus drei Einträgen besteht, muss die Ausgangsvariable **ConfigData** ein Array of WORD mit der Größe 3 sein (z.B.: Array [0..2] of WORD).

Rumpf

Wenn **sys_bIsFirstScan** TRUE ist, d.h. im ersten Zyklus, wird die Funktion ausgeführt. Der Wert der Variablen **ConfigData** entspricht der Struktur der Eingangsvariablen **ProcessData** sowie der Anzahl und Typen der Elemente.

LD



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

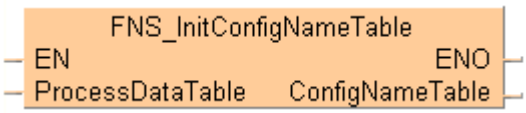
```
If sys_bIsFirstScan then
      ConfigData:=FNS_InitConfigDataTable(ProcessData);
end_if;
```

FNS_InitConfigNameTable

Function

Erklärung

This function creates a ConfigNameTable from the variable **ProcessDataTable**, which can be a single-element data type or a mulit-element data type.



☞

- **Stellen Sie sicher, dass die Größe der Variable ConfigNameTable der Struktur von ProcessDataTable entspricht, z.B. wenn ProcessDataTable aus drei Einträgen besteht, dann sollte die Variable ConfigNameTable ein "Array[0..2] of WORD" sein, dessen Größe der Anzahl der Einträge entspricht. Wenn die Variable ProcessDataTable nur über einen Eintrag verfügt (z.B. WORD), dann sollte die Variable ConfigNameTable ein "Array[0..0] of WORD" (mit der Größe 1) sein.**
- **Zulässige Eingangsdatentypen sind alle 16-Bit (INT, WORD), 32-Bit (DINT, DWORD, TIME (32 Bit), REAL) und 64-Bit Variablen oder deren Arrays. Die 64-Bit Variablen sind als 2-dimensionale Arrays definiert, z.B. "Array[0..0,0..3] of INT" ist eine 64-Bit Variable, während "Array[0..3] of INT" einen Array mit vier Elementen einer 16-Bit Variable darstellt.**
- **Die Datentypen BOOL, STRING und Arrays dieses Typs sind in einer Eingangsvariable NICHT zulässig.**
- **Die Ausgabe ConfigNameTable der Funktion muss ein Array von WORD sein.**

SPS Typen

s. S. 1192

Datentypen

Variable	Data types	Function
ProcessDataTable	INT, WORD, DINT, DWORD, REAL, TIME, and ARRAYS of these types	Input and output of process data variables
ConfigNameTable	ARRAY of WORD	Configuration data for FP-FNS blocks. The array-size of the variable ConfigNameTable has to correspond to the number of elements of the ProcessDataTable variable.

Process-DataTable

The following syntax table shows how to declare 16-bit, 32-bit and 64-bit variables and arrays thereof when using them as ProcessDataTable input.

Input Data type	Size of Input	Comment
INT, WORD	16-bit	
DINT, WORD, REAL, TIME	32-bit	
Array[0..0,0..3] of INT **Array[0..0,0..3] of WORD**	64-bit	2-dimensional array; size of second dimension = 4
Array[a ..b] of INT/Array[a ..b] of WORD	Array of 16-bit Size = b-a+1	1-dimensional array

Input Data type	Size of Input	Comment
Array[a ..b] of DINT/Array[a ..b] of DWORD/ Array[a ..b] of REAL/Array[a ..b] of TIME	Array of 32-bit Size = b-a+1	1-dimensional array
Array[0..x,0..3] of INT Array[0..x,0..3] of WORD	Array of 64-bit Size = x+1	2-dimensional array; size of second dimension = 4

Operanden

For	Relay				T/C		Register			Constant
ProcessDataTable	-	-	-	-	-	-	DT	FL	-	-
ConfigNameTable	-	-	-	-	-	-	DT	FL	-	-

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

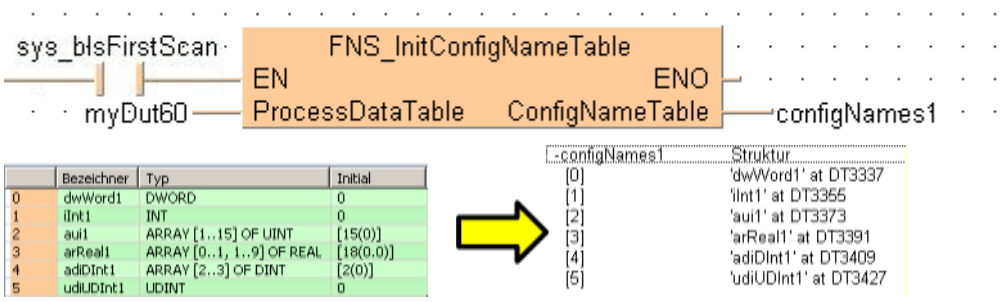
	Class	Identifier	Type	Initial
0	VAR	adiDInt1	ARRAY[2..3] OF DINT	[2(0)]
1	VAR	arReal1	ARRAY[0..1, 1..9] OF REAL	[18(0.0)]
2	VAR	aui1	ARRAY[1..15] OF UINT	[15(0)]
3	VAR	dwWord1	DWORD	0
4	VAR	iInt1	INT	0
5	VAR	udiUDInt1	UDINT	0

Die Größe der Variable **configNames1** muss der Anzahl der Elemente in der Eingangsvariable **myDUT60** entsprechen.

Da der SDT aus drei Einträgen besteht, muss die Ausgangsvariable **configNames1** ein Array von WORD mit der Größe 3 sein (z.B.: Array [0..2] of WORD).

Rumpf Wenn **sys_bIsFirstScan** TRUE ist, d.h. im ersten Zyklus, wird die Funktion ausgeführt. Der Wert der Variable **configNames1** entspricht der Struktur der Eingangsvariable **myDUT60** sowie der Anzahl und Typen der Elemente.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
If sys_bIsFirstScan then
        ConfigNames1:=FNS_InitConfigNameTable(myDUT60);
end_if;
```

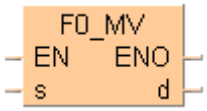
Kapitel 22

Dataübertragung in der SPS

F0_MV

16-Bit-Transfer

Erklärung Der binäre 16-Bit-Wert des Speicherregisters **s** oder eine Konstante s wird in den 16-Bit-Bereich des Speicherregisters **d** kopiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden MOVE (s. S. 59). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen **Verfügbarkeit von F0_MV (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	16-Bit-Quellregister
d		16-Bit-Zielregister

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Erklärung am Beispielwert 16#0089

Quelle

Bit	15 . . 12	11 . . 8	7 . . 4	3 . . 0
s	0 0 0 0	0 0 0 0	1 0 0 0	1 0 0 1

Ziel

Bit	15 . . 12	11 . . 8	7 . . 4	3 . . 0
d	0 0 0 0	0 0 0 0	1 0 0 0	1 0 0 1

Sollwert dieses Beispiels: 16#0089

Beispiel In diesem Beispiel wird die Funktion F0_MV im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	INT	137	contains the source value
2	VAR	output_value	INT	0	the area, where the source value will be copied to. result after a 0->1 leading edge from start: 137
3	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

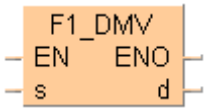
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F0_MV(input_value, output_value);
END_IF;
```

F1_DMV

32-Bit-Transfer

Erklärung Der binäre 32-Bit-Wert des Speicherregisters **s** oder eine Konstante s wird in den 32-Bit-Bereich des Speicherregisters **d** kopiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden MOVE (s. S. 59).

Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen **Verfügbarkeit von F1_DMV (s. S. 1186)**

Datentypen

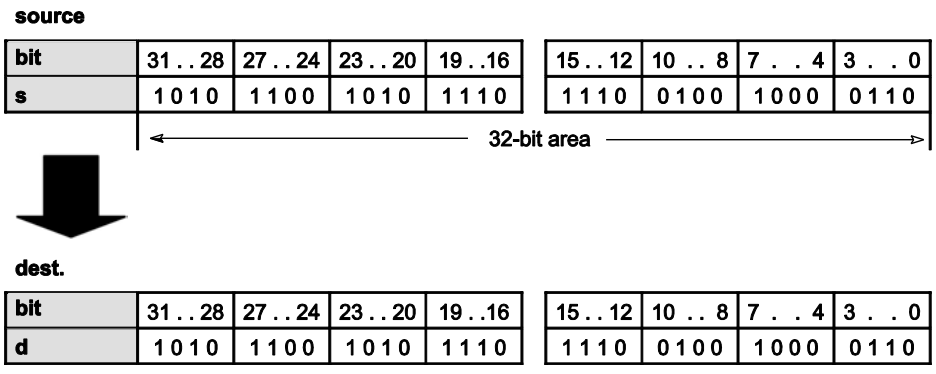
Variable	Datentyp	Funktion
s	ANY32	32-Bit-Quellregister
d		32-Bit-Zielregister

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Erklärung am Beispielwert 16#ACAEE486



Sollwert dieses Beispiels: 16#ACAEE486

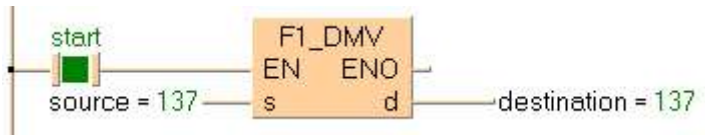
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	source	DINT	137	contains the source value
2	VAR	destination	DINT	0	the area, where the source value will be copied to
3	VAR				result after a 0->1 leading edge from start: 137

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



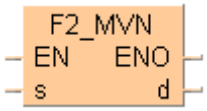
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F1_DMV(source, destination);
END_IF;
```

F2_MVN

16-Bit-Transfer invertiert

Erklärung Der binäre 16-Bit-Wert des Speicherregisters **s** oder eine Konstante **s** wird invertiert und in den 16-Bit-Bereich des Speicherregisters **d** übertragen. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F2_MVN (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	zu invertierendes 16-Bit-Quellregister
d		16-Bit-Zielregister

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Erklärung am Beispielwert 16#5555

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
source bit	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dest. bit	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Jedes Bit wird invertiert, Zielwert dieses Beispiels: 16#AAAA

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	WORD	16#1234	this value will be
2	VAR	output_value	WORD	0	result after a 0->1 leading edge from start: 16#EDCB

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

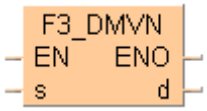
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F2_MVN(input_value, output_value);
END_IF;
```

F3_DMVN

32-Bit-Transfer invertiert

Erklärung Der binäre 32-Bit-Wert des Speicherregisters **s** oder eine Konstante **s** wird invertiert und in den 32-Bit-Bereich des Speicherregisters **d** übertragen. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F3_DMVN (s. S. 1189)**

Datentypen

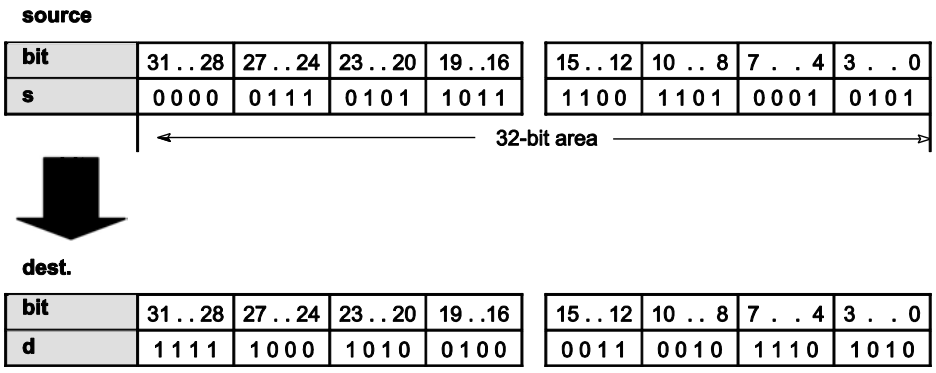
Variable	Datentyp	Funktion
s	ANY32	zu invertierendes 32-Bit-Quellregister
d		32-Bit-Zielregister

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Erklärung am Beispielwert 16#75BCD15



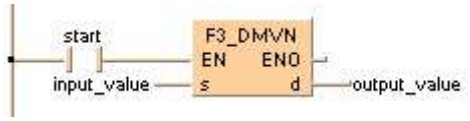
Jedes Bit wird invertiert, Zielwert dieses Beispiels: 16#F8A432EA

Beispiel In diesem Beispiel wird die Funktion F3_DMVN im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



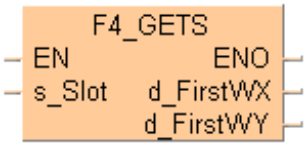
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F3_DMVN(input_value, output_value);
END_IF;
```

F4_GETS

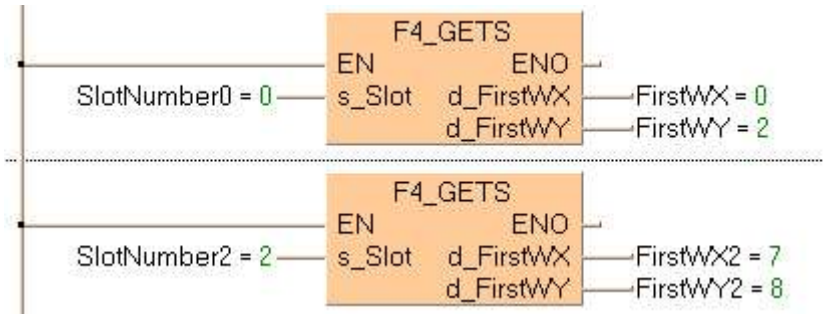
Offset des ersten WX und WY im angegebenen Steckplatz lesen

Erklärung Der Offset des ersten Wortes im angegebenen Steckplatz wird gelesen.



Doppelklicken Sie im Projektnavigator auf "E/A-Adressen", um die E/A-Adressen manuell zu konfigurieren.

Beispiel:



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F4_GETS (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s_Slot	ANY16	Steckplatznummer von der die Information benötigt wird
d_FirstWX		Offset des ersten WX im angegebenen Steckplatz
d_FirstWY		Offset des ersten WY im angegebenen Steckplatz

Operanden

Für	Merker				T/C		Register			Konstante
s_Slot	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d_FirstWX	WX	WY	WR	WL	SV	EV	DT	LD	FL	
d_FirstWY	-	WY	WR	WL	SV	EV	DT	LD	FL	-

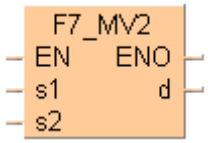
Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die mit Indexmodifizierern definierte Adresse den zulässigen Bereich überschreitet
R9008	%MX0.900.8	kurzzeitig	▪ die Steckplatznummer kein Wert von 0-31 ist

F7_MV2

Zwei 16-Bit-Daten Transfer

Erklärung Die zwei 16-Bit-Werte der Speicherregister **s1** und **s2** bzw. die zwei Konstanten **s1** und **s2** werden in das 32-Bit-Speicherregister **d** kopiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

☞ **Mit dem Befehl F190_MV3 (s. S. 811) oder P190_MV3 können Sie drei 16-Bit-Werte auf einmal transferieren.**

SPS-Typen **Verfügbarkeit von F7_MV2 (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1, s2	ANY16	16-Bit-Quellregister
d	ANY32	32-Bit-Zielregister

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

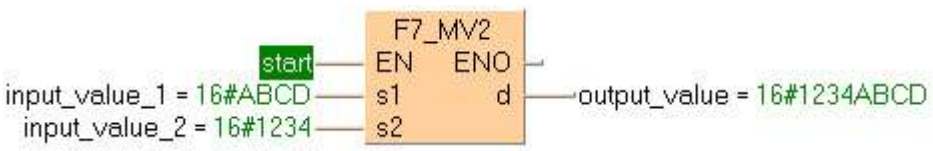
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value1	WORD	16#ABCD	
2	VAR	input_value2	WORD	16#1234	
3	VAR	output_value	DWORD	0	result after a 0->1 leading edge from start: 16#1234ABCD
4	VAR				

Hier wurden die Eingangsvariablen **input_value_1** und **input_value _2** deklariert. Statt dessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



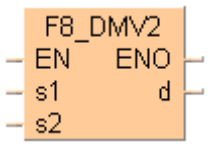
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F7_MV2(input_value1, input_value2, output_value);
END_IF;
```

F8_DMV2

Zwei 32-Bit-Daten Transfer

Erklärung Die Funktion kopiert zwei 32-Bit-Werte an den Eingängen **s1** und **s2** in ein 32-Bit-ARRAY mit zwei Elementen am Ausgang **d**.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

☞ **Mit dem Befehl F191_DMV3 (s. S. 813) oder P191_DMV3 können Sie drei 32-Bit-Werte auf einmal kopieren.**

SPS-Typen **Verfügbarkeit von F8_DMV2 (s. S. 1192)**

Datentypen

Variable	Datentyp	Funktion
s1, s2	ANY32	32-Bit-Quellregister
d	ARRAY [0..1] OF ANY32	unteres 32-Bit-Register des 64-Bit-Zielregisters

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

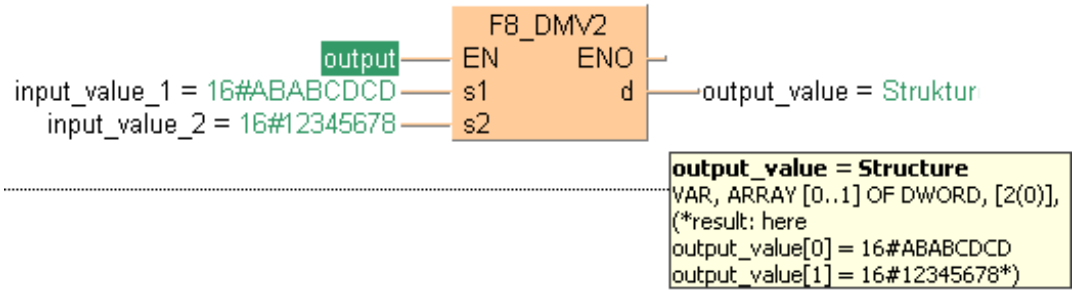
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value1	DWORD	16#ABABCDCD	
2	VAR	input_value2	DWORD	16#12345678	
3	VAR	output_value	ARRAY [0..1] OF DWORD	[2(0)]	result: here output_value[0] = 16#ABABCDCD output_value[1] = 16#12345678
4	VAR				

Hier wurden die Eingangsvariablen **input_value_1** und **input_value _2** deklariert. Statt dessen können Sie im Rumpf auch Konstanten direkt an den Eingang der Funktion schreiben.

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



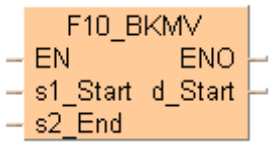
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F8_DMV2(input_value_1, input_value_2, output_value);
END_IF;
```

F10_BKMV

Block-Transfer

Erklärung Die Daten des Speicherbereichs mit der Anfangsadresse **s1** und der Endadresse **s2** werden in den Speicherbereich mit der Anfangsadresse **d** transferiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Die Operanden **s1** und **s2** sollten wie folgt sein:

- im gleichen Operanden
- **s1** ≤ **s2**

Wenn **s1**, **s2** und **d** im gleichen Datenbereich liegen:

- **s1** = **d**: Daten werden wieder in den gleichen Datenbereich kopiert.

source	15 . 12	11 . . 8	7 . 4	3 . . 0
[0]	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1
[1]	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 0
[2]	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1
[3]	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0
[4]	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 1

dest.	15 . 12	11 . . 8	7 . 4	3 . . 0
[0]	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 0
[1]	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1
[2]	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F10_BKMV (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Anfangsadresse im 16-Bit-Quellregister
s2		Endadresse im 16-Bit-Quellregister
d		Anfangsadresse im 16-Bit-Zielregister

Die Variablen **s1, s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

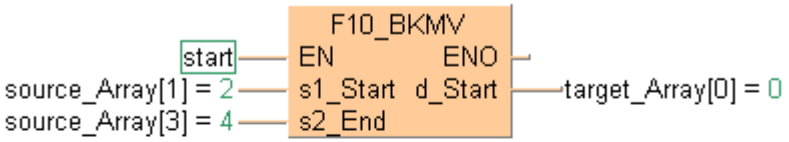
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	source_Array	ARRAY [0..4] OF INT	[1,2,3,4,5]	
2	VAR	target_Array	ARRAY [0..2] OF INT	[3(0)]	result after a 0->1 leading edge from start: [2,3,4]

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Daten des Speicherbereichs mit der Anfangsadresse **s1** und der Endadresse **s2** werden in den Speicherbereich mit der Anfangsadresse **d** transferiert.

KOP



F10_BKMV_LD	Struktur
start	2#0 an R262
+source_Array	Struktur
-target_Array	Struktur
[0]	0 an DT1216
[1]	0 an DT1217
[2]	0 an DT1218

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

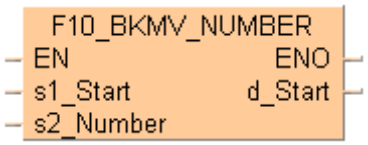
```
IF start THEN
    F10_BKMV( s1_Start:= source_Array[1],
        s2_End:= source_Array[3],
        d_Start=> target_Array[0]);
END_IF;
```

F10_BKMV_NUMBER_OFFSET

Block-Transfer einer Anzahl von Worten mit Offset

Erklärung

Wenn die Ausführungsbedingung (Trigger) gesetzt ist, dann werden die Daten des Speicherbereichs **s1_Start** und die Anzahl der festgelegten Worte mit **s2_Number** auf den Speicherbereich transferiert, der mit der Startadresse **d_Start** beginnt.



Dieser Befehl ist eine Modifikation des Befehls F10_BKMV (s. S. 775), der vom Compiler generiert wird.

☞ **Der Wert für 's2_Number' muss größer als 0 sein.**

Falls **s1_Start** und **d_Start** im selben Speicherbereich liegen, gilt:

- **s1_Start** = **d_Start**: die Daten werden wieder in den gleichen Datenbereich zurückkopiert.

SPS-Typen

Verfügbarkeit von F10_BKMV_NUMBER (s. S. 1186)

Datentypen

Variable	Datentyp	Funktion
s1_Start	ANY16	Anfangsadresse im 16-Bit-Quellregister
s2_Number		Anzahl der Worte, die transferiert werden (Quelle)
d_Start		Anfangsadresse im 16-Bit-Zielregister

Die Variablen **s1_Start, s2_Number** und **d_Start** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1_Start	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2_Number	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d_Start	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel

In diesem Beispiel wird die Funktion F10_BKMV_NUMBER im Kontaktplan (KOP) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

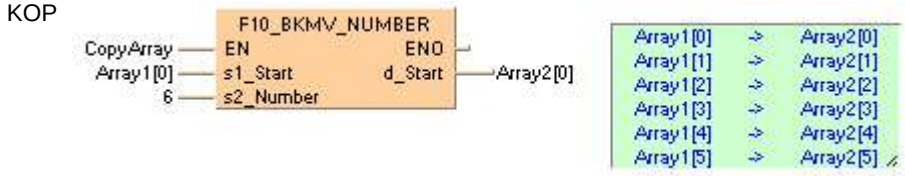
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Array1	ARRAY [0..5] OF INT	[6(0)]
1	VAR	Array2	ARRAY [0..5] OF INT	[6(0)]
2	VAR	CopyArray	BOOL	FALSE

Rumpf

Wenn die Variable **CopyArray** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie kopiert die Daten des Speicherbereichs **s1_Start** und die Anzahl der Worte festgelegt mit **s2_Number** auf den Speicherbereich, der mit der Startadresse **d_Start** beginnt.

KOP


F10_BKMV_NUMBER
CopyArray
EN
ENO
Array1[0]
s1_Start
d_Start
Array2[0]
6
s2_Number
Array1[0] -> Array2[0]
Array1[1] -> Array2[1]
Array1[2] -> Array2[2]
Array1[3] -> Array2[3]
Array1[4] -> Array2[4]
Array1[5] -> Array2[5]

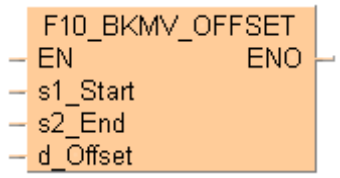
F10_BKMV_OFFSET

Block-Transfer mit Offset

Erklärung

Dieser Befehl ist eine Variante des Befehls F10_BKMV (s. S. 775), die vom Compiler erzeugt wird.

Wenn die Ausführungsbedingung (Trigger) gesetzt ist, dann werden die Daten des Speicherbereichs von **s1_Start** (16-Bit-Startadresse) bis **s2_End** (16-Bit-Endadresse) auf den Speicherbereich transferiert, der mit **d_Offset** definiert wird.



Falls **s1_Start** und **s2_End** im selben Speicherbereich liegen, gilt:

- **d_Offset** = 0: die Daten werden wieder in den gleichen Datenbereich zurückkopiert.

SPS-Typen **Verfügbarkeit von F10_BKMV_OFFSET (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
s1_Start	ANY16	Anfangsadresse im 16-Bit-Quellregister
s2_End		Endadresse im 16-Bit-Quellregister
d_Offset		Offset von s1_Start, Zielregister

Die Variablen **s1_Start**, **s2_End** und **d_Offset** müssen vom gleichen Datentyp sein.

Operanden

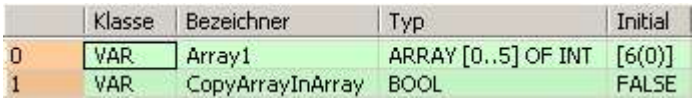
Für	Merker				T/C		Register			Konstante
s1_Start, s2_End	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
d_Offset	-	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Beispiel

In diesem Beispiel wird die Funktion F10_BKMV_OFFSET im Kontaktplan (KOP) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



	Klasse	Bezeichner	Typ	Initial
0	VAR	Array1	ARRAY [0..5] OF INT	[6(0)]
1	VAR	CopyArrayInArray	BOOL	FALSE

Rumpf

Wenn die Variable **CopyArrayInArray** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie kopiert die Daten des Speicherbereichs **s1_Start** (16-Bit-Startadresse) bis **s2_End** (16-Bit-Endadresse) auf den Speicherbereich, der mit **d_Offset** definiert wird.

KOP

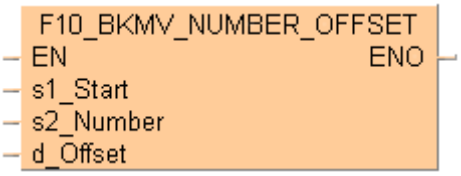
F10_BKMV_NUMBER_OFFSET

Block-Transfer einer Anzahl von Worten mit Offset

Erklärung

Dieser Befehl ist eine Variante des Befehls F10_BKMV (s. S. 775), die vom Compiler erzeugt wird.

Wenn die Ausführungsbedingung (Trigger) gesetzt ist, dann werden die Daten des Speicherbereichs von **s1_Start** (16-Bit-Startadresse) und die Anzahl der Worte, festgelegt mit **s2_Number** auf den Speicherbereich (ab 16-Bit-Startadresse **s1_Start**) transferiert, der mit **d_Offset** definiert wird.



- **Der Wert für 's2_Number' muss größer als 0 sein.**
- **Wenn d_Offset = 0: die Daten werden wieder in den gleichen Datenbereich zurückkopiert.**

SPS-Typen **Verfügbarkeit von F10_BKMV_NUMBER_OFFSET (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
s1_Start	ANY16	Anfangsadresse im 16-Bit-Quellregister
s2_Number		Anzahl der Worte, die transferiert werden (Quelle)
d_Offset		Anfangsadresse im 16-Bit-Zielregister

Die Variablen **s1_Start**, **s2_Number** und **d_Offset** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1_Start	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2_Number	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d_Offset	-	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Beispiel

In diesem Beispiel wird die Funktion F10_BKMV_NUMBER_OFFSET im Kontaktplan (KOP) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

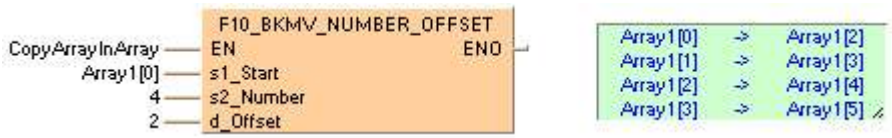
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Array1	ARRAY [0..5] OF INT	[6(0)]
1	VAR	CopyArrayInArray	BOOL	FALSE

Rumpf

Wenn die Variable **CopyArrayInArray** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie kopiert die Daten des Speicherbereichs **s1_Start** (16-Bit-Startadresse) und die Anzahl der Worte, festgelegt mit **s2_Number** auf den Speicherbereich (ab 16-Bit-Startadresse **s1_Start**), der mit **d_Offset** definiert wird.

KOP

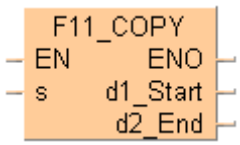

F10_BKMV_NUMBER_OFFSET
CopyArrayInArray
EN
ENO
Array1[0]
s1_Start
4
s2_Number
2
d_Offset
Array1[0] -> Array1[2]
Array1[1] -> Array1[3]
Array1[2] -> Array1[4]
Array1[3] -> Array1[5]

F11_COPY

Block setzen

Erklärung

Der 16-Bit-Wert des Speicherregisters **s** oder die Konstante **s** wird wiederholt in den Speicherbereich mit der Anfangsadresse **d1** und der Endadresse **d2** transferiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Die Operanden **d1** und **d2** sollten wie folgt sein:

- im gleichen Operanden
- Es gilt **d1** ≤ **d2**

source	15 . 12	11 . . 8	7 . . 4	3 . . 0
	0 0 0 0	0 0 0 0	0 0 0 0	1 0 1 1

dest.	15 . 12	11 . . 8	7 . . 4	3 . . 0
[0]	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1
[1]	0 0 0 0	0 0 0 0	0 0 0 0	0 0 1 1
[2]	0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 1
[3]	0 0 0 0	0 0 0 0	0 0 0 0	1 0 1 1
[4]	0 0 0 0	0 0 0 0	0 0 0 0	1 0 1 1
[5]	0 0 0 0	0 0 0 0	0 0 0 0	1 0 1 1

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F11_COPY (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	16-Bit-Quellregister
d1		Anfangsadresse im 16-Bit-Zielregister
d2		Endadresse im 16-Bit-Zieldatenregister

Die Variablen **s, d1** und **d2** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-

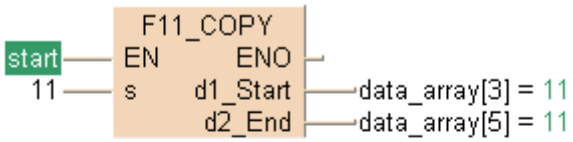
Beispiel In diesem Beispiel wird die Funktion F11_COPY im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_array	ARRAY [0..6] OF INT	[1,3,5,7,9,11,13]	result after a 0->1 leading edge from start: [1,3,5,11,11,11,13]
2	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



-F11_COPY_LD	Struktur
start	2#0 an R250
-data_array	Struktur
[0]	1 an DT1202
[1]	3 an DT1203
[2]	5 an DT1204
[3]	7 an DT1205
[4]	9 an DT1206
[5]	11 an DT1207
[6]	13 an DT1208

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

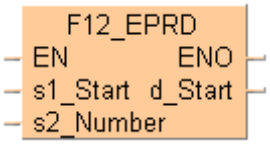
```
IF start THEN
    (* Copy the value 11 to data_array[3], *)
    (* data_array[4] and data_array[5] *)
    F11_COPY( s:= 11,
        d1_Start=> data_array[3],
        d2_End=> data_array[5]);
END_IF;
```

F12_EPRD

EEPROMSpeicher lesen

Erklärung

Mit diesem Befehl werden Daten aus dem EEPROM/ Flash-ROM in den Datenspeicher (DT) kopiert. Diese Kopierfunktion arbeitet nur blockweise. Damit ist das Kopieren einzelner Worte nicht möglich. Die Blockgröße und Blockmenge ist in der Tabelle aufgeführt. Achten Sie darauf, dass Sie für den Zielspeicher mindestens 64 bzw. 2048 freie Datenregister (1 Block = 64 bzw. 2048 Wörter (DT)) reservieren.



SPS-Typen **Verfügbarkeit von F12_EPRD (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
EN	BOOL	Aktivierung der Funktion (wenn EN den Status TRUE hat, wird die Funktion bei jedem Zyklus ausgeführt)
s1	ANY32	EEPROM Startblock-Nummer
s2		Zahl der auszulesenden Blöcke (1 Block = 64 Wörter bzw. 2048 Wörter (DT)
d	ANY16	DT-Anfangsadresse für die Datenspeicherung
ENO	BOOL	Wenn die Funktion ausgeführt wurde, wird ENO auf TRUE gesetzt. Das ist nützlich, um mehrere Funktionen mit EN-Funktionalität zu kaskadieren.

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2	DWX	DWY	DWR	-	DSV	DEV	DDT	-	-	dez., hex.
d	-	-	-	-	-	-	DT	-	-	-

SPS-spezifische Daten

SPS-Typ	FP0 2,7k C10/C14/C16 und FP-e	FP0 5k C32	FP0 10k T32CP	FP-Sigma, FP-X, FP0R
Zusatzspeicher	EEPROM	EEPROM	EEPROM	Flash-ROM
Blockgröße (1 Block)	64 Wörter (64x16 Bit)	64 Wörter (64x16 Bit)	64 Wörter (64x16 Bit)	2048 Worte
EEPROM Startblock-Nummer	0 bis 9	0 bis 95	0 bis 255	0 bis 15
Anzahl der zu lesenden / schreibenden Blöcke pro Ausführung	1 bis 2	1 bis 8	1 bis 255	1 (schreiben) 1 to 16 (lesen)
Schreibdauer (zusätzliche Zykluszeit)	< 20 ms pro Block	< 5 ms pro Block	< 5 ms pro Block	< 100ms pro Block
Lesedauer (zusätzliche Zykluszeit)	Weniger als 1 ms pro Block	Weniger als 1 ms pro Block	Weniger als 1 ms pro Block	9.94µs + (1562.6*Anzahl der Blöcke) µs
Max. Anzahl an Schreibvorgängen ☞ **Stromausfall und Wechsel von RUN-> in PROG-Modus werden mitgezählt.**	100,000	10,000	10,000	10,000
Max Lesedauer	Keine Obergrenze	Keine Obergrenze	Keine Obergrenze	Keine Obergrenze

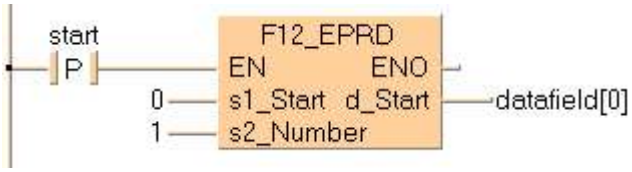
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the instruction
1	VAR	datafield	ARRAY [0..63] OF INT	[64(0)]	data field to be uploaded data from EEPROM

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie liest einen Block (= 64 Wörter) ab Startblock Nummer 0 aus dem EEPROM und schreibt die Information in das Datenfeld von Datenfeld[0] bis Datenfeld[63].

KOP

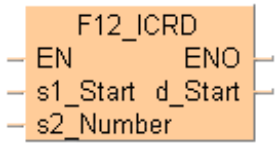


F12_ICRD

Erw. Speicherbereich auf IC-Karte lesen

Erklärung

Die Anzahl der Wörter, die in **s2** spezifiziert worden sind, werden im erweiterten Speicherbereich der IC-Karte, der in **s1** spezifiziert wurde, gelesen und in den Bereich **d** geschrieben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F12_ICRD (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	Anfangsadresse im 32-Bit-Bereich, ab der im erweiterten Speicherbereich gelesen werden soll
s2		Anzahl der Wörter, die gelesen werden (32 Bit)
d	ANY16	Anfangsadresse im 16-Bit-Zieldatenregister

Operanden

Für	Merker				T/C		Register			Konstante
s1	-	-	-	-	-	-	-	-	-	dez., hex.
s2	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel

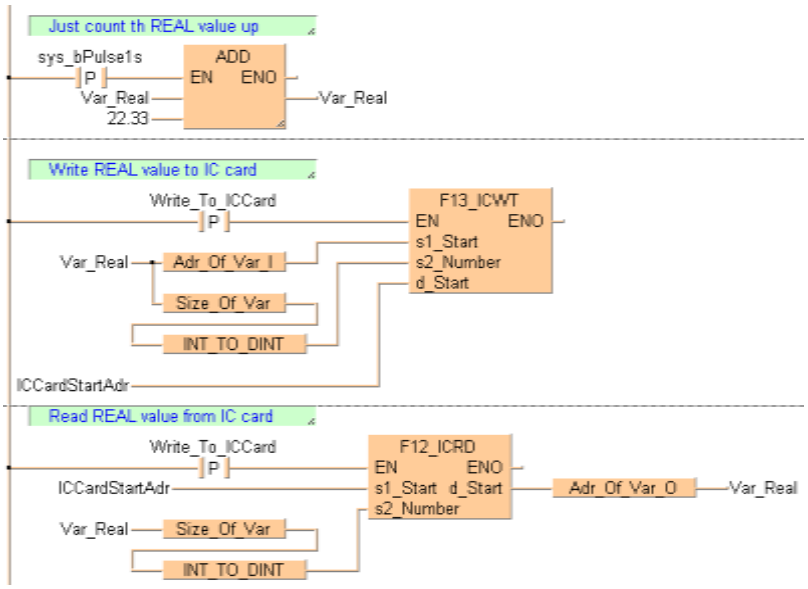
In diesem Beispiel wird die Funktion F12_ICRD und F13_ICWT im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Var_Real	REAL	0.0
1	VAR	Write_To_ICCard	BOOL	FALSE
2	VAR	Read_From_ICCard	BOOL	FALSE
3	VAR	ICCardStartAdr	DINT	0

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

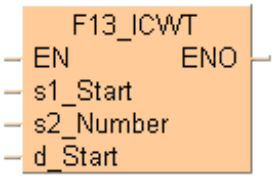
```
IF DF(R901C) THEN
    Var_Real := Var_Real + 22.33;
END_IF;

(* Write a REAL value to the IC Card *)
IF DF(Write_To_ICCard) THEN
    F13_ICWT( s1_Start:= Adr_Of_Var( Var_Real ) , s2_Number:= INT_TO_DINT(
Size_Of_Var( Var_Real ) ), d_Start:= ICCardStartAdr );
END_IF;

(*Read a REAL value from the IC Card*)
IF DF(Read_From_ICCard) THEN
    F12_ICRD( s1_Start:= ICCardStartAdr, s2_Number:= INT_TO_DINT(
Size_Of_Var( Var_Real ) ) ,
        d_Start=> Adr_Of_Var( Var_Real ) );
END_IF;
```

F13_ICWT

Erw. Speicherbereich auf IC-Karte beschreiben

Erklärung

Die Anzahl der Wörter, die in **s2_Number** spezifiziert worden sind, werden von dem Bereich, der in **s1_Start** spezifiziert wurde, gelesen und in den erweiterten Speicherbereich der IC-Karte, der in **d_Start** spezifiziert wurde, geschrieben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F13_ICWT (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s1_Start	ANY16	Quelldaten, 16-Bit-Anfangsadresse
s2_Number	ANY32	Anzahl der zu lesenden Worte, die dann auf die IC-Karte geschrieben werden
d_Start		Anfangsadresse, ab der auf die IC-Karte geschrieben wird

Operanden

Für	Merker				T/C		Register			Konstante
s1_Start	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2_Number	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d_Start	-	-	-	-	-	-	-	-	-	dez., hex.

Beispiel

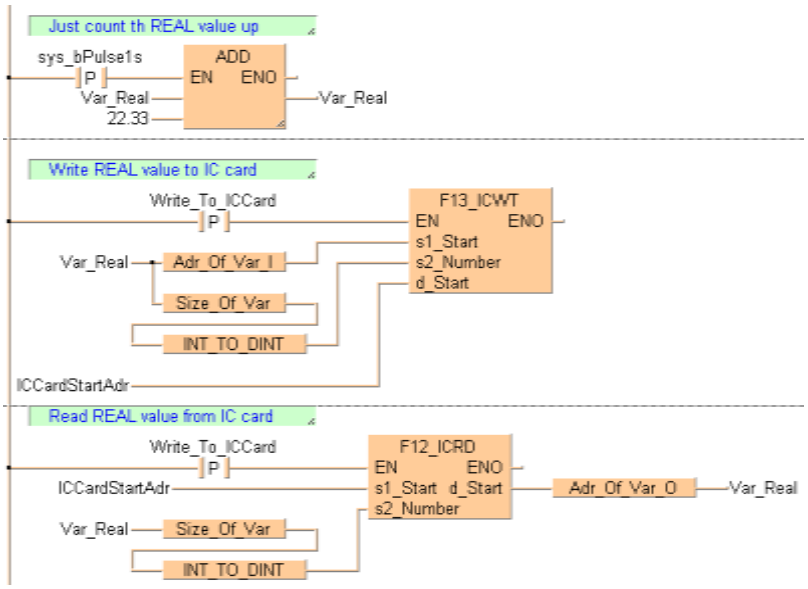
In diesem Beispiel wird die Funktion F12_ICRD und F13_ICWT im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Var_Real	REAL	0.0
1	VAR	Write_To_ICCard	BOOL	FALSE
2	VAR	Read_From_ICCard	BOOL	FALSE
3	VAR	ICCardStartAdr	DINT	0

KOP

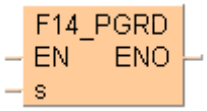


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(R901C) THEN
    Var_Real := Var_Real + 22.33;
END_IF;

(* Write a REAL value to the IC Card *)
IF DF(Write_To_ICCard) THEN
    F13_ICWT( s1_Start:= Adr_Of_Var( Var_Real ) , s2_Number:= INT_TO_DINT(
Size_Of_Var( Var_Real ) ), d_Start:= ICCardStartAdr );
END_IF;

(*Read a REAL value from the IC Card*)
IF DF(Read_From_ICCard) THEN
    F12_ICRD( s1_Start:= ICCardStartAdr, s2_Number:= INT_TO_DINT(
Size_Of_Var( Var_Real ) ) ,
        d_Start=> Adr_Of_Var( Var_Real ) );
END_IF;
```

F14_PGRD Programm lesen

Erklärung Wenn das Ausführungsereignis von **F/P14_PGRD** auftritt, wird das derzeitige Programm bis zum Befehl END abgearbeitet. Anschließend wird das Programm, das durch **s** spezifiziert wurde abgearbeitet.



s: Anfangsadresse des Programmspeichers

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F14_PGRD (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s	ARRAY **[0..5] of** WORD	Anfangsadresse des Programmspeichers

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	-

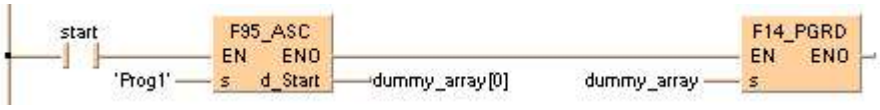
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	dummy_array	ARRAY [0..5] OF WORD	[6(0)]	contains the file name in HEX_ASCII format
2	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Die Funktion liest den Dateinamen Prog1 von der IC-Karte und führt die Datei aus.

KOP



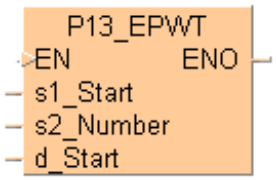
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F95_ASC( s:= 'Prog1',
        d_Start=> dummy_array[0]);
    F14_PGRD( dummy_array );
END_IF;
```

P13_EPWT

EEPROM-Speicher beschreiben

Erklärung Mit diesem Befehl wird ein Bereich aus dem Datenspeicher (DT) in das EEPROM/ Flash-ROM kopiert.



Der EEPROM-Speicher ist nicht mit dem selbsthaltenden Speicherbereich zu verwechseln. Der selbsthaltende Speicherbereich sichert Daten in Echtzeit. Bei Stromausfall werden die Daten aus dem selbsthaltenden Speicherbereich im EEPROM gespeichert. Der Befehl P13_EPWT sendet nur dann Daten an das EEPROM, wenn er ausgeführt wird. Dieser Befehl kann nur begrenzt oft Daten in das EEPROM schreiben (siehe Tabelle unten). Stellen Sie sicher, dass der Befehl P13_EPWT nicht öfter ausgeführt wird, als die angegebene Zahl von Speichervorgängen zulässt.

Wenn Sie zum Beispiel den Befehl P13_EPWT mit einem R901A-Relais (Impulsdauer 0,1 s) ausführen, wird das EEPROM nach 100.000 * 0,1 s = 10.000 s (2,8 Stunden) funktionsunfähig. Wenn Sie jedoch Profildaten wie Positionierungsparameter oder andere Parameterwerte, die selten geändert werden, speichern möchten, ist dieser Befehl sehr nützlich.

SPS-Typen **Verfügbarkeit von P13_EPWT (s. S. 1194)**

☞ **Einer der beiden Eingangsvariablen 's2' oder 'd' muss ein konstanter Zahlenwert sein.**

Datentypen

Variable	Datentyp	Funktion
EN	BOOL	Aktivierung der Funktion (wenn EN den Status TRUE hat, wird die Funktion bei jedem Zyklus ausgeführt)
s1	INT, WORD	DT-Startadresse des/der zu speichernden Blöcke
s2	DINT, DWORD	Zahl der zu schreibenden Blöcke (1 Block = 64 Wörter bzw. 2048 Wörter (DT)
d	DINT, DWORD	EEPROM Startblock-Nummer
ENO	BOOL	Wenn die Funktion ausgeführt wurde, wird ENO auf TRUE gesetzt. Das ist nützlich, um mehrere Funktionen mit EN-Funktionalität zu kaskadieren.

Operanden

Für	Merker				T/C		Register			Konstante
s1	-	-	-	-	-	-	DT	-	-	-
s2, d	DWX	DWY	DWR	-	DSV	DEV	DDT	-	-	dez., hex.

■ SPS-spezifische Daten

SPS-Typ	FP0 2,7k C10/C14/C16 und FP-e	FP0 5k C32	FP0 10k T32CP	FP-Sigma, FP-X, FP0R
Zusatzspeicher	EEPROM	EEPROM	EEPROM	Flash-ROM
Blockgröße (1 Block)	64 Wörter (64x16 Bit)	64 Wörter (64x16 Bit)	64 Wörter (64x16 Bit)	2048 Worte
EEPROM Startblock-Nummer	0 bis 9	0 bis 95	0 bis 255	0 bis 15
Anzahl der zu lesenden / schreibenden Blöcke pro Ausführung	1 bis 2	1 bis 8	1 bis 255	1 (schreiben) 1 to 16 (lesen)
Schreibdauer (zusätzliche Zykluszeit)	< 20 ms pro Block	< 5 ms pro Block	< 5 ms pro Block	< 100ms pro Block
Lesedauer (zusätzliche Zykluszeit)	Weniger als 1 ms pro Block	Weniger als 1 ms pro Block	Weniger als 1 ms pro Block	9.94µs + (1562.6*Anzahl der Blöcke) µs
Max. Anzahl an Schreibvorgängen ☞ **Stromausfall und Wechsel von RUN-> in PROG-Modus werden mitgezählt.**	100,000	10,000	10,000	10,000
Max Lesedauer	Keine Obergrenze	Keine Obergrenze	Keine Obergrenze	Keine Obergrenze

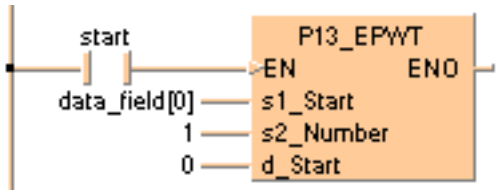
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	DataField	ARRAY (0..63] OF INT	(1,2,3,4,5,6,7,8,9,10,11,12,52(0)]	data field to be uploaded data from EEPROM

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie liest den Inhalt von Datenfeld[0] bis Datenfeld[63] (s2* = 1 => 1 Block = 64 Wörter) und schreibt die Information ab Startblock Nummer 0 in das EEPROM.

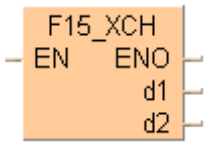
KOP



F15_XCH

16-Bit-Austausch

Erklärung Der 16-Bit-Wert des Speicherregisters **d1** wird in das Speicherregister **d2** transferiert. Gleichzeitig wird der 16-Bit-Wert des Speicherregisters **d2** in das Speicherregister **d1** transferiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein.



Bit	15 . . 12	11 . . 8	7 . . 4	3 . . 0
d1	0 0 0 0	0 0 0 0	0 0 0 1	0 0 0 1
d2	0 0 0 0	0 0 0 0	0 0 0 1	1 0 0 0

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F15_XCH (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
d1	ANY16	16-Bit-Bereich, der mit **d2** ausgetauscht wird
d2		16-Bit-Bereich, der mit **d1** ausgetauscht wird

Die Variablen **d1** und **d2** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
d1, d2	-	WY	WR	WL	SV	EV	DT	LD	FL	-

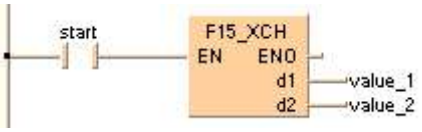
Beispiel In diesem Beispiel wird die Funktion F15_XCH im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value_1	INT	17	result after a 0->1 leading
2	VAR	value_2	INT	24	result after a 0->1 leading
3	VAR				edge from start: 17

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

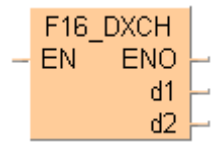
```
IF start THEN
    F15_XCH(value_1, value_2);
END_IF;
```

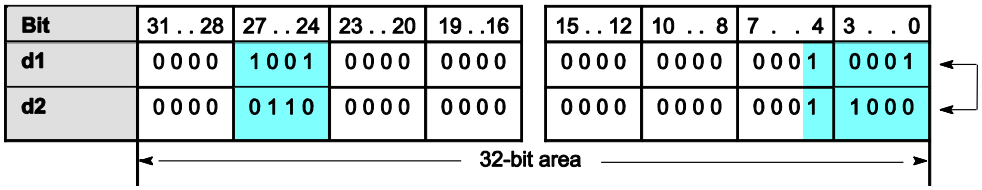
F16_DXCH

32-Bit-Austausch

Erklärung

Der 32-Bit-Wert des Speicherregisters **d1** wird in das Speicherregister **d2** transferiert. Gleichzeitig wird der 16-Bit-Wert des Speicherregisters **d2** in das Speicherregister **d1** transferiert. Dazu muss der Trigger **EN** auf EIN gesetzt sein.





Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F16_DXCH (s. S. 1187)

Datentypen

Variable	Datentyp	Funktion
d1	ANY32	32-Bit-Bereich, der mit **d2** ausgetauscht wird
d2		32-Bit-Bereich, der mit **d1** ausgetauscht wird

Die Variablen **d1** und **d2** müssen vom gleichen Datentyp sein.

Operanden

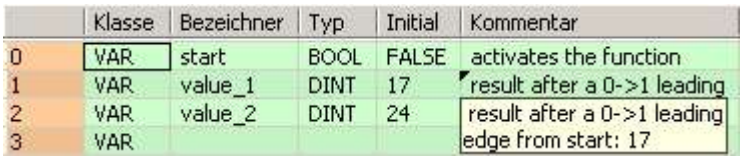
Für	Merker				T/C		Register			Konstante
d1, d2	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	value_1	DINT	17	result after a 0->1 leading
2	VAR	value_2	DINT	24	result after a 0->1 leading edge from start: 17
3	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

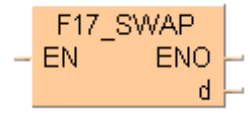
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F16_DXCH(value_1, value_2);
END_IF;
```

F17_SWAP

Byte-Austausch

Erklärung Die Bytes des Speicherregisters **d** werden gegeneinander ausgetauscht. Dazu muss der Trigger **EN** auf EIN gesetzt sein. 1 Byte entspricht 8 Bit.



Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
DT770	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1
16#	2	3	4	5

higher byte (8-bit) lower byte (8-bit)

Bit	15 . . 12	10 . . 8	7 . . 4	3 . . 0
DT770	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1
16#	4	5	2	3

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F17_SWAP (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
d	ANY16	16-Bit-Bereich in dem die höherwertigen und die niederwertigen Bytes ausgetauscht werden

Operanden

Für	Merker				T/C		Register			Konstante
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

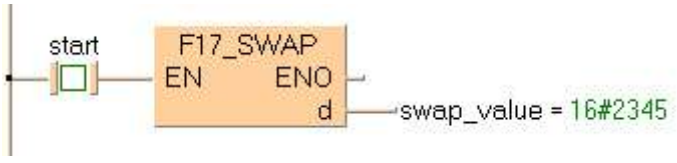
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	swap_value	WORD	16#2345	result after 0->1 leading
2	VAR				edge from start: 16#4523

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

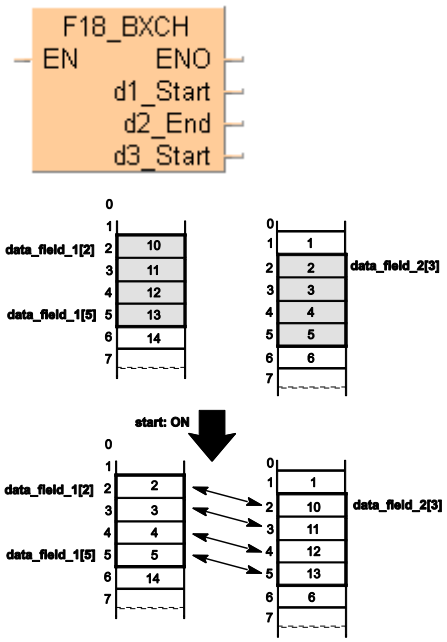
```
IF start THEN
    F17_SWAP(swap_value);
END_IF;
```

F18_BXCH

16-Bit-Blockaustausch

Erklärung

Die Funktion tauscht einen 16-Bit-Datenblock mit einem anderen aus. Der Anfang des ersten Datenblocks wird am Ausgang **d1** und das Ende am Ausgang **d2** ausgegeben. Der Ausgang **d3** gibt den Anfang des zweiten Datenblocks aus.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F18_BXCH (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
d1	ANY16	Anfangsadresse des 16-Bit-Bereichs des Datenblocks 1
d2		Zieladresse des 16-Bit-Bereichs des Datenblocks 1
d3		Anfangsadresse des 16-Bit-Bereichs des Datenblocks 2

Operanden

Für	Merker				T/C		Register			Konstante
d1, d2, d3	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermer-ker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Adresse der Variablen an den Ausgängen von **d1** > **d2** ist ▪ der auszutauschende Datenblock größer ist als der Zielbereich ▪
R9008	%MX0.900.8	kurzzeitig	

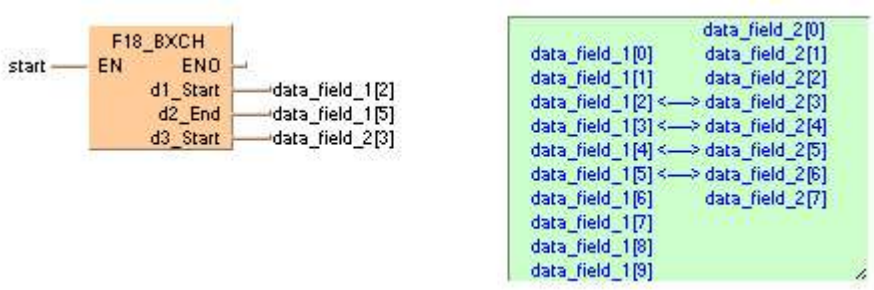
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	data_field_1	ARRAY [0..9] OF INT	[8,9,10,11,12,13,14,15,16,17]	Arbitrarily large data field
2	VAR	data_field_2	ARRAY[0..7] OF INT	[-1,0,1,2,3,4,5,6]	Arbitrarily large data field

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Sie tauscht die Daten des ARRAY **data_field_1** (vom 2. bis zum 5. Element) mit den Daten des ARRAY **data_field_2** (ab dem 3. Element) aus.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F18_BXCH(
        d1_Start=> d1[2], d2_End=> d1[4], d3_Start=> d2[1]);
END_IF;
```

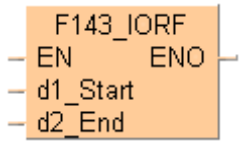
F143_IORF

Partial I/O update

Erklärung

Mit dem Wert von **d1_Start** wird die Anfangsadresse und mit dem Wert von **d2_End** wird die Endadresse des E/A-Bereichs festgelegt, die aktualisiert werden soll, sobald der Trigger EN auf TRUE gesetzt ist.

Die partielle E/A-Aktualisierung (IORF) ist nur für E/A-Module der Basis- und Erweiterungseinheit anwendbar. Bei dezentral betriebenen E/A-Modulen kann die partielle E/A-Aktualisierung nicht angewendet werden.



Nur für SPS mit konfigurierbaren, fortlaufenden E/A-Adressen: FP2, FP2SH, FP3 /5 /10 /10SH (SPS mit Baugruppenträger)

Wenn Sie diesen Befehl verwenden, können Sie die Ein- und Ausgänge ohne Zeitverzögerung beim Abfragen aktualisieren.

Es sollte für **d1_Start** und **d2_end** der gleiche Operandentyp festgelegt werden.

- Definieren Sie die Wortadresse als 0 ≤ **d1** ≤ **d2** ≤ 127.
  Wenn nur WX10 bzw. WY10 entsprechend der Master-E/A-Adressenkonfiguration aktualisiert werden soll, werden **d1** und **d2** folgendermaßen gesetzt: **d1** = 10 und **d2** = 10.
- Legen Sie zum Aktualisieren von nur einem Wort die gleiche Wortadresse in **d1** und **d2** fest.

☞

- **Mit der FP-Sigma und der FP0 erfolgt die mit dem IORF-Befehl ausgelöste E/A-Aktualisierung nur am CPU-Modul. Wenn 'd1 und d2' keine Konstanten sind, greift der Compiler automatisch über das Indexregister auf die Argumente zu.**
- **Die Variablen d1 und d2 müssen vom gleichen Typ sein.**
- **Für die Aktualisierung der Eingänge sollte für d1 und d2 die Adresse WX0 zugewiesen werden.**
- **Für die Aktualisierung der Ausgänge sollte für d1 und d2 die Adresse WY0 zugewiesen werden.**

SPS-Typen

Verfügbarkeit von F143_IORF (s. S. 1187)

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Datentypen

Variable	Datentyp	Funktion
d1	ANY16	Anfangsadresse
d2		Endadresse

Operanden

Für	Merker				T/C		Register			Konstante
d1	WX	WY	-	WL	SV	EV	DT	-	FL	dez., hex.
d2	WX	WY	-	WL	SV	EV	DT	-	FL	dez., hex.

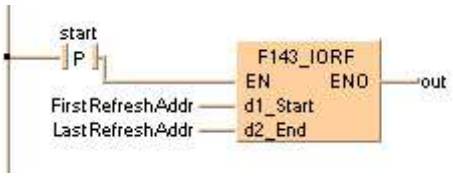
Beispiel In diesem Beispiel wird die Funktion F143_IORF im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the funtion
1	VAR	FirstRefreshAddr	INT	10	
2	VAR	LastRefreshAddr	INT	10	

Rumpf Wenn die Variable **start** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt. Wenn nur WX10 bzw. WY10 entsprechend der Master-E/A-Adressen-Konfiguration aktualisiert werden soll, wird Folgendes gesetzt: **d1** = 10 and **d2** = 10.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
(* PLCs with backplanes  FP-C/FP2/FP2SH/FP3/FP10SH *)
IF start THEN
    (* Updates the input/output relay of word no. 0 to 1 *)
    F143_IORF( 0, 1);
END_IF;
(* PLCs without backplanes FP0, FP-Sigma *)
IF start THEN
    (* Updates the input/output relay of word no. 0 to 1 *)
    F143_IORF(WX0, WX1);
    F143_IORF(WY0, WY1);
END_IF;
```

F147_PR

Drucken von ASCII-Zeichen

Erklärung

Es werden insgesamt 12 Zeichen im ASCII-Code ausgegeben. Die Zeichen sind in 6 Datenworten ab der Adresse **s** gespeichert. Die Ausgabe erfolgt über das Ausgangswort **d**, wenn der Trigger **EN** gesetzt ist. Wenn ein Drucker an Ausgang **d** angeschlossen ist, wird ein dem ASCII-Code entsprechendes Zeichen ausgedruckt.

Nur die Bits 0 bis 8 in **d** werden für den aktuellen Ausdruck verwendet. Der ASCII-Code wird beginnend mit dem niedrigstwertigen Byte des Anfangsbereichs ausgegeben. Es werden drei Zyklen für die Übertragung eines Zeichens benötigt. Es sind daher 37 Zyklen für die Übertragung aller 12 möglichen Zeichen erforderlich.

Da es nicht möglich ist, den Befehl **F147_PR** innerhalb eines Zyklus mehrfach auszuführen, verwenden Sie den Sondermerker R9033, damit kein weiterer PR-Befehl ausgeführt wird. Wenn die Zeichen in den ASCII-Code konvertiert werden, wird die Verwendung des Befehls F95_ASC (s. S. 640) empfohlen.

SPS-Typen **Verfügbarkeit von F147_PR (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	16-Bit-Speicherbereich (Startadresse) um 12 Byte (6 Worte) zu speichern
d	WORD	Ausgabe des ASCII-Codes

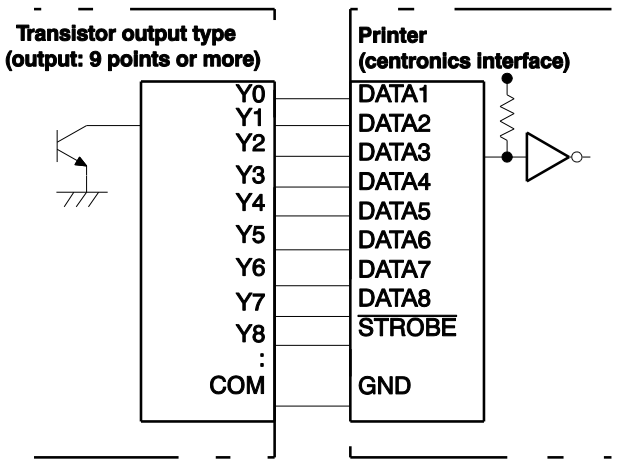
Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
d	-	WY	-	-	-	-	-	-	-	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Endadresse für die Speicherung des ASCII-Codes überschreitet die Grenzen ▪ der Trigger eines anderen F147_PR-Befehls gesetzt wird, während ein F147_PR-Befehl ausgeführt wird
R9008	%MX0.900.8	kurzzeitig	
R9033	%MX0.903.3	permanent	▪ ein F147_PR-Befehl ausgeführt wird

■ Kommunikationsbeispiel



Beispiel In diesem Beispiel wird die Funktion F147_PR im Kontaktplan (KOP) programmiert.
Die in der Folge **PrintOutString** gespeicherten ASCII-Codes werden über den externen Merker WY0 ausgegeben, wenn der Trigger **Start** gesetzt ist.

Source: ASCII code for 12 character A, B, C, D, E, F, G, H, I and J

PrintOutString											
ASCII HEX code	0D	0A	4A	49	48	47	46	45	44	43	42	41
ASCII character	C_R	LF	J	I	H	G	F	E	D	C	B	A

Control data for printer — ASCII codes

start: ON

Destination

	YF	YE	YD	YC	YB	YA	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
WY0																

Y0 to YF: for data signals of printer
(Y0 to Y7 correspond to DA A1 to DA A8 of printer
Y8: for strobe signal of printer
Y9 to YF: not used

GVL In der Globalen Variablen Liste können Sie Variablen festlegen, die von allen POEs des Projektes verwendet werden können.

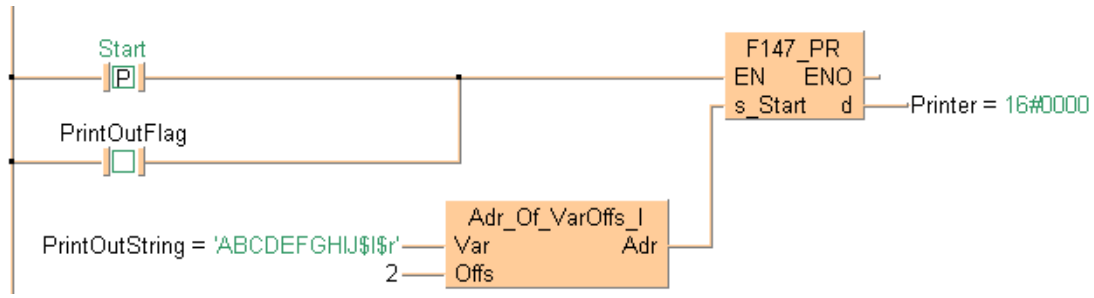
Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	Printer	WY1	%QW1	WORD	0
1	VAR_GLOBAL	PrintOutFlag	R9033	%MX0.903.3	BOOL	FALSE

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Start	BOOL	FALSE	
1	VAR_EXTERNAL	PrintOutFlag	BOOL	FALSE	
2	VAR	PrintOutString	STRING[12]	'ABCDEFGHIJLR'	$L = line feed $R = carriage return
3	VAR_EXTERNAL	Printer	WORD	0	

Rumpf KOP



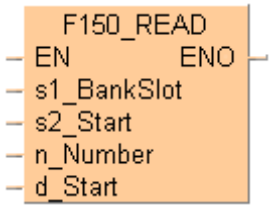
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) OR PrintOutFlag THEN
    F147_PR( Adr_Of_VarOffs( PrintOutString, 2), Printer);
END_IF;
```

F150_READ

Lesen vom Spezialmodul

Erklärung Daten werden vom Speicher eines Spezialmoduls gelesen.



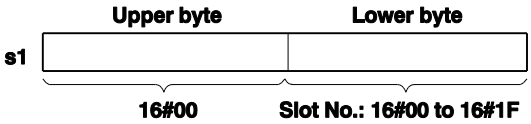
Die **n** Worte ab Anfangsadresse **s2** im Spezialmodul, das durch **s1** festgelegt ist, werden gelesen und in der CPU im Speicherbereich mit der Anfangsadresse **d** gespeichert.

Die Anzahl der Variablenargmenten an den Eingängen hängt von den verfügbaren Indexregistern der SPS ab.

s1 spezifizieren

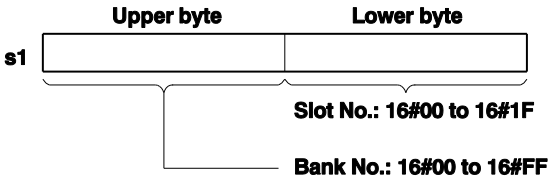
Intelligentes Modul ohne Speicherbank

Legen Sie der Stechkplatznummer des intelligenten Ziel-Moduls fest.



Intelligentes Modul mit Speicherbank

Legen Sie die Steckplatznummer und die Speichbanknummer (hex. Konstante) des intelligenten Ziel-Moduls fest.



Hinweis: Intelligentes Modul mit Speicherbank

Name	Bestellnummer
FP3 Datenspeichermodul	AFP32091 AFP32092
FPΣ Datenspeichermodul	AFPG201

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F150_READ (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Legt Speicherbanknr. und Steckplatznr. im Speicher des intelligenten Moduls fest
s2		Legt Anfangsadresse im Speicher des intelligenten Moduls fest (Quelle)
n	INT	Legt die Anzahl der Worte fest, die gelesen werden
d	ANY16	Anfangsadresse in der CPU, ab der Daten gespeichert werden (Ziel)

Operanden

Für	Merker				T/C		Register			Konstante
s1	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
s2	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ s1 die Grenze des festgelegten Bereichs überschreitet ▪ die gelesenen Daten den Bereich von d überschreitet
R9008	%MX0.900.8	kurzzeitig	

Beispiel

In diesem Beispiel wird die Funktion F150_READ im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

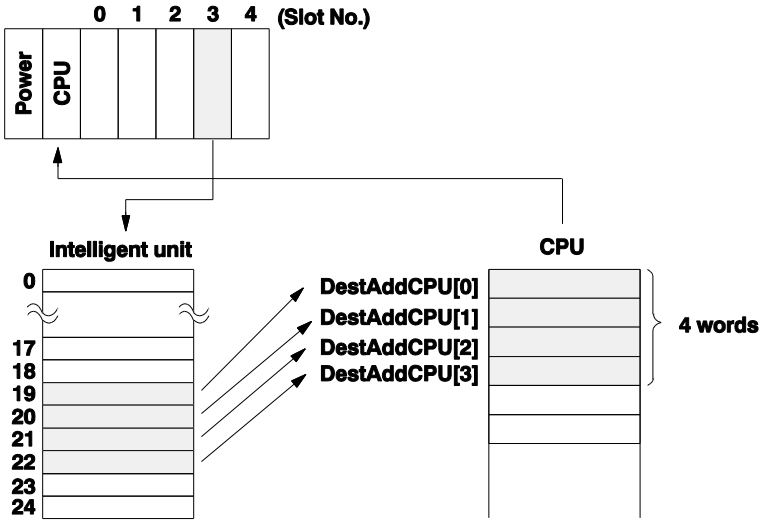
POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

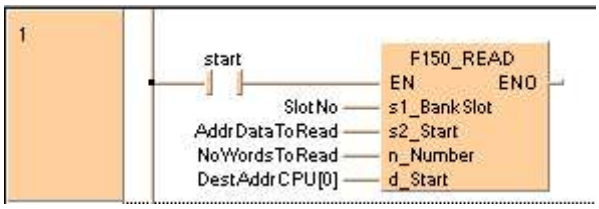
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Start	BOOL	FALSE	activates the function
1	VAR	Slot No	WORD	16#03	if start is TRUE, this value
2	VAR	Addr Data To Read	INT	19	Starting address in intelligent
3	VAR	No Words To Read	INT	4	
4	VAR	Dest Addr CPU	ARRAY [0..3] OF INT	[4(0)]	Starting address in CPU to store data read
5	VAR				

Rumpf

4 Worte werden ab Adresse 19, die in **AddrDataToRead** festgelegt ist, aus dem Speicherbereich des intelligenten Moduls (im Steckplatz 3) gelesen. Dann werden sie im Array **DestAddrCPU** gespeichert, wenn Freigabe auf TRUE gesetzt ist.



KOP



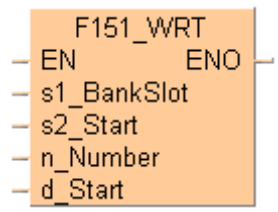
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F150_READ( s1_BankSlot:= SlotNo,
        s2_Start:= AddrDataToRead,
        n_Number:= NoWordsToRead,
        d_Start:= DestAddrCPU[0]);
END_IF;
```

F151_WRT

Schreiben zum Spezialmodul

Erklärung Daten werden in den Speicher eines intelligenten Moduls geschrieben.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

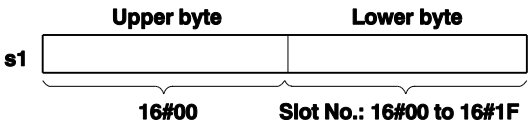
n Worte werden im CPU gelesen und in den Speicher des intelligenten Moduls geschrieben. Mit **s2** legen Sie die Anfangsadresse im Speicherbereich der CPU fest. Mit **d** legen Sie die Anfangsadresse im Speicherbereich des intelligenten Moduls fest, und mit **s1** legen Sie die Speicherbanknummer und die Steckplatznummer des intelligenten Moduls fest.

Die Anzahl der Variablenargumenten an den Eingängen hängt von den verfügbaren Indexregistern der SPS ab.

s1 spezifizieren

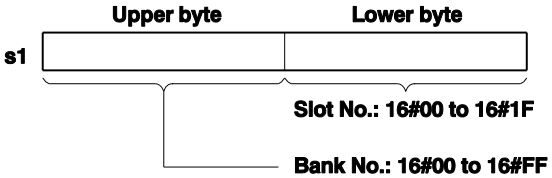
Intelligentes Modul ohne Speicherbank

Legen Sie der Steckplatznummer des Ziel-Intelligent-Moduls fest.



Intelligentes Modul mit Speicherbank

Legen Sie die Steckplatznummer und die Speicherbanknummer (hex. Konstante) des Ziel-Intelligent-Moduls fest.



Hinweis: Intelligentes Modul mit Speicherbank

Name	Bestellnummer
FP3 Datenspeichermodul	AFP32091 AFP32092

SPS-Typen **Verfügbarkeit von F151_WRT (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Legt Speicherbanknr. und Steckplatznr. im Speicher des intelligenten Moduls fest
s2		Anfangsadresse der Daten im Speicherbereich der CPU
n	INT	Legt die Anzahl der Worte fest, die in den Speicher des intelligenten Moduls geschrieben werden
d	ANY16	Legt die Anfangsadresse im Speicherbereich des intelligenten Moduls fest, ab der die Daten geschrieben werden (Zieladresse)

Operanden

Für	Merker				T/C		Register			Konstante
s1	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
s2	-	WY	WR	WL	SV	EV	DT	LD	FL	-
n	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

Fehlermerker

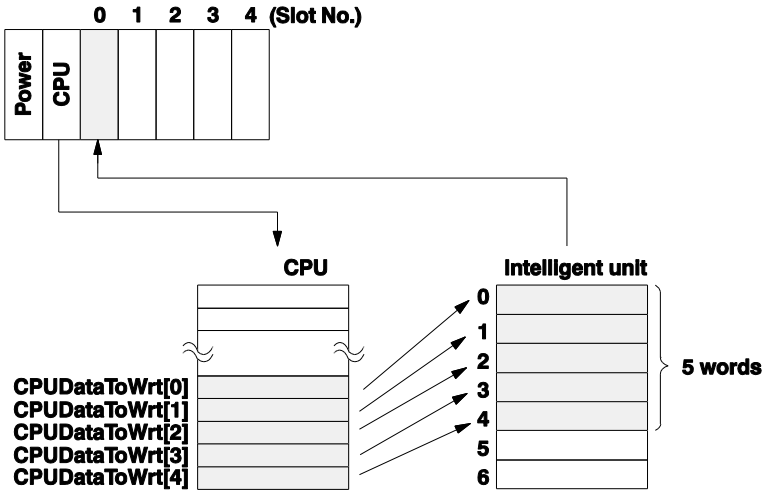
Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ s1 die Grenze des festgelegten Bereichs überschreitet ▪ die gelesenen Daten den Bereich von d überschreitet
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird die Funktion F151_WRT im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

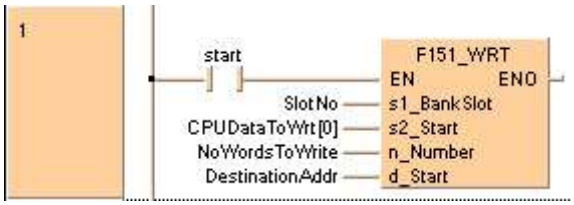
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Start	BOOL	FALSE	activates the function
1	VAR	Slot No	WORD	16#00	if start is TRUE, this value
2	VAR	CPU Data To Wrt	ARRAY [0..4] OF INT	[5,10,15,20,25]	
3	VAR	No Words To Write	INT	5	
4	VAR	Destination Addr	INT	0	Starting 16-bit address for storing data in the intelligent unit
5	VAR				

Rumpf Fünf Worte, die in **CPUDataToWrt** spezifiziert werden, werden in den Speicher des intelligenten Moduls von Adresse 0 bis 4 geschrieben, wenn Freigabe AN ist.



KOP



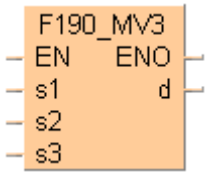
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
     F151_WRT( s1_BankSlot := SlotNo,
          s2_Start := CPUDataToWrt[0],
          n_Number := NoWordsToWrite,
          d_Start := DestinationAddr);
END_IF;
```

F190_MV3

Drei 16-Bit-Daten Transfer

Erklärung Die Funktion kopiert drei 16-Bit-Werte an den Eingängen **s1**, **s2** und **s3** in ein ARRAY mit drei Elementen, das am Ausgang **d** zurückgegeben wird.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

☞ **Mit dem Befehl F7_MV2 (s. S. 771) oder P7_MV2 können Sie zwei 16-Bit-Werte auf einmal transferieren.**

SPS-Typen **Verfügbarkeit von F190_MV3 (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s1, s2, s3	ANY16	16-Bit-Quellregister
d	ARRAY [0..2] OF ANY16	Unteres 16-Bit-Register des 48-Bit-Zielregisters

Die Variablen **s1**, **s2** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1,s2,s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

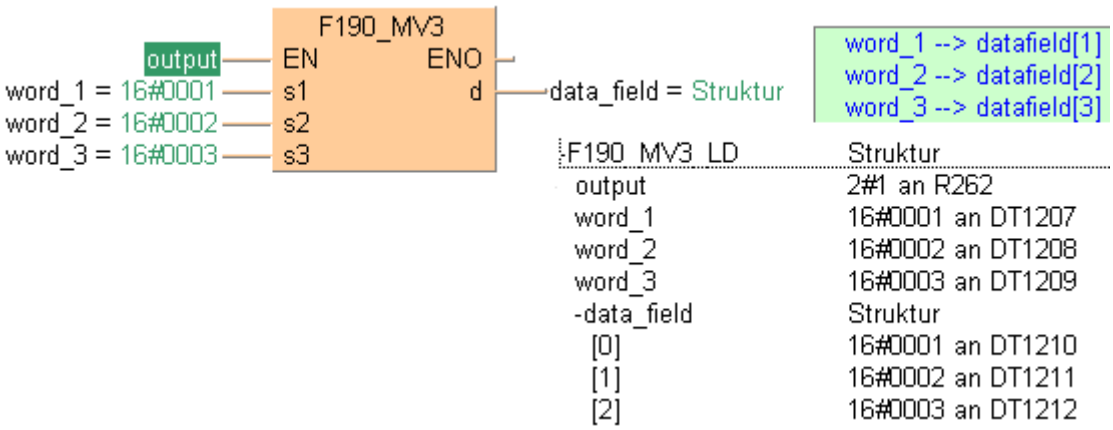
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	output	BOOL	FALSE	activates the function
1	VAR	word_1	WORD	1	
2	VAR	word_2	WORD	2	
3	VAR	word_3	WORD	3	
4	VAR	data_field	ARRAY [0..2] OF WORD	[3(0)]	

Rumpf Wenn die Variable start auf **TRUE** gesetzt wird, wird die Funktion ausgeführt.

KOP



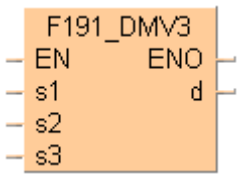
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F190_MV3(word_1, word_2, word_3, data_field);
END_IF;
```

F191_DMV3

Drei 32-Bit-Daten Transfer

Erklärung Die Funktion kopiert drei 32-Bit-Werte an den Eingängen **s1**, **s2** und **s3** in ein ARRAY mit drei Elementen, das am Ausgang **d** zurückgegeben wird.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

☞ **Mit dem Befehl F8_DMV2 (s. S. 773) oder P8_DMV2 können Sie zwei 32-Bit-Werte auf einmal transferieren.**

SPS-Typen **Verfügbarkeit von F191_DMV3 (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s1, s2, s3	ANY32	32-Bit-Quellregister
d	ARRAY [0..2] OF ANY32	unteres 32-Bit-Register des 96-Bit-Zielregisters

Die Variablen **s1, s2, s3** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s1,s2,s3	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

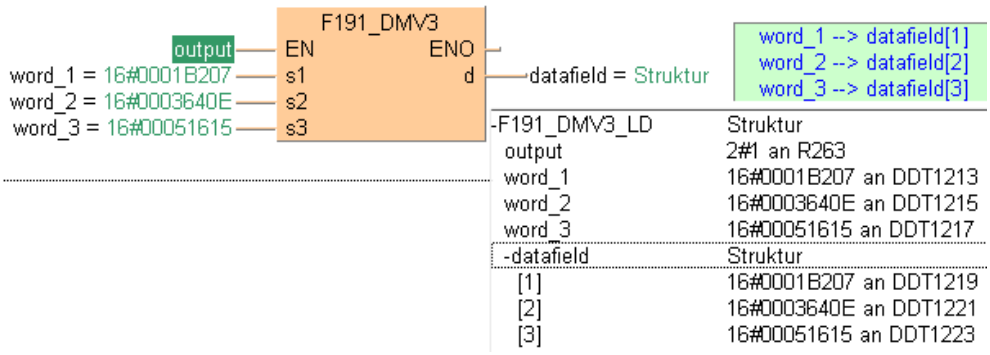
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	word_1	DWORD	111111	
2	VAR	word_2	DWORD	222222	
3	VAR	word_3	DWORD	333333	
4	VAR	data_field	ARRAY [0..2] OF DWORD	[3(0)]	result after a 0->1 leading edge from start: [16#12345678, 16#90123456, 16#78901234]
5	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



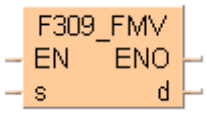
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F191_DMV3(word_1, word_2, word_3, data_field);
END_IF;
```

F309_FMV

Transfer von Fließkommawerten

Erklärung Wird EN gesetzt, dann wird der Fließkommawert (32-Bit-Wert), der mit **s** festgelegt wird, in den Adressbereich, der mit **d** festgelegt wird, kopiert. Der Datenbereich für die REAL-Zahlen kann wie folgt eingestellt werden:



- Positiver Bereich: 0,0000001 bis 9999999,0
- Negativer Bereich: -9999999,0 bis -0,000001

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Es empfiehlt sich, anstelle dieses FP-Befehls den entsprechenden IEC-Befehl zu verwenden MOVE (s. S. 59). Beachten Sie auch die Informationen im Abschnitt Vorteile der IEC-Befehle.

SPS-Typen **Verfügbarkeit von F309_FMV (s. S. 1190)**

☞ **Dieser Befehl kann nicht im Interrupt-Programm programmiert werden.**

Datentypen

Variable	Datentyp	Funktion
s	Fließkommakonstante	Fließkommawert, 32 Bit-Quellbereich
d	REAL	32-Bit-Zielbereich

Operanden

Für	Merker				T/C		Register			Konstante
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

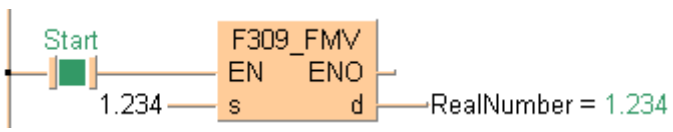
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Start	BOOL	FALSE
1	VAR	RealNumber	REAL	0.0

Rumpf Wenn die Variable **Start** auf TRUE gesetzt wird, wird der Fließkommawert am Eingang **s** in den 32-Bit-Bereich kopiert, den der Compiler der Variablen **RealNumber** zugewiesen hat. Das Monitorwertsymbol ist aktiviert.

KOP



22.1 Datenübertragung von und zu Sonderdatenregistern

FPWIN Pro bietet drei Möglichkeiten an, von Sondermerkern/Sonderdatenregistern zu lesen oder auf sie zu schreiben.

1. Über Systemvariablen (empfohlen ab Version 5.1)

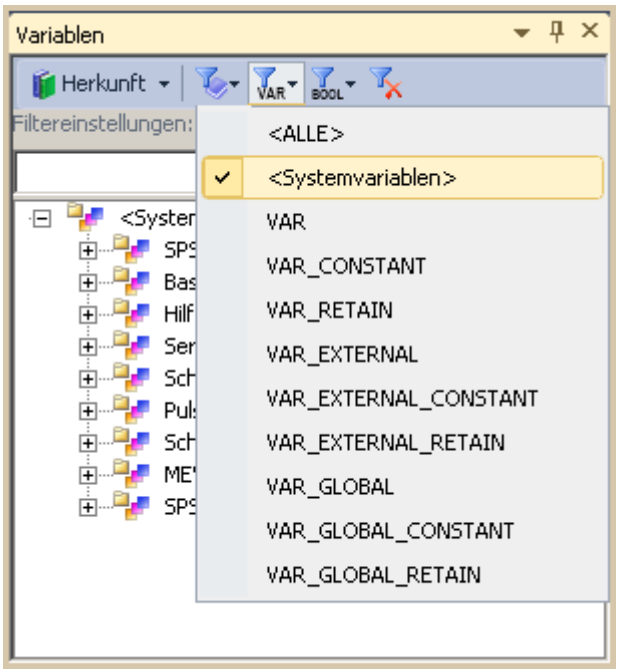
   Für jedes Sonderdatenregister und jeden Sondermerker gibt es eine Systemvariable mit folgender Syntax:

   sys_ * _system variable

   - b BOOL
   - w WORD
   - dw DWORD
   - i INT
   - di DINT

   Sie können diese Systemvariablen über das Dialogfeld "Variablen" in den Rumpf einfügen.

   Tipp: Setzen Sie den Klassenfilter  auf <Systemvariablen>, um ausschließlich die Systemvariablen anzuzeigen.



   Zusätzlich werden diese Systemvariablen auch unter **Monitor → Sondermerker und -datenregister** als letzte Einträge in den Kommentaren angezeigt, z.B. "**sys_w_HSC_ControlFlags**".

   Beispiel für den Zugriff auf die Sonderdatenregister der Schnellen-Zähler

   Beispiel für den Zugriff auf die Sonderdatenregister der Echtzeituhr

2. Über Globale Variablen
3. Über die direkte Adressvergabe im Rumpf

22.2 Datenübertragung zu und von File-Registerbank 1 oder 2

In diesem Abschnitt:

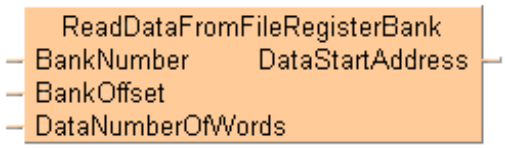
ReadDataFromFile RegisterBank

Daten aus File-Registerbank 1 oder 2 lesen

Erklärung

Der Befehl liest die Anzahl der Daten, angegeben durch **DataNumberOfWords**, aus der File-Registerbank 1 oder 2, angegeben durch **BankNumber**, beginnend mit **BankOffset** und schreibt sie in **DataStartAddress**.

Mit dieser Funktion lassen sich keine Daten im FL-Bereich (File Register Bank 0) lesen, d.h. die Variable, die bei **DataStartAddress** verwendet wird, darf sich nicht im FL-Bereich befinden.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von ReadDataFromFileRegisterBank (s. S. 1195)

Datentypen

Variable	Datentyp	Funktion
BankNumber	INT	gibt die Banknummer an
BankOffset	INT	gibt den Offset der Banknummer an
DataNumberOfWords	INT	Anzahl der Worte, die aus der File-Registerbank gelesen werden
DataStartAddress	ANY16	gibt die Anfangsadresse der Daten an; diese wird aus der File-Registerbank gelesen

Operanden

Für	Merker				T/C		Register			Konstante
BankNumber	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
BankOffset	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
DataNumberOfWords	WX	WY	WR	WL	-	-	DT	LD	FL	-
DataStartAddress	WX	WY	WR	WL	-	-	DT	LD	-	-

Beispiel

In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

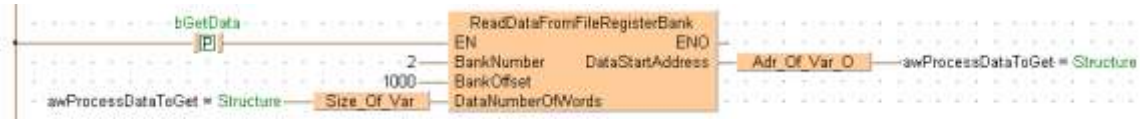
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	awProcessDataToStore	ARRAY [0..10] OF INT	[-111,111,222,333,444,555,666,777,888,999,1100]
1	VAR	awProcessDataToGet	ARRAY [0..10] OF INT	[11(0)]
2	VAR	bStoreData	BOOL	FALSE
3	VAR	bGetData	BOOL	FALSE

Rumpf

Wenn sich **bGetData** von FALSE in TRUE, ändert, wird die gesamte SDT-Variable **awProcessDataToGet** (ein SDT enthält 11 Elemente) mit den Daten aus der File-Registerbank 2 BankOffset 1000 gefüllt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

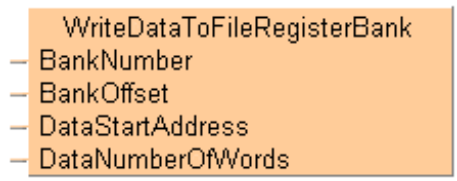
```
if (DF(bGetData)) then
      ReadDataFromFileRegisterBank(BankNumber := 2,
      BankOffset := 1000,
      DataNumberOfWords := Size_Of_Var(awProcessDataToGet),
      DataStartAddress => Adr_Of_Var(awProcessDataToGet));
end_if;
```

WriteDataToFile RegisterBank

Daten in File-Registerbank 1 oder 2 schreiben

Erklärung

Der Befehl entnimmt die Anzahl der Worte, angegeben durch **DataNumberOfWords**, aus **DataStartAddress** und schreibt sie in die File-Registerbank 1 oder 2, angegeben durch **BankNumber** beginnend mit **BankOffset**.



Mit dieser Funktion lassen sich keine Daten in den FL-Bereich (File Register Bank 0) schreiben, d.h. die Variable, die bei **DataStartAddress** verwendet wird, darf sich nicht im FL-Bereich befinden.

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS-Typen

Verfügbarkeit von WriteDataToFileRegisterBank (s. S. 1199)

Datentypen

Variable	Datentyp	Funktion
BankNumber	INT	gibt die Banknummer an
BankOffset	INT	gibt den Offset der Banknummer an
DataStartAddress	ANY16	gibt die Anfangsadresse der Daten an; diese wird in die File-Registerbank geschrieben
DataNumberOfWords	INT	Anzahl der Worte, die in die File-Registerbank geschrieben werden

Operanden

Für	Merker				T/C		Register			Konstante
BankNumber	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
BankOffset	WX	WY	WR	WL	-	-	DT	LD	FL	dez., hex.
DataStartAddress	WX	WY	WR	WL	-	-	DT	LD	-	-
DataNumberOfWords	WX	WY	WR	WL	-	-	DT	LD	FL	-

Beispiel

In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

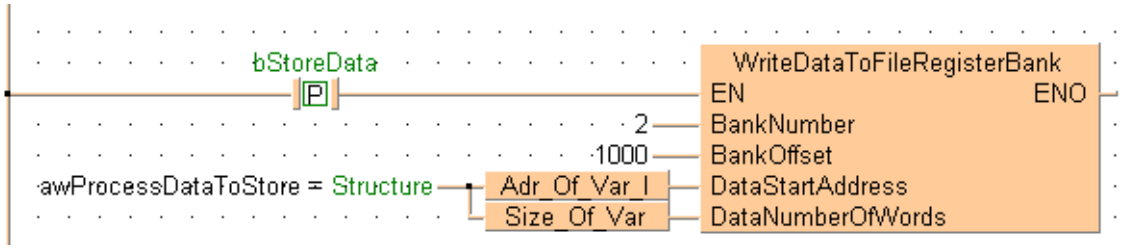
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	awProcessDataToStore	ARRAY [0..10] OF INT	[-111,111,222,333,444,555,666,777,888,999,1100]
1	VAR	awProcessDataToGet	ARRAY [0..10] OF INT	[11(0)]
2	VAR	bStoreData	BOOL	FALSE
3	VAR	bGetData	BOOL	FALSE

Rumpf

Wenn sich **bStoreData** von FALSE in TRUE, ändert, wird die gesamte SDT-Variable **awProcessDataToStore** (ein SDT enthält 11 Elemente) mit den Daten aus der File-Registerbank 2 BankOffset 1000 gefüllt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

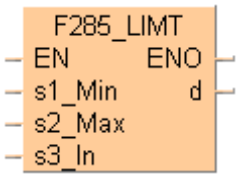
```
if (DF(bStoreData)) then
      WriteDataToFileRegisterBank(BankNumber := 2,
      BankOffset := 1000,
      DataStartAddress := Adr_Of_Var(awProcessDataToStore),
      DataNumberOfWords := Size_Of_Var(awProcessDataToStore));
end_if;
```

Kapitel 23

Auswahlfunktionen

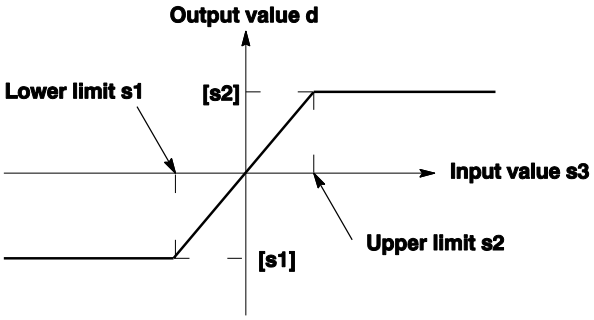
F285_LIMT 16-Bit-Begrenzer

Erklärung Die Funktion vergleicht den Eingangswert am Eingang **s3** mit einem unteren und einem oberen Grenzwert. Der untere Grenzwert wird dem Eingang **s1** und der obere Grenzwert dem Eingang **s2** übergeben. Das Funktionsergebnis wird am Ausgang **d** wie folgt zurückgegeben:



- Ist der Eingangswert am Eingang **s3** < **s1**, wird der untere Grenzwert des Eingangs **s1** am Ausgang **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3** < **s2**, wird der obere Grenzwert des Eingangs **s2** am Ausgang **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s2** ≥ **s3** ≥ **s1**, wird der Eingangswert **s2** an Ausgang **d** zurückgegeben.

Wenn Sie den Ausgangswert nur durch den oberen Grenzwert am Eingang **s2** steuern wollen, geben Sie am Eingang **s1** den Wert -32768 oder 16#8000 ein. Soll nur der untere Grenzwert benutzt werden, geben Sie am Eingang **s2** den Wert 32767 oder 16#7FFF ein.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F285_LIMT (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	der Bereich, in dem der untere Grenzwert gespeichert ist oder die unteren Grenzwert-Daten
s2		der Bereich, in dem der obere Grenzwert gespeichert ist oder die oberen Grenzwert-Daten
s3		der Bereich, in dem der Eingangswert gespeichert ist oder die Eingangswert-Daten
d		der Bereich, in dem die Ausgangswert-Daten gespeichert sind

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert an s1> s2 ist.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ das Funktionsergebnis zwischen dem unteren und dem oberen Grenzwert liegt.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

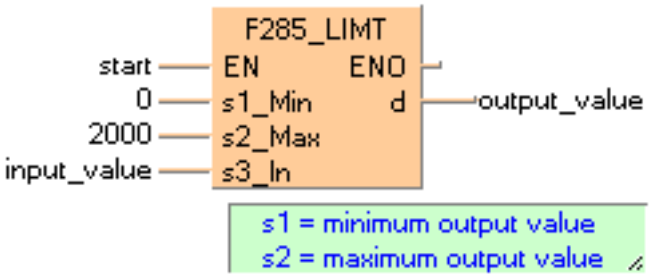
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	INT	2222	
2	VAR	output_value	INT	0	result: here 2000

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. An den Eingängen s1 und s2 stehen die Konstanten 0 (unterer Grenzwert) und 2000 (oberer Grenzwert). Stattdessen können Sie auch Variablen im POE-Kopf deklarieren und im Rumpf an die Eingänge der Funktion schreiben.

KOP



ST

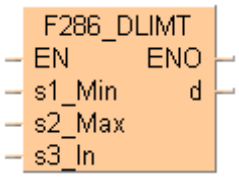
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F285_LIMT( 0, 2000, input_value, output_value);
END_IF;    (* 0=lower limit, 2000=upper limit *)
```

F286_DLIMT

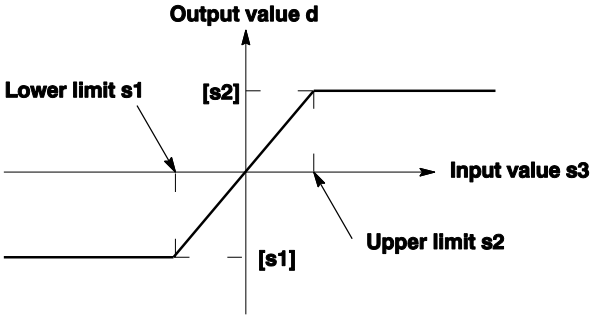
32-Bit-Begrenzer

Erklärung Die Funktion vergleicht den Eingangswert am Eingang **s3** mit einem unteren und einem oberen Grenzwert. Der untere Grenzwert wird dem Eingang **s1** und der obere Grenzwert dem Eingang **s2** übergeben. Das Funktionsergebnis wird am Ausgang **d** wie folgt zurückgegeben:



- Ist der Eingangswert am Eingang **s3** < **s1**, wird der untere Grenzwert des Eingangs **s1** am Ausgang **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s3** < **s2**, wird der obere Grenzwert des Eingangs **s2** am Ausgang **d** zurückgegeben.
- Ist der Eingangswert am Eingang **s2** ≥ **s3** ≥ **s1**, wird der Eingangswert **s2** an Ausgang **d** zurückgegeben.

Wenn Sie den Ausgangswert nur durch den oberen Grenzwert am Eingang **s2** steuern wollen, geben Sie am Eingang **s1** den Wert -2147483648 oder 16#80000000 ein. Soll nur der untere Grenzwert benutzt werden, geben Sie am Eingang **s2** den Wert 2147483647 oder 16#7FFFFFFF ein.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F286_DLIMT (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY32	der Bereich, in dem der untere Grenzwert gespeichert ist oder die unteren Grenzwert-Daten
s2		der Bereich, in dem der obere Grenzwert gespeichert ist oder die oberen Grenzwert-Daten
s3		der Bereich, in dem der Eingangswert gespeichert ist oder die Eingangswert-Daten
d		der Bereich, in dem die Ausgangswert-Daten gespeichert sind

Operanden

Für	Merker				T/C		Register			Konstante
s1, s2, s3	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	dez., hex.
d	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ der Wert an s1> s2 ist.
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ das Funktionsergebnis zwischen dem unteren und dem oberen Grenzwert liegt.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

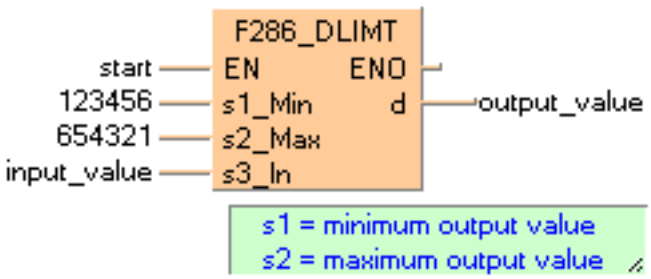
	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function
1	VAR	input_value	DINT	0	
2	VAR	output_value	DINT	0	

In diesem Beispiel wird die Eingangsvariable **input_value** deklariert. Stattdessen können Sie im Rumpf eine Konstante auch direkt an den Eingang der Funktion schreiben.

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt. An den Eingängen s1 und s2 stehen die Konstanten -123456 (unterer Grenzwert) und 654321 (oberer Grenzwert). Stattdessen können Sie auch Variablen im POE-Kopf deklarieren und im Rumpf an die Eingänge der Funktion schreiben.

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F286_DLIMT( 123456, 654321, input_value, output_value);
END_IF;      (* 123456= lower limit, 654321=upper limit *)
```

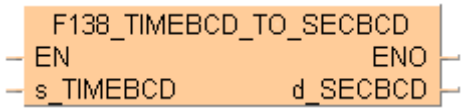
Kapitel 24

Arithmetische Funktionen für Datentypen der Zeit

F138_TIMEBCD_TO_SECBCD

h:min:s -> s Umwandlung

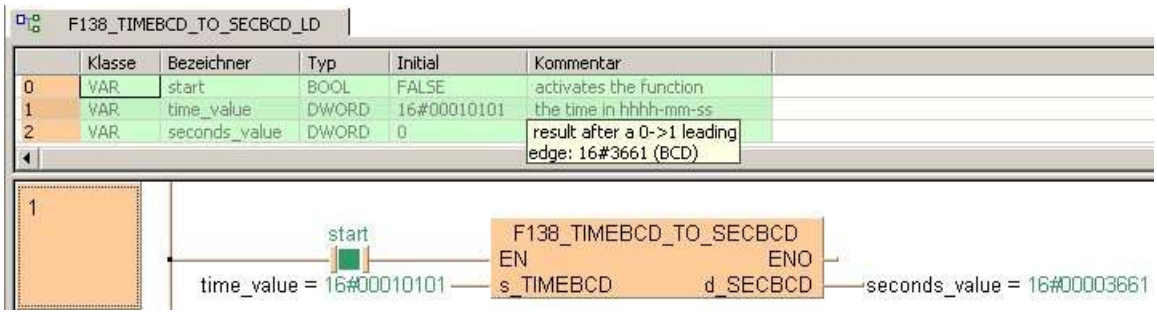
Erklärung Die Zeitangabe Stunden/Minuten/Sekunden des 32-Bit-Wertes des Speicherregisters **s** wird in die Zeitangabe Sekunden konvertiert, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Das Ergebnis steht als 32-Bit-Wert im Speicherregister **d**. Quell- und Zieldaten sind BCD-codiert. Es können maximal 9.999 Stunden, 59 Minuten und 59 Sekunden in 35.999.999 Sekunden umgewandelt werden.

SPS-Typen **Verfügbarkeit von F138_TIMEBCD_TO_SECBCD (s. S. 1187)**

Beispiel:



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Datentypen

Variable	Datentyp	Funktion
s_TIMEBCD	DWORD	Quellbereich zum Speichern der Stunden, Minuten und Sekunden
d_SECBCD	DWORD	Zielbereich zum Speichern der konvertierten Sekunden

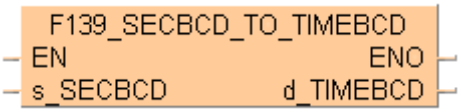
Operanden

Für	Merker				T/C		Register			Konst.
s_TIMEBCD	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
d_SECBCD	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

F139_SECBCD_TO_TIMEBCD

s -> h:min:s Umwandlung

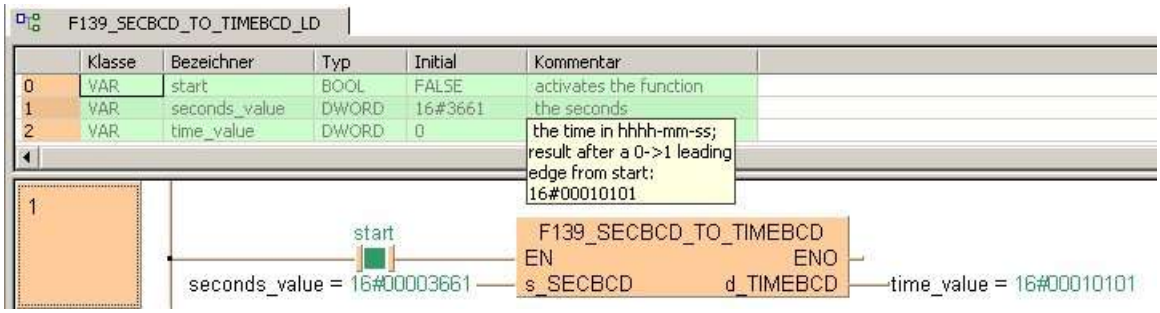
Erklärung Die Zeitangabe Sekunden des 32-Bit-Wertes des Speicherregisters **s** wird in die Zeitangabe Stunden, Minuten und Sekunden konvertiert, wenn der **EN**-Eingang auf TRUE gesetzt ist.



Das Ergebnis steht als 32-Bit-Wert im Speicherregister **d**. Die Sekunden vor der Konvertierung und die Stunden, Minuten und Sekunden nach der Konvertierung liegen im BCD-Format vor. Der maximale Dateneingabewert ist 35.999.999 Sekunden und lässt sich in 9.999 Stunden, 59 Minuten und 59 Sekunden konvertieren.

SPS-Typen **Verfügbarkeit von F139_SECBCD_TO_TIMEBCD (s. S. 1187)**

Beispiel:



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Datentypen

Variable	Datentyp	Funktion
s_SECBCD	DWORD	Quellbereich zum Speichern der Sekunden
d_TIME_BCD	DWORD	Quellbereich zum Speichern der konvertierten Stunden, Minuten und Sekunden

Operanden

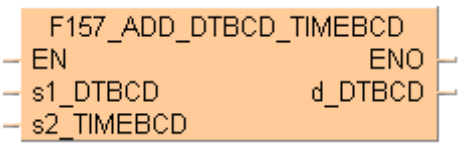
Für	Merker				T/C		Register			Konst.
s_SECBCD	DWX	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-
d_TIME_BCD	-	DWY	DWR	DWL	DSV	DEV	DDT	DLD	DFL	-

F157_ADD_DTBCD _TIMEBCD

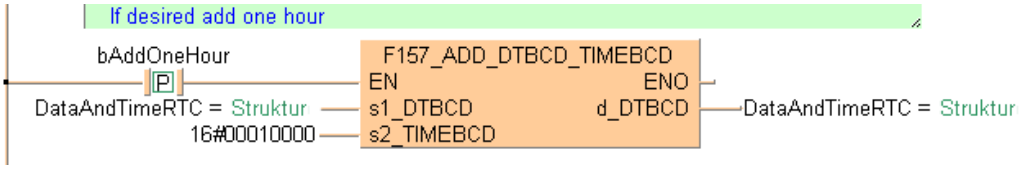
Addition zweier Zeiten

Erklärung Addiert eine Zeitdauer zu einer Datums- und Uhrzeitangabe und speichert das Ergebnis im definierten Bereich. Wenn die Ausführungsbedingung gesetzt ist, wird die Zeitdauer **s2_TIMEBCD** (2 Worte) zur Datums- und Uhrzeitangabe **s1_DTBCD** (3 Worte) addiert. Das Ergebnis wird in der Adresse **d_DTBCD** (3 Worte, Format wie **s1_DTBCD**) gespeichert. Dieser Befehl behandelt alle Daten im BCD-Format.

Symbol:



Example:



☞ **Die Datenregister DT9054 bis DT9056 (DT90054 bis DT90056 für FP2/FP2SH, FP10/FP10S und FP10SH) dürfen im Operanden d_DTBCD nicht angegeben werden. Diese Register speichern die internen Echtzeituhrwerte. Um die Datums- und Zeitangabe in der Echtzeituhr zu ändern, speichern Sie zunächst das Ergebnis in einem anderen Speicherbereich und übertragen Sie es anschließend mit dem Befehl SET_RTC_DTBCD (s. S. 838) in die Sonderdatenregister.**

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Beispiel 1: Angabe von Datum und Uhrzeit im Format DTBCD	**SDT-Element**	**Ergebnis**
1. August 1992, Zeit: 14:23:31 (Std./Min./Sek)	MinSec	16#2331 (Min./Sek.)
	DayHour	16#0114 (Tag/Std.)
	YearMon	16#9208 (Jahr/Monat)
Beispiel 2: Zeitangabe im Format TIMEBCD		
32 Stunden; 50 Minuten; 45 Sekunden		16#00325045 hex (Stunden/Minuten/Sekunden)

SPS-Typen **Verfügbarkeit von F157_ADD_DTBCD_TIMEBCD (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s1_DTBCD	DTBCD	Augend, Uhrzeit und Datum, Werte im BCD-Format
s2_TIMEBCD	DWORD	Summand, 32-Bit-Datenbereich für das Speichern der Zeitdauer im BCD-Format
d_DTBCD	DTBCD	Summe im BCD-Format

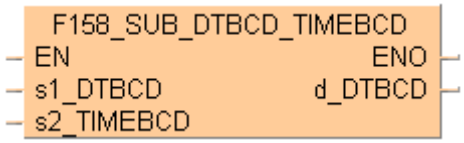
Operanden

Für	Merker				T/C		Register			Konst.
s1_DTBCD	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2_TIMEBCD	-	WY	WR	WL	SV	EV	DT	LD	FL	-
d_DTBCD	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

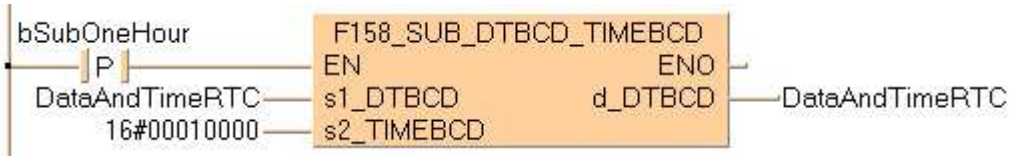
F158_SUB_DTBCD_TIMEBCD

Subtraktion zweier Zeiten

Erklärung Subtrahiert eine Zeitdauer von einer Datums- und Uhrzeitangabe und speichert das Ergebnis im definierten Bereich. Wenn die Ausführungsbedingung gesetzt ist, wird die Zeitdauer s2_TIMEBCD (2 Worte) von der Datums- und Uhrzeitangabe s1_DTBCD (3 Worte) subtrahiert. Das Ergebnis wird in der Adresse **d_DTBCD** (3 Worte, Format wie **s1_DTBCD**) gespeichert. Dieser Befehl verarbeitet alle Daten im BCD-Code.



Beispiel:



☞ **Die Datenregister DT9054 bis DT9056 (DT90054 bis DT90056 für FP2/FP2SH, FP10/FP10S und FP10SH) dürfen im Operanden d_DTBCD nicht angegeben werden. Diese Register speichern die internen Echtzeituhrwerte. Um die Datums- und Zeitangabe in der Echtzeituhr zu ändern, speichern Sie zunächst das Ergebnis in einem anderen Speicherbereich und übertragen Sie es anschließend mit dem Befehl SET_RTC_DTBCD (s. S. 838) in die Sonderdatenregister.**

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Beispiel 1: Angabe von Datum und Uhrzeit im Format DTBCD	**SDT-Element**	**Ergebnis**
1. August 1992, Zeit: 14:23:31 (Std./Min./Sek)	MinSec	16#2331 (Min./Sek.)
	DayHour	16#0114 (Tag/Std.)
	YearMon	16#9208 (Jahr/Monat)
Beispiel 2: Zeitangabe im Format TIMEBCD		
32 Stunden; 50 Minuten; 45 Sekunden		16#00325045 hex (Stunden/Minuten/Sekunden)

SPS-Typen **Verfügbarkeit von F158_SUB_DTBCD_TIMEBCD (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s1_DTBCD	DTBCD	Minuend, Uhrzeit und Datum, Werte im BCD-Format
s2_TIMEBCD	DWORD	Subtrahend, 32-Bit-Datenbereich für das Speichern der Zeitdauer im BCD-Format
d_DTBCD	DTBCD	Ergebnis in BCD-Format

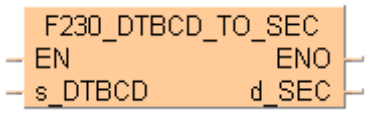
Operanden

Für	Merker				T/C		Register			Konst.
s1_DTBCD	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
s2_TIMEBCD	-	WY	WR	WL	SV	EV	DT	LD	FL	-
d_DTBCD	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

F230_DTBCD_TO_SEC

Zeitumwandlung in Sekunden

Erklärung Diese Funktion wandelt Zeitdaten (Datum und Uhrzeit) in Sekundenzahlen um. Berechnet wird die Zeitspanne zwischen dem angegebenen Zeitpunkt und dem 01.01.2001 0.00 Uhr. Die Zeitdaten werden mit dem strukturierten Datentyp "DTBCD" angegeben.



Eine Umwandlung von Sekunden in Datum und Uhrzeit ist mit dem Befehl F231_SEC_TO_DTBCD (s. S. 836) möglich.

Beispiel:

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F230_DTBCD_TO_SEC (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s_DTBCD	DTBCD	Bereich, in dem die Eingangszeitdaten gespeichert werden.
d_SEC	ANY32	Bereich, in dem die in Sekunden umgewandelten Zeitdaten gespeichert werden (32 Bit).

Operanden

Für	Merker				T/C		Register			Konstante
s_DTBCD	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
d_SEC	-	WY	WR	WL	SV	EV	DT	LD	FL	-

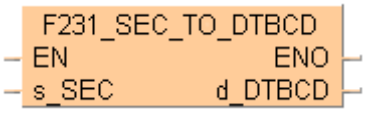
Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die mit Indexmodifizierern definierte Adresse den zulässigen Bereich überschreitet ▪ die für 's' angegebenen Werte keine BCD-Werte sind ▪ ein für 's' angegebener Zeitwert außerhalb des gültigen Bereichs liegt
R9008	%MX0.900.8	kurzzeitig	

F231_SEC_TO_DTBCD

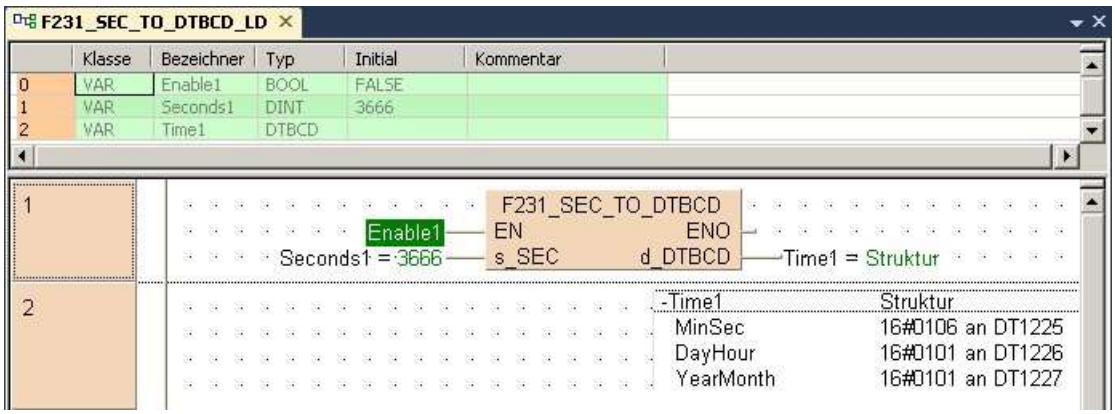
Umwandlung von Sekunden in Datum und Uhrzeit

Erklärung Diese Funktion wandelt eine Sekundenangabe in eine Datums- und Uhrzeitangabe um. Der berechnete Zeitpunkt bezieht sich auf den 01.01.2001 0.00 Uhr.



Eine Umwandlung von Datum und Uhrzeit in Sekunden ist mit dem Befehl F230_DTBCD_TO_SEC (s. S. 835) möglich.

Beispiel:



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F231_SEC_TO_DTBCD (s. S. 1189)**

Datentypen

Variable	Datentyp	Funktion
s_SEC	ANY32	Bereich, in dem die Sekunden gespeichert werden (32 Bit).
d_DTBCD	DTBCD	Bereich in dem Datum und Uhrzeit gespeichert werden.

Operanden

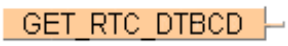
Für	Merker				T/C		Register			Konstante
s_SEC	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
d_DTBCD	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die mit Indexmodifizierern definierte Adresse den zulässigen Bereich überschreitet ▪ die Anzahl der Sekunden 's' > =16#BC191380 (gültig bis 31.12.2100 23:59:59). ▪ der Datenspeicher für 'd' den zulässigen Bereich überschreiten
R9008	%MX0.900.8	kurzzeitig	

ET_RTC_DTBCD **Auswerten der Echtzeituhr**

Erklärung Mit diesem SPS-unabhängigen Befehl können Sie die Daten der Echtzeituhr der SPS auswerten. Der Befehl überträgt die Werte aus den Sonderdatenregistern DT90054 bis DT90056 (DT9054 bis DT9056) in den strukturierten Datentyp DTBCD. Außerdem können Sie die Systemvariablen verwenden, um die Echtzeituhr einzustellen. Für detaillierte Informationen zur Benutzung von Systemvariablen siehe Datenübertragung von/zu Sonderdatenregistern (s. S. 817).



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Beispiel:

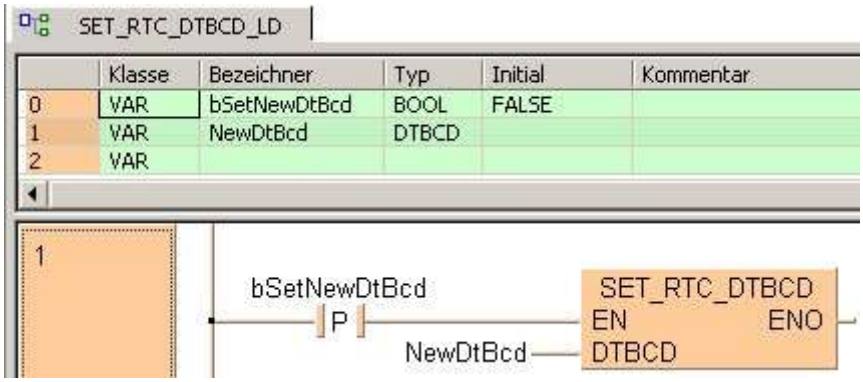
SET_RTC_DTBCD

Echtzeituhr einstellen

Erklärung

Mit diesem SPS-unabhängigen Befehl können Sie die Datums- und Zeitangabe im BCD-Format (DTBCD) auf die Echtzeituhr übertragen. Wenn die Variable **SetNewDtBcd** auf TRUE gesetzt wird, werden die Werte des strukturierten Datentyps DTBCD ain die Sonderdatenregister DT90054 bis DT90056 (DT9054 bis DT9056) übertragen und der Wert 16#8000 wird in das Sonderdatenregister DT90058 (DT9058) geschrieben. Diese Werte stellen die Echtzeituhr auf die neue Zeitangabe ein. Außerdem können Sie die Systemvariablen verwenden, um die Echtzeituhr einzustellen. Für detaillierte Informationen zur Benutzung von Systemvariablen siehe Datenübertragung von/zu Sonderdatenregistern (s. S. 817).

Beispiel:



Kapitel 25

Bistabile Funktionsbausteine

KEEP

Flip-flop

Erklärung Mit dem KEEP-Befehl wird ein Flip-flop realisiert.

Symbol:

Wenn der Setzeingang **SetTrigger** gesetzt ist, wird der Status des angegebenen Merkers gesetzt. Der Status des angegebenen Adressausgangs wird zurückgesetzt, wenn der Rücksetzeingang **ResetTrigger** gesetzt wird. Der Status des Adressausgangs wird behalten, bis der Rücksetzeingang **ResetTrigger** gesetzt ist. Signalwechsel am Setzeingang **SetTrigger** wirken sich nicht mehr auf den Zustand des Flip-flops aus. Wenn beide Eingänge **SetTrigger** und **ResetTrigger** gleichzeitig gesetzt werden, hat der Rücksetzeingang **ResetTrigger** Vorrang und das Flip-flop wird zurückgesetzt.

SPS-Typen **Verfügbarkeit von KEEP (s. S. 1194)**

Datentypen

Variable	Datentyp	Funktion
Set Trigger	BOOL	Setzt Adressausgang, d.h. schaltet ihn auf ON
Reset Trigger	BOOL	Setzt Adressausgang zurück, d.h. schaltet ihn auf OFF
Adresse	BOOL	Spezifiziert Merker, dessen Status (gesetzt oder zurückgesetzt) behalten wird

Operanden

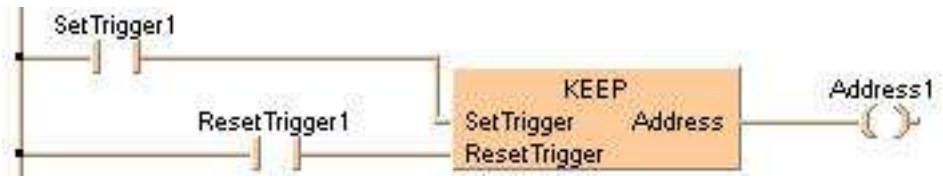
Für	Merker				T/C		Register			Konstante
Set Trigger, Reset Trigger	X	Y	R	L	T	C	-	-	-	-
o	-	Y	R	L	-	-	-	-	-	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	SetTrigger1	BOOL	FALSE	if SetTrigger1 is ON, the output Address1 will turn ON
1	VAR	ResetTrigger1	BOOL	FALSE	if reset_trigger is ON, the
2	VAR	Address1	BOOL	FALSE	Address1 will turn OFF

Rumpf KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Address1 :=KEEP (SetTrigger1, ResetTrigger1);
```

SET

RESET, Ausgang setzen oder rücksetzen

Erklärung

SET: Wenn die Ausführungsbedingungen überprüft worden sind, wird der Ausgang gesetzt und der Zustand wird gehalten.
RST: Wenn die Ausführungsbedingungen überprüft worden sind, wird der Ausgang zurückgesetzt und der Zustand wird gehalten.

- Wenn Sie die Befehle **SET** und **RST** verwenden, können Merker mit der gleichen Nummerierung beliebig häufig eingesetzt werden. (Auch bei einer Gesamtüberprüfung wird dies nicht als Syntaxfehler behandelt.)
- Wenn die Befehle **SET** und **RST** verwendet werden, wechselt der Ausgang während des Betriebs mit jedem Ausführungsschritt.
- Um Ergebnisse während des laufenden Betriebs zu erhalten, verwenden Sie den Befehl für die parallele E/A-Aktualisierung (**F143**).
- Die Ausgangsfestlegung durch einen **SET**-Befehl wird auch während der Ausführung eines **MC**-Befehls beibehalten.
- Die Ausgangsfestlegung durch einen **SET**-Befehl wird zurückgesetzt, wenn der Modus von RUN zu PROG geändert wird oder die Stromversorgung unterbrochen wird. Dies gilt auch dann, wenn ein interner Merker selbst erhaltenden Typs als Ausgangsfestlegung eingesetzt wird.
- Durch Voranstellen eines **DF**-Befehls (oder durch die Festlegung einer steigenden Flanke im KOP) vor die Befehle **SET** und **RST** wird sichergestellt, dass der Befehl nur bei steigender Flanke ausgeführt wird.

Merker

- Merker können mit dem Befehl **RST** ausgeschaltet werden.
- Die Verwendung verschiedener Merker mit des Befehlen **SET** und **RST** führt nicht zu doppelten Ausgangssignalen.
- Es ist nicht möglich, einen Pulsmerker (P) als Ausgang für einen **SET-** oder **RST-**Befehl festzulegen.

Operanden

Für	Merker				T/C		Register			Konstante
SET RST	-	Y	R	L	-	-	-	E	-	-

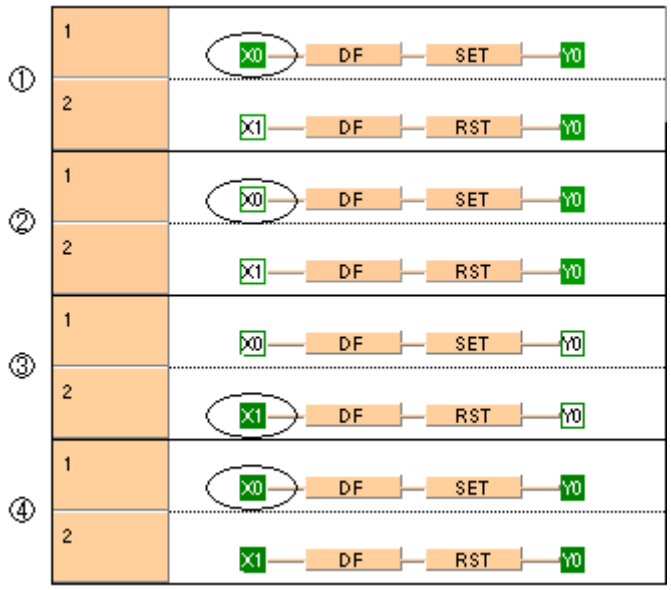
Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe. Solange Adressen durch die Verwendung von FP-Adressen zugeordnet werden, ist kein POE-Kopf nötig.

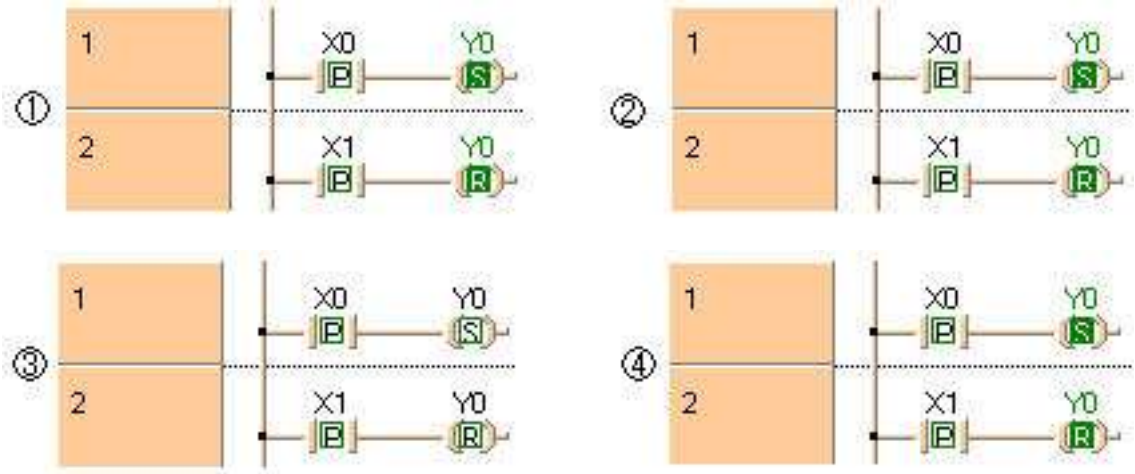
Rumpf

Der Gebrauch des DF-Befehls oder die Festlegung einer steigenden Flanke verbessert das Programm, da ein Programmierschritt nur für eine Überprüfung gültig ist:
(1) Wenn der Eingang X0 aktiviert ist, wird der Ausgang Y0 gesetzt.
(2) Wenn der Eingang X0 deaktiviert ist, bleibt der Ausgang Y0 gesetzt.
(3) Wenn der Eingang X1 aktiviert ist, wird der Ausgang Y0 zurückgesetzt.
(4) Wenn der Eingang X0 wieder aktiviert ist, wird der Ausgang Y0 gesetzt.

FBS



KOP Im Kontaktplan wird eine steigende Flanke am Kontakt und der Befehl SET oder RESET an der Spule festgelegt:



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

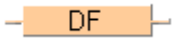
```
(*TRUE and FALSE are assigned to Y0*)
IF DF(X0) THEN
        Y0:= TRUE;
END_IF;

IF DF(X1) THEN
        Y0:= FALSE;
END_IF;
```

Kapitel 26

Flankenerkennung

DF — Positive Flankenerkennung

Erklärung Mit dem **DF**-Befehl realisieren Sie die Auswertung der steigenden (positiven) Flanke beim Signalwechsel eines vorgeschalteten Operanden. Wenn der Signalzustand des vorgeschalteten Operanden **i** von 0 auf 1 wechselt, wird der Ausgang **o** des **DF**-Befehls für die Dauer des Programmzyklus gesetzt.



SPS-Typen **Verfügbarkeit von DF (s. S. 1185)**

☞ **Vorsicht bei der Programmierung im Zusammenhang mit Anweisungen, welche die Rangfolge der Ausführung ändern. Das können z.B. Sprunganweisungen oder Schleifen innerhalb der Ablaufsprache oder eines Funktionsbausteins sein. Der Ablauf der Anweisungen könnte sich in Abhängigkeit vom Zeitpunkt der Anweisungsausführung und vom Eingangswert verändern. (Systemspezifische Sprung- oder Schleifenanweisungen sind: MC zu MCE Anweisung, JP zu LBL Anweisung, F19_SJP zu LBL Anweisung, LOOP zu LBL Anweisung).**

Datentypen

Variable	Datentyp
Eingang	BOOL
Ausgang	BOOL

Operanden

Für	Merker				T/C		Register			Konstante
i	X	Y	R	L	T	C	-	-	-	-
o	-	Y	R	L	-	-	-	-	-	-

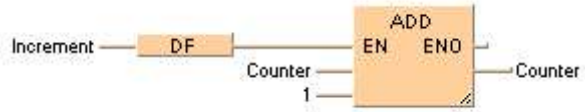
Beispiel In diesem Beispiel wird die Funktion DF im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Increment	BOOL	FALSE
1	VAR	Counter	INT	0

Rumpf Jede steigende Flanke am Eingang **Increment** erhöht den Zähler.

KOP

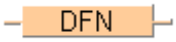


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(Increment) THEN
    Counter:=Counter+1;
END_IF;
```

DFN Negative Flankenerkennung

Erklärung Wenn der Signalzustand des vorgeschalteten Operanden **i** von 1 auf 0 wechselt, wird der Ausgang **o** des **DFN**-Befehls für die Dauer des Programmzyklus gesetzt.



SPS-Typen **Verfügbarkeit von DFN (s. S. 1185)**

☞ **Vorsicht bei der Programmierung im Zusammenhang mit Anweisungen, welche die Rangfolge der Ausführung ändern. Das können z.B. Sprunganweisungen oder Schleifen innerhalb der Ablaufsprache oder eines Funktionsbausteins sein. Der Ablauf der Anweisungen könnte sich in Abhängigkeit vom Zeitpunkt der Anweisungsausführung und vom Eingangswert verändern. (Systemspezifische Sprung- oder Schleifenanweisungen sind: MC zu MCE Anweisung, JP zu LBL Anweisung, F19_SJP zu LBL Anweisung, LOOP zu LBL Anweisung).**

Datentypen

Variable	Datentyp
Eingang	BOOL
Ausgang	BOOL

Operanden

Für	Merker				T/C		Register			Konstante
i	X	Y	R	L	T	C	-	-	-	-
o	-	Y	R	L	-	-	-	-	-	-

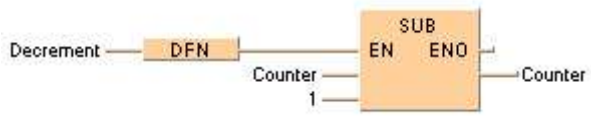
Beispiel In diesem Beispiel wird die Funktion DFN im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Decrement	BOOL	FALSE
1	VAR	Counter	INT	0

Rumpf Jede fallende Flanke am Eingang **Decrement** setzt den Zähler herab.

KOP



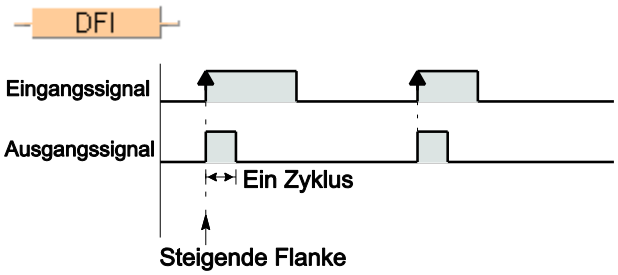
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DFN(Decrement) THEN
    Counter:=Counter-1;
END_IF;
```

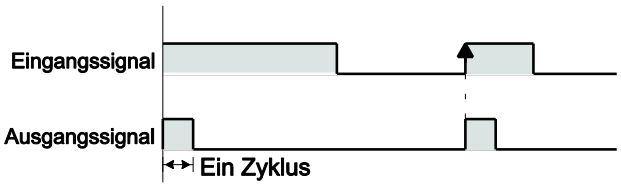
DFI

Differenzierung bei steigender Flanke

Erklärung Die Funktion ändert bei jeder steigenden Flanke des Eingangssignals (Eingang **i**) den Zustand des Ausgangssignals (Ausgang **o**) für die Zeitdauer des aktuellen Zyklus auf TRUE.



Die Erkennung der steigende Flanke des Eingangssignals ist auch im ersten Zyklus gewährleistet.



☞ **Vorsicht bei der Programmierung im Zusammenhang mit Anweisungen, welche die Rangfolge der Ausführung ändern. Das können z.B. Sprunganweisungen oder Schleifen innerhalb der Ablaufsprache oder eines Funktionsbausteins sein. Der Ablauf der Anweisungen könnte sich in Abhängigkeit vom Zeitpunkt der Anweisungsausführung und vom Eingangswert verändern. (Systemspezifische Sprung- oder Schleifenanweisungen sind: MC zu MCE Anweisung, JP zu LBL Anweisung, F19_SJP zu LBL Anweisung, LOOP zu LBL Anweisung).**

Die Anzahl der verwendeten DFI-Funktionen ist nicht limitiert.

Wenn beim Einschalten der Steuerung ein bestehendes Eingangssignal = TRUE nicht als erste steigende Flanke interpretiert werden soll, muss die Funktion DF verwendet werden.

SPS-Typen **Verfügbarkeit von DFI (s. S. 1185)**

Datentypen

Variable	Datentyp
Eingang	BOOL
Ausgang	BOOL

Operanden

Für	Merker				T/C		Register			Konstante
i	X	Y	R	L	T	C	-	-	-	-
o	-	Y	R	L	-	-	-	-	-	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	input_value	BOOL	FALSE
1	VAR	output_value	BOOL	FALSE

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=DFI(input_value);
```

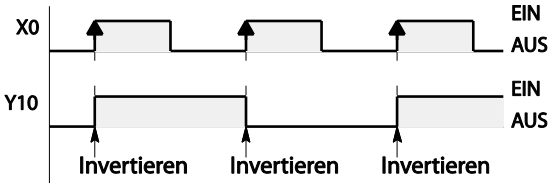
ALT

Invertierung bei steigender Flanke

Erklärung Die Funktion invertiert bei jeder steigenden Flanke des Eingangssignals (Eingang **i**), den Zustand des Ausgangssignals (Ausgang **o**).

Wenn das Eingangssignal beim Wechsel zwischen PROG nach RUN Modus oder beim Einschalten der Versorgungsspannung, wenn sich die Steuerung im RUN Modus befindet, bereits den Wert TRUE besitzt, wird im ersten Zyklus keine steigende Flanke erkannt.

Zeitdiagramm



Vorsicht bei der Programmierung im Zusammenhang mit Anweisungen, welche die Rangfolge der Ausführung ändern. Das können z.B. Sprunganweisungen oder Schleifen innerhalb der Ablaufsprache oder eines Funktionsbausteins sein. Der Ablauf der Anweisungen könnte sich in Abhängigkeit vom Zeitpunkt der Anweisungsausführung und vom Eingangswert verändern. (Systemspezifische Sprung- oder Schleifenanweisungen sind: MC zu MCE Anweisung, JP zu LBL Anweisung, F19_SJP zu LBL Anweisung, LOOP zu LBL Anweisung).

SPS-Typen **Verfügbarkeit von ALT (s. S. 1184)**

Datentypen

Variable	Datentyp
Eingang	BOOL
Ausgang	BOOL

Operanden

Für	Merker				T/C		Register			Konstante
i	X	Y	R	L	T	C	-	-	-	-
o	-	Y	R	L	-	-	-	-	-	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	input_value	BOOL	FALSE
1	VAR	output_value	BOOL	FALSE

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=(ALT(input_value));
```

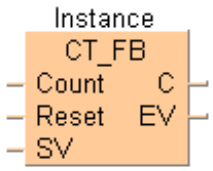
Kapitel 27

Zähler

CT_FB

Rückwärtszähler

Erklärung Die mit dem Funktionsbaustein CT_FB erzeugten Zähler sind Rückwärtszähler. Der Zählbereich **SV** (set value = Sollwert) liegt zwischen 1 und 32767.



Für den Funktionsbaustein CT_FB deklarieren Sie:

Count **Zähleingang**
bei jeder steigenden Flanke an Count wird der Wert 1 vom Istwert **EV** abgezogen, bis der Wert Null erreicht ist

Reset **Rücksetzeingang**
bei jeder steigenden Flanke an Reset wird dem Istwert **EV** der Wert 0 zugewiesen und der Signalausgang **C** zurückgesetzt; bei jeder fallenden Flanke an Reset wird der Wert am Sollwert **SV** dem Istwert **EV** zugewiesen

SV **Sollwert**
ist der Wert, den der Istwert **EV** nach einem Rücksetzvorgang (Reset) hat

C **Signalausgang**
wird gesetzt, wenn der Istwert **EV** den Wert 0 erreicht hat

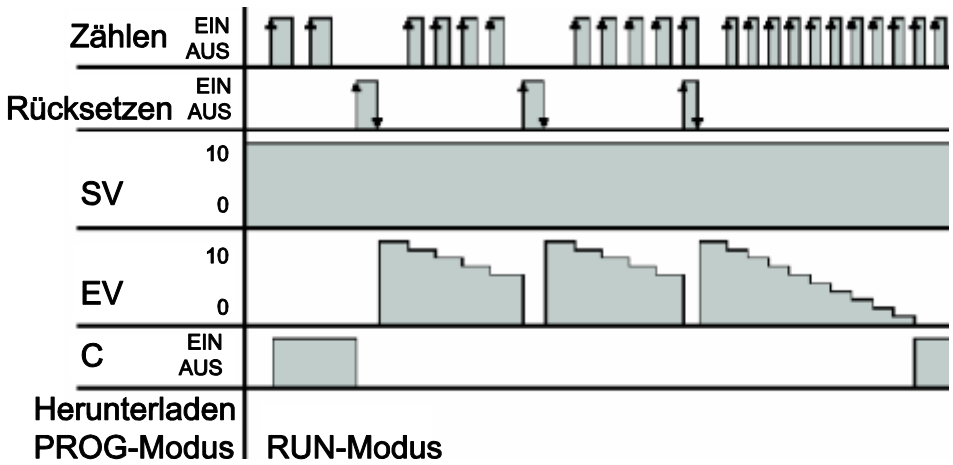
EV **Istwert**
ist der aktuelle Zählwert

SPS-Typen **Verfügbarkeit von CT_FB (s. S. 1185)**

Datentypen

Variable	Datentyp	Funktion
Count	BOOL	Zähleingang (abwärts)
Reset	BOOL	Rücksetzeingang
SV	INT	Sollwert
C	BOOL	wird gesetzt, wenn der Istwert EV den Wert 0 hat
EV	INT	Istwert

Zeitdia-gramm



☞

- **Damit der CT_FB einwandfrei arbeitet, muss vor jeder Benutzung ein Rücksetzvorgang (Reset) erfolgen.**
- **Die Anzahl der verfügbaren Zähler ist begrenzt und hängt von den Einstellungen in den Systemregistern 5 und 6 ab. Der Compiler vergibt an jede Zählerinstanz eine NUM*-Adresse. Die Vergabe erfolgt abwärtszählend, beginnend mit der höchstmöglichen Adresse.**
- **Die FP-Funktion CT (s. S. 853) Rückwärtszähler) benutzt den gleichen Adressbereich (Num*-Eingang). Um Fehlern (Adresskonflikten) vorzubeugen, sollten Sie die Funktion CT und den Funktionsbaustein CT_FB nicht gleichzeitig in einem Projekt verwenden.**

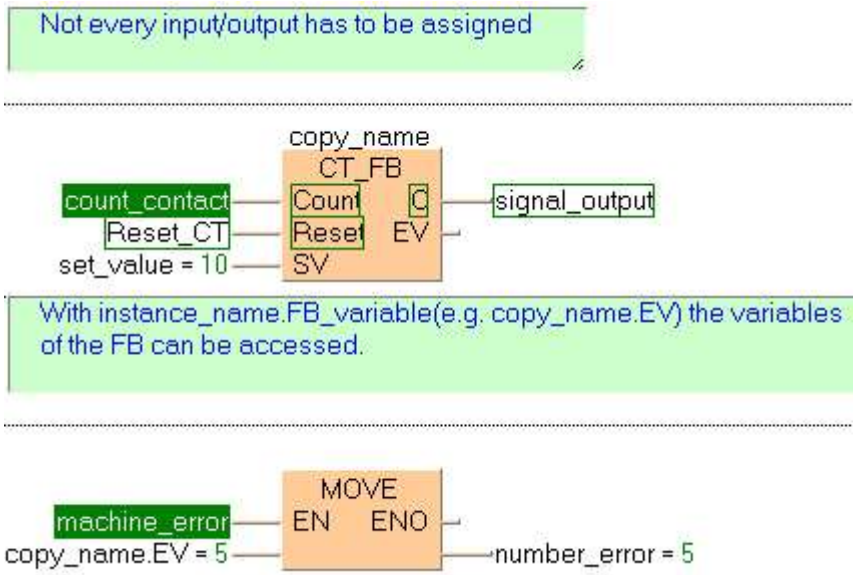
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins CT_FB verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **copy_name** abgespeichert und ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial
0	VAR	copy_name	CT_FB	
1	VAR	set_value	INT	10
2	VAR	signal_output	BOOL	FALSE
3	VAR	count_contact	BOOL	FALSE
4	VAR	Reset_CT	BOOL	FALSE
5	VAR	machine_error	BOOL	FALSE
6	VAR	number_error	INT	0

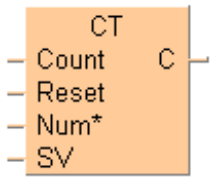
Rumpf In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden. bei jeder steigenden Flanke an **count_contact** wird der Wert 1 von **set_value** abgezogen. **Signal_output** wird auf TRUE gesetzt, wenn set_value den Wert 0 erreicht hat.

KOP



CT — Zähler

Erklärung Erniedrigt schrittweise einen voreingestellten Zähler. Die Funktion umfasst die folgenden Parameter: Count, Reset, Num*, SV und C. Informationen zur Funktion finden Sie in der Datentyptabelle unten.



Wenn der Eingang **Reset** gesetzt ist, wird der Sollwert (**SV**) auf den zugewiesenen Wert zurückgesetzt. Der Sollwert kann als Dezimalkonstante zwischen 0 und 32767 gesetzt werden.

Wenn der Eingang **Count** gesetzt wird, wird der Sollwert schrittweise herabgesetzt. Wenn dieser Wert 0 erreicht, wird die Ausgabe der Zählerwerte (**C**) gestartet.

Wenn die Eingänge **Count** und **Reset** beide zur gleichen Zeit gesetzt werden, erhält der Eingang **Reset** Vorrang.

Wenn der Eingangswert **Count** ansteigt und der Eingangswert **Reset** zur gleichen Zeit fällt, wird der Zählereingang ignoriert und die Voreinstellung ausgeführt.

SPS-Typen **Verfügbarkeit von CT (s. S. 1185)**

Datentypen

Variable	Datentyp	Funktion
Count	BOOL	Bei jeder Aktivierung wird 1 vom Sollwert subtrahiert
Reset	BOOL	Setzt den Zähler zurück, wenn er auf EIN steht
Num*	ANY16	Dem Zähler zugeordnete Nummer (siehe Systemregister 5) muss eine Konstante sein
SV		Der Sollwert ist die Zahl, bei der der Zähler beginnt zu subtrahieren
C	BOOL	Der Zähler wird gesetzt, wenn er den Sollwert SV erreicht

Operanden

Für	Merker				T/C		Register			Konstante
Count	X	Y	R	L	T	C	-	-	-	-
Reset	X	Y	R	L	T	C	-	-	-	-
Num*	-	-	-	-	-	-	-	-	-	dez., hex.
C	-	Y	R	L	-	-	-	-	-	-
SV	-	-	-	-	SV	-	-	-	-	dez., hex.

☞ **Diese Funktion erfordert am Ausgang "C" keine Variable.**

■ **Details zur Zählernummerierung des Rückwärtszählers CT:**

Typ	Anzahl der Zähler	Verwendbare Zählernummern
FP2SH/FP10SH	72 Zähler	3000 bis 3071
FP3	56 Zähler	200 bis 255
FP2	24 Zähler	1000 bis 1023
FP-Sigma	24 Zähler	1000 bis 1023
FP-M C16T **FP1 C14, C16**	28 Zähler	100 bis 127
FP-M C20, C32 **FP-e, FP0 T32C** **FP1 C24, C40, C56, C72**	44 Zähler	100 bis 143

Die Anzahl der Zähler kann in Systemregister 5 geändert werden. Die Anzahl der Zähler kann für FP2SH und FP10SH auf 3.072, für FP-C und FP3 auf 256, für FP-Sigma auf 1.024 und FP2 auf 1.024, für FP-M C16T und FP0 auf 144 erhöht werden. Beachten Sie, dass eine Erhöhung der Zähleranzahl die Anzahl der Zeitgeber herabsetzt.

Die folgenden Nummern können für die Zähler in Abhängigkeit vom FP0 C10/C14/C16/C32-Typ verwendet werden.

Typ	Verwendbare Zählernummern
FP0 C10, C14 und C16	44 Zähler (C100 bis C143) Nicht haltender Typ: 40 Zähler (C100 bis C139) Haltender Typ: 4 Zähler (C140 bis C143)
FP0 C32	44 Zähler (C100 bis C143) Nicht haltender Typ: 28 Zähler (C100 bis C127) Haltender Typ: 16 Zähler (C128 bis C143)

Für alle Modelle mit Ausnahme von FP0 C10, C14, C16 und C32 gibt es Zähler eines haltenden Typs, die den Zählerstatus behalten, wenn die Stromversorgung ausgeschaltet oder der Modus RUN zu PROG geändert wird, sowie Zähler nicht haltenden Typs, die unter diesen Bedingungen zurückgesetzt werden. Systemregister 6 kann für die Festlegung der nicht haltenden Zähler verwendet werden.

■ **Sollwert und Istwert des Rückwärtszählers CT**

Wenn der Reset-Eingang während des Rückzählens zurückgesetzt wird, wird der als Sollwert (SV) festgelegte Wert als Istwert (EV) gesetzt.

Wenn der Reset-Eingang gesetzt ist, wird der Istwert auf 0 zurückgesetzt.

Wenn der Zähler-Eingang gesetzt ist, wird der Sollwert zurückgezählt und der Kontakt Cn (n ist die Zählernummer) wird aktiviert, wenn der Istwert den Wert 0 erreicht.

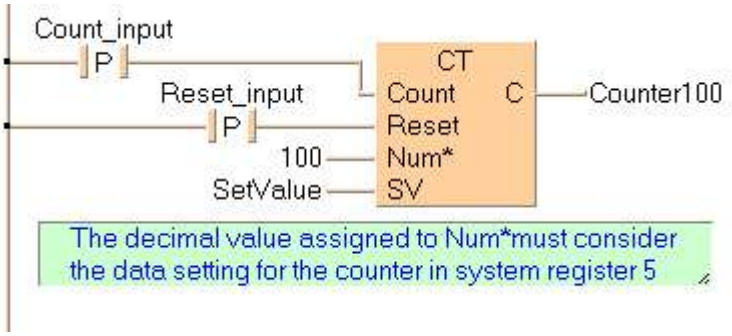
Beispiel In diesem Beispiel wird die Funktion CT im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	Count_input	BOOL	FALSE	Listed in Global Variable List
1	VAR_EXTERNAL	Reset_input	BOOL	FALSE	Listed in Global Variable List
2	VAR	SetValue	INT	10	Decrements by one each time
3	VAR	Counter100	BOOL	FALSE	Turns on when Count_input has been activated 10 times
4	VAR				

Rumpf Der Sollwert SV wird auf 10 gesetzt, wenn **Reset_input** aktiviert ist. Wenn **Count_input** aktiviert ist, wird der Wert SV um 1 zurückgezählt. Wenn der Wert 0 erreicht ist, wird **Counter100** aktiviert. Num* ist die Zählernummer, die mindestens so groß wie die im Systemregister 5 festgelegte Nummer sein muss.

KOP



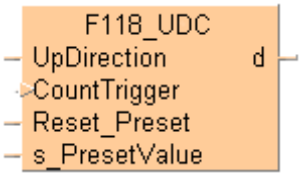
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Counter100 :=CT( Count := Count_input ,
        Reset := Reset_input ,
        Num := 100 ,
        SV := Setvalue );
            (* Num*, 100 in this example, must be a constant *)
```

F118_UDC

Vor- und Rückwärtszähler

Erklärung UD_Trig: Steuereingang; bei Steuereingang = 0 zählt der Zähler rückwärts. Bei Steuereingang = 1 zählt der Zähler vorwärts.



Cnt_Trig: Addiert oder subtrahiert den Wert 1 bei steigender Flanke.

Rst_Trig: Rücksetzeingang; ist aktiv bei Signal = 1.

Das Speicherregister für den aktuellen Zählwert wird 0, wenn die steigende Flanke des Trigger ermittelt wird (FALSE → TRUE). Der Wert im Speicherregister s wird nach d übertragen, wenn eine fallende Flanke des Trigger ermittelt wird (TRUE → FALSE).

s: 16-Bit-Bereich oder äquivalente Konstante für voreingestellten Wert des Zählers

d: 16-Bit-Bereich für Istwert des Zählers

SPS-Typen **Verfügbarkeit von F118_UDC (s. S. 1186)**

Datentypen

Variable	Datentyp	Funktion
UD_Trig	BOOL	Setzt Zähler auf Aufwärts- (ON) oder Abwärtszählen (OFF)
Cnt_Trig	BOOL	Startet Zähler
Rst_Trig	BOOL	Setzt Zähler zurück
s	ANY16	16-Bit-Bereich oder äquivalente Konstante für voreingestellten Wert des Zählers
d		16-Bit-Bereich für Istwert des Zählers

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
UD_Trig, Cnt_Trig, Rst_Trig	X	Y	R	L	T	C	-	-	-	-
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel In diesem Beispiel wird die Funktion F118_UDC im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	up	BOOL	FALSE	declares, if the counter
1	VAR	count	BOOL	FALSE	at a rising edge on count
2	VAR	reset	BOOL	FALSE	resets the counter to
3	VAR	set_value	INT	0	the starting value
4	VAR	output_value	INT	0	the actual value

Rumpf Wenn am Eingang **CNT_Trig** eine steigende Flanke anliegt, wird der Zähler ausgeführt. Die Zählrichtung wird am Eingang **UD_Trig** (TRUE bedeutet vorwärtszählen, FALSE rückwärtszählen). Wenn am Eingang **Rst_Trig** TRUE anliegt, wird der Zähler auf den Anfangswert zurückgesetzt.

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
output_value:=F118_UDC(UD_Trig:=up,Cnt_Trig:=count,Rst_Trig:=reset,s:=
set_value);
(* output_value contains the count value *)
```

Kapitel 28

Schneller Zähler und Pulsausgabe

28.1 Einführung in die schnellen Zähler

Die schnellen Zählerbefehle und die Pulsausgabebefehle können auf folgenden Steuerungen der FP-Serie verwendet werden: FP0, FP-e, FPΣ, FP-X, FP0R.

28.2 Steuercode für schnellen Zähler schreiben

Auf das Sonderdatenregister, das den Steuercode für den schnellen Zähler und die Pulsausgabe speichert, kann mit der Systemvariablen sys_wHscOrPulseControlCode zugegriffen werden. (Die Systemvariable sys_wHscOrPulseControlCode entspricht dem Sonderdatenregister DT90052.)

Funktionen, die vom Steuercode des schnellen Zählers ausgeführt werden:

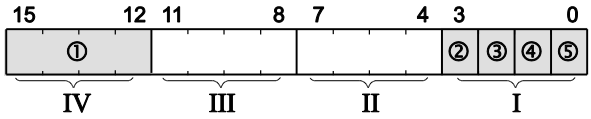
- Schnelle Zählerbefehle löschen (Bit 3)
- Rücksetzeingang (Hardware-Reset) des schnellen Zählers aktivieren/deaktivieren (Bit 2)
- Zählen aktivieren/deaktivieren (Bit 1)
- Istwert des schnellen Zählers auf 0 zurücksetzen (Software-Reset) (Bit 0)

Die Einstellungen des Steuercodes können mit den Systemvariablen sys_wHscChannelxControlCode oder sys_wPulseChannelxControlCode (wobei x=Kanalnummer) für jeden einzelnen Kanal abgefragt werden.

Die Einstellungen dieser Systemvariablen bleiben erhalten, bis sie neu definiert werden.

Beschreibung für FPΣ, FP-X, FP0R:

Die Bits 0–15 des Steuercodes sind in Vierergruppen angeordnet. Die Biteinstellung in jeder Gruppe wird durch eine Hexadezimalzahl dargestellt (z.B. 0002 0000 0000 1001 = 16#2009).



①	Kanalnummer (Kanal n: 16#n)	
②	Schneller Zählerbefehl (Bit 3)	
	0: Fortsetzen	1: Löschen
③	Rücksetzeingang (Bit 2) (siehe Hinweis)	
	0: Aktiviert	1: Deaktiviert
④	Zählen (Bit 1)	
	0: Erlauben	1: Verhindern
⑤	Istwert auf 0 zurücksetzen (Bit 0)	
	0: Nein	1: Ja

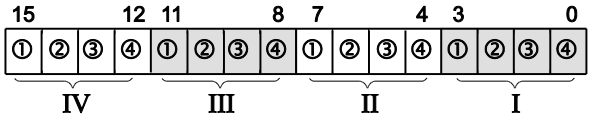
Beispiel: 16#2009

Gruppe	Wert	Beschreibung	
IV	2	Kanalnummer: 2	
III	0	(fest)	
II	0	(fest)	
I	9	Hex 9 entspricht der Binärzahl 1001	
		Schneller Zählerbefehl: Löschen (Bit 3)	1
		Rücksetzeingang: Aktiviert (Bit 2)	0
		Zählen: Erlauben (Bit 1)	0
		Istwert auf 0 zurücksetzen: Ja (Bit 0)	1

Beschreibung für FP0, FP-e:

Die Bits 0–15 des Steuercodes sind in Vierergruppen angeordnet, wobei jede Gruppe die Einstellungen für einen Kanal enthält. Die Biteinstellung in jeder Gruppe wird durch eine

Hexadezimalzahl dargestellt (z.B. 0000 0000 1001 0000 = 16#90).



Gruppe	IV	III	II	I
Kanal	3	2	1	0

①	Schneller Zählerbefehl (Bit 3)	
	0: Fortsetzen	1: Löschen
②	Rücksetzeingang (Bit 2) (siehe Hinweis)	
	0: Aktiviert	1: Deaktiviert
③	Zählen (Bit 1)	
	0: Erlauben	1: Verhindern
④	Istwert auf 0 zurücksetzen (Bit 0)	
	0: Nein	1: Ja

Beispiel: 16#90

Gruppe	**Wert**	**Beschreibung**	
IV	0	–	
III	0	–	
II	9	Kanalnummer: 1 Hex 9 entspricht der Binärzahl 1001	
		Schneller Zählerbefehl: Löschen (Bit 3)	1
		Rücksetzeingang: Aktiviert (Bit 2)	0
		Zählen: Erlauben (Bit 1)	0
		Istwert auf 0 zurücksetzen: Ja (Bit 0)	1
I	0	–	

☞ **Wenn der Rücksetzeingang auf TRUE gesetzt wird, wird der Istwert auf 0 zurückgesetzt. Mit Bit 2 (Rücksetzeingang des schnellen Zählers aktivieren/deaktivieren) können Sie den in den Systemregistern festgelegten Rücksetzeingang deaktivieren.**

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert. Das erste Beispiel zeigt, wie ein Software-Reset für Kanal 0 durchgeführt wird; das zweite Beispiel zeigt, wie ein Software-Reset für Kanal 1 durchgeführt wird.

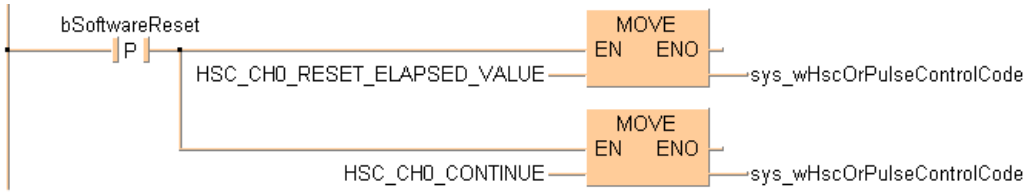
Beispiel 1: Software-Reset für Kanal 0

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bSoftwareReset	BOOL	FALSE	Activates the function
1	VAR_CONSTANT	H5C_CH0_RESET_ELAPSED_VALUE	WORD	16#0001	Resets elapsed value of channel 0
2	VAR_CONSTANT	H5C_CH0_CONTINUE	WORD	16#0000	Continues counting in channel 0

Rumpf Das Zurücksetzen wird in Schritt 1 ausgeführt; dann wird direkt anschließend 0 in Schritt 2 eingegeben, um den Zähler zu starten. Durch das Zurücksetzen alleine wird der Zähler nicht gestartet.

KOP



Software-Reset für Kanal 1 (FPΣ, FP-X, FP0R)

POU header

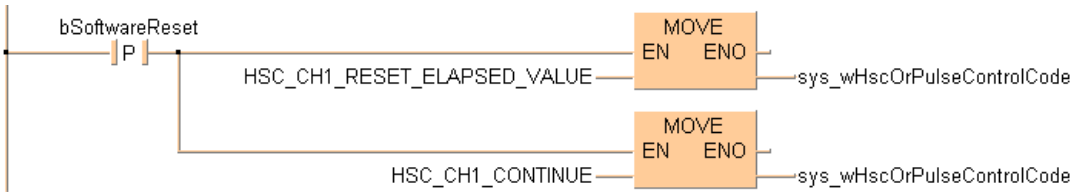
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung dieser Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bSoftwareReset	BOOL	FALSE	Activates the function
1	VAR_CONSTANT	HSC_CH1_RESET_ELAPSED_VALUE	WORD	16#1001	Resets elapsed value of channel 0
2	VAR_CONSTANT	HSC_CH1_CONTINUE	WORD	16#1000	Continues counting in channel 0

Body

Das Zurücksetzen wird in Schritt 1 ausgeführt; dann wird direkt anschließend 0 in Schritt 2 eingegeben, um den Zähler zu starten. Durch das Zurücksetzen alleine wird der Zähler nicht gestartet.

LD



28.3 Istwert des schnellen Zählers schreiben und lesen

Der Istwert wird als 32-Bit-Zeichen in den Sonderdatenregistern gespeichert. Sie können mit der Systemvariablen sys_diHscChannelxElapsedValue auf die Sonderdatenregister zugreifen (wobei x= Kanalnummer).

Systemvariablen für vorgesehene Speicherbereiche:

- FP-Sigma
- FP-X, Transistortypen
- FP-X, Relaistypen
- FP0R
- FP0, FP-e

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert. Das erste Beispiel zeigt, wie ein Anfangswert (Istwert) in den schnellen Zähler geschrieben wird. Das zweite Beispiel zeigt, wie der Istwert gelesen und in eine Variable kopiert wird.

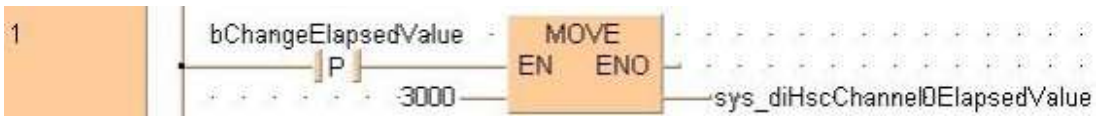
Beispiel 1: Istwert in schnellen Zähler schreiben

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bChangeElapsedValue	BOOL	FALSE	Changes the elapsed value

Rumpf Der Anfangswert 3000 (Istwert) wird in Kanal 0 des schnellen Zählers geschrieben.

KOP



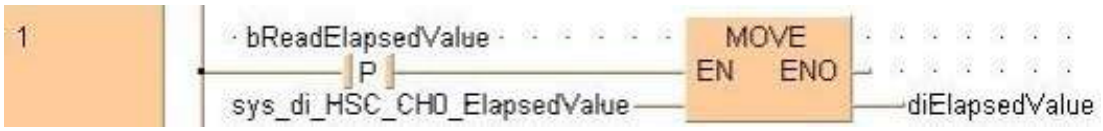
Istwert lesen und in eine Variable kopieren

POU header Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung dieser Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bReadElapsedValue	BOOL	FALSE	Reads the elapsed value
1	VAR	diElapsedValue	DINT	0	Outputs elapsed value

Body Der Istwert des schnellen Zählers wird von Kanal 0 des schnellen Zählers gelesen und in die Variable diElapsedValue kopiert.

LD



28.4 Steuercode für die Pulsausgabe schreiben

Auf das Sonderdatenregister, das den Steuercode für den schnellen Zähler und die Pulsausgabe speichert, kann mit der Systemvariablen sys_wHscOrPulseControlCode zugegriffen werden. (Die Systemvariable sys_wHscOrPulseControlCode entspricht dem Sonderdatenregister DT90052.)

Funktionen, die vom Pulsausgabe-Steuercode ausgeführt werden:

- Referenzpunkt-Sucheingang setzen/zurücksetzen
- Pulsausgabe fortsetzen/stoppen (erzwungener Stopp)
- Zählen aktivieren/deaktivieren
- Istwert des schnellen Zählers zurücksetzen (Software-Reset)
- Schnelle Zähler- und Positionierbefehle löschen (nur FP0R)

Die Einstellungen des Steuercodes können mit den Systemvariablen sys_wHscChannelxControlCode oder sys_wPulseChannelxControlCode (wobei x=Kanalnummer) für jeden einzelnen Kanal abgefragt werden.

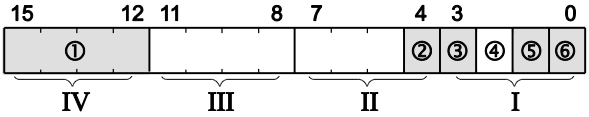
Die Einstellungen dieser Systemvariablen bleiben erhalten, bis sie neu definiert werden.

☞

- **Wenn ein Stopp erzwungen wird, kann dies zu einem unterschiedlichen Zählwert am Ausgang der SPS und am Eingang des Motors führen. Nachdem die Pulsausgabe angehalten wurde, sollten Sie deshalb eine Referenzpunktfahrt ausführen.**
- **Der Referenzpunkt-Sucheingang kann nicht eingestellt werden, wenn das Zählen verhindert oder der Istwert zurückgesetzt werden soll.**

Beschreibung für FPΣ:

Die Bits 0–15 des Steuercodes sind in Vierergruppen angeordnet. Die Biteinstellung in jeder Gruppe wird durch eine Hexadezimalzahl dargestellt (z.B. 0002 0000 0000 1001 = 16#2009).



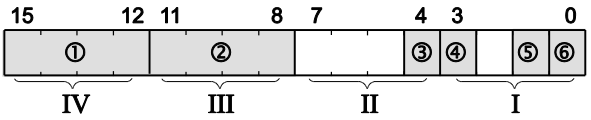
Gruppe IV	①	Kanalnummer (Kanal n: 16#n)	
Gruppe III		0 (fest)	
Gruppe II	②	Referenzpunkt-Sucheingang (Bit 4)	
		0: FALSE	1: TRUE
Gruppe I	③	Pulsausgabe (Bit 3)	
		0: Fortsetzen	1: Stopp
	④	0 (Bit 2, fest)	
	⑤	Zählen (Bit 1)	
		0: Erlauben	1: Verhindern
	⑥	Istwert auf 0 zurücksetzen (Bit 0)	
		0: Nein	1: Ja

Beispiel: 16#2009

Gruppe	Wert	Beschreibung	
IV	2	Kanalnummer: 2	
III	0	(fest)	
II	0	Referenzpunkt-Sucheingang: FALSE	
I	9	Hex 9 entspricht der Binärzahl 1001	
		Pulsausgabe: Stopp (Bit 3)	1
		(Bit 2, fest)	0
		Zählen: Erlauben (Bit 1)	0
		Istwert auf 0 zurücksetzen: Ja (Bit 0)	1

Beschreibung für FP-X:

Die Bits 0–15 des Steuercodes sind in Vierergruppen angeordnet. Die Biteinstellung in jeder Gruppe wird durch eine Hexadezimalzahl dargestellt (z.B. 0002 0001 0000 1001 = 16#2109).



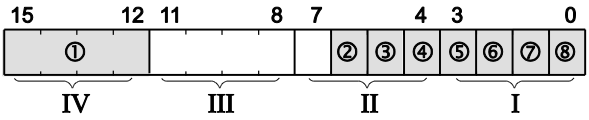
Gruppe IV	①	Kanalnummer (Kanal n: 16#n)	
Gruppe III	②	1 (fest)	
Gruppe II	③	Referenzpunkt-Sucheingang (Bit 4) (siehe Hinweis)	
		0: FALSE	1: TRUE
Gruppe I	④	Pulsausgabe (Bit 3)	
		0: Fortsetzen	1: Stopp
	⑤	Zählen (Bit 1)	
		0: Erlauben	1: Verhindern
	⑥	Istwert auf 0 zurücksetzen (Bit 0)	
		0: Nein	1: Ja

Beispiel: 16#2109

Gruppe	Wert	Beschreibung	
IV	2	Kanalnummer: 2	
III	1	(fest)	
II	0	Referenzpunkt-Sucheingang: FALSE	
I	9	Hex 9 entspricht der Binärzahl 1001	
		Pulsausgabe: Stopp (Bit 3)	1
		(Bit 2, fest)	0
		Zählen: Erlauben (Bit 1)	0
		Istwert auf 0 zurücksetzen: Ja (Bit 0)	1

Beschreibung für FP0R:

Die Bits 0–15 des Steuercodes sind in Vierergruppen angeordnet. Die Biteinstellung in jeder Gruppe wird durch eine Hexadezimalzahl dargestellt (z.B. 0002 0001 0000 1001 = 16#2109).



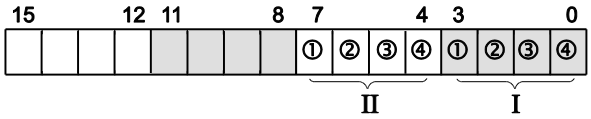
Gruppe I V	①	Kanalnummer (Kanal n: 16#n)	
Gruppe I I I		1 (fest)	
Gruppe I I	②	Startsignal für Positionierung	
		0: Deaktiviert	1: Aktiviert
	③	Signal für gebremsten Halt	
		0: Deaktiviert	1: Aktiviert
	④	Referenzpunkt-Sucheingang (Bit 4) (siehe Hinweis)	
		0: FALSE	1: TRUE
Gruppe I	⑤	Pulsausgabe (Bit 3)	
		0: Fortsetzen	1: Stopp
	⑥	Positionierbefehl löschen (Bit 2)	
		0: Fortsetzen	1: Stopp
	⑦	Zählen (Bit 1)	
		0: Erlauben	1: Verhindern
	⑧	Istwert auf 0 zurücksetzen (Bit 0)	
		0: Nein	1: Ja

Beispiel: 16#2109

Gruppe	Wert	Beschreibung	
I V	2	Kanalnummer: 2	
I I I	1	(fest)	
I I	0	Startsignal für Positionierung: Deaktiviert	
		Signal für gebremsten Halt: Deaktiviert	
		Referenzpunkt-Sucheingang: FALSE	
I	9	Hex 9 entspricht der Binärzahl 1001	
		Pulsausgabe: Stopp (Bit 3)	1
		Positionierbefehl löschen (Bit 2)	0
		Zählen: Erlauben (Bit 1)	0
		Istwert auf 0 zurücksetzen: Ja (Bit 0)	1

Beschreibung für FP0, FP-e:

Die Bits 0–15 des Steuercodes sind in Vierergruppen angeordnet, wobei jede Gruppe die Einstellungen für einen Kanal enthält. Die Biteinstellung in jeder Gruppe wird durch eine Hexadezimalzahl dargestellt (z.B. 0000 0000 1001 0000 = 16#90).



Gruppe	II	I
Kanal	1	0

①	Pulsausgabe (Bit 3)	
	0: Fortsetzen	1: Stopp
②	Referenzpunkt-Sucheingang (Bit 2) (siehe Hinweis)	
	0: FALSE	1: TRUE
③	Zählen (Bit 1)	
	0: Erlauben	1: Verhindern
④	Istwert auf 0 zurücksetzen (Bit 0)	
	0: Nein	1: Ja

Beispiel: 16#90

Gruppe	Wert	Beschreibung	
II	9	Kanalnummer: 1 Hex 9 entspricht der Binärzahl 1001	
		Pulsausgabe: Stopp (Bit 3)	1
		Referenzpunkt-Sucheingang: FALSE (Bit 2)	0
		Zählen: Erlauben (Bit 1)	0
		Istwert auf 0 zurücksetzen: Ja (Bit 0)	1
I	0	–	

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert. Das erste Beispiel zeigt, wie der Referenzpunkt-Sucheingang für Kanal 2 aktiviert wird; im zweiten Beispiel sehen Sie, wie die Pulsausgabe für Kanal 0 beendet wird.

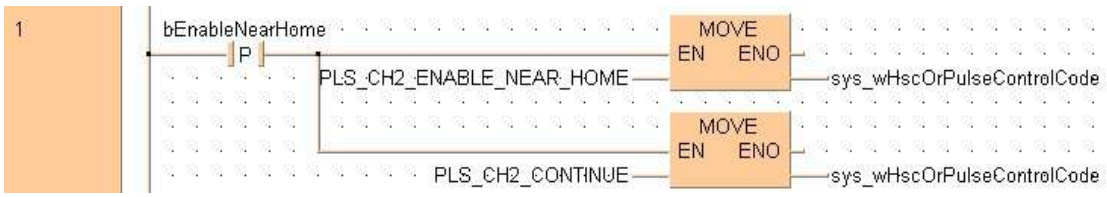
Beispiel 1: Referenzpunkt-Sucheingang für Kanal 2 (FPΣ) aktivieren

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bEnableNearHome	BOOL	FALSE	
1	VAR_CONSTANT	PLS_CH2_ENABLE_NEAR_HOME	WORD	16#2010	Enables near home input for channel 2
2	VAR_CONSTANT	PLS_CH2_CONTINUE	WORD	16#2000	Disables near home input for channel 2 and starts deceleration
3	VAR				

Rumpf Bei Referenzpunktfahrten ist der Referenzpunkt-Sucheingang für Kanal 2 aktiviert.

KOP



Erzwungenen Stopp für Kanal 0 ausführen (FP0, FP-e, FPΣ)

POU header

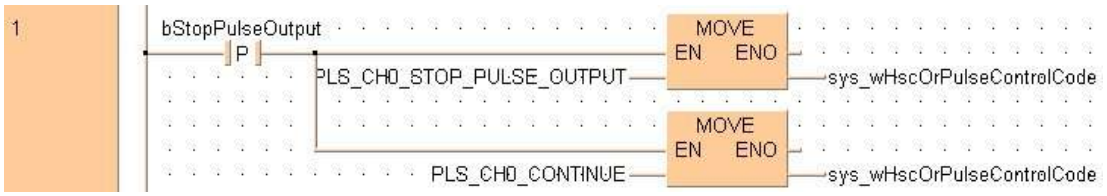
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung dieser Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bStopPulseOutput	BOOL	FALSE	
1	VAR_CONSTANT	PLS_CH0_STOP_PULSE_OUTPUT	WORD	16#0008	Stop pulse output
2	VAR_CONSTANT	PLS_CH0_CONTINUE	WORD	16#0000	Start preset operation

Body

Für Kanal 0 wird ein erzwungener Stopp ausgeführt.

LD



☞ **Wenn ein Stopp erzwungen wird, kann dies zu einem unterschiedlichen Zählwert am Ausgang der SPS und am Eingang des Motors führen. Nachdem die Pulsausgabe angehalten wurde, sollten Sie deshalb eine Referenzpunktfahrt ausführen.**

28.5 Istwert der Pulsausgabe schreiben und lesen

Der Istwert wird als 32-Bit-Zeichen in den Sonderdatenregistern gespeichert. Sie können mit der Systemvariablen sys_diPulseChannelxElapsedValue auf die Sonderdatenregister zugreifen (wobei x= Kanalnummer).

Systemvariablen für vorgesehene Speicherbereiche:

- FP-Sigma
- FP-X, Transistortypen
- FP-X, Relaistypen
- FP0R
- FP0

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert. Das erste Beispiel zeigt, wie Sie vorgehen, um einen Anfangswert (Istwert) in den schnellen Zähler zu schreiben. Das zweite Beispiel zeigt, wie Sie vorgehen, um den Istwert zu lesen und in eine Variable zu kopieren.

Beispiel 1: Istwert in schnellen Zähler schreiben

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bChangeElapsedValue	BOOL	FALSE	Changes the elapsed value

Rumpf Der Anfangswert 3000 (Istwert) wird in Kanal 0 des schnellen Zählers geschrieben.

KOP

Istwert lesen und in eine Variable kopieren

POU header Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung dieser Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bReadElapsedValue	BOOL	FALSE	Reads the elapsed value
1	VAR	diElapsedValue	DINT	0	Outputs elapsed value

Body Der Istwert des schnellen Zählers wird von Kanal 0 des schnellen Zählers gelesen und in die Variable diElapsedValue kopiert.

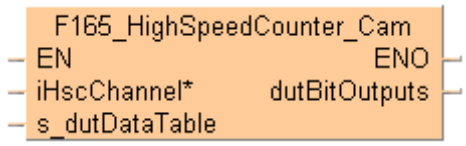
LD

F165_HighSpeedCounter_Cam

Nockensteuerung

Erklärung

Dieser Befehl führt eine Nockensteuerung mit den Parametern im angegebenen SDT und maximal 31 Sollwerten durch. Immer, wenn ein Istwert einem Sollwert entspricht, kann ein Interrupt-Programm ausgeführt werden.

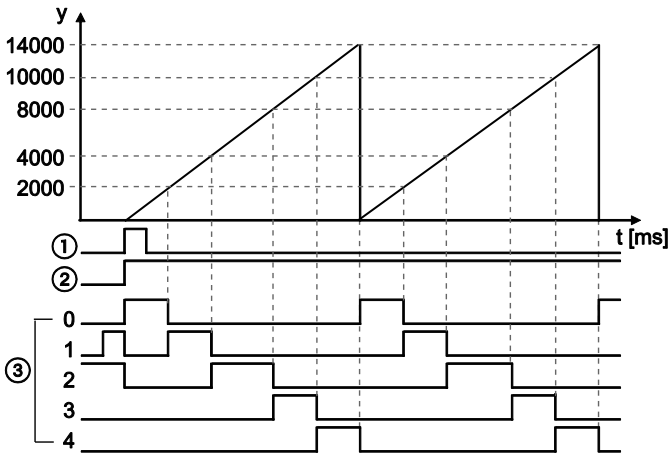


Erstellen Sie auf der Basis des folgenden Musters ihren eigenen SDT:
F165_HighSpeedCounter_Cam_8_Values_DUT

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Wortadresse für Ausgang
- Anzahl der Sollwerte
- Sollwert 1
- ...
- Sollwert n
- Maximaler Sollwert

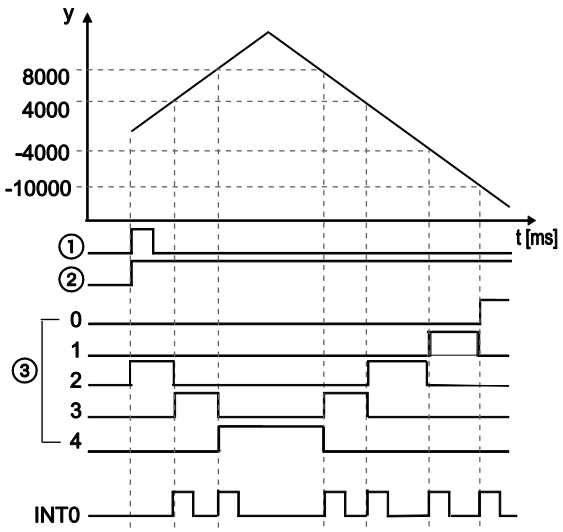
Merkmale der Nockensteuerung



y	Istwert des schnellen Zählers	14000	Maximaler Sollwert
①	Ausführungsbedingung	10000	Sollwert 4
②	Kontrollmerker des schnellen Zählers	8000	Sollwert 3
③	Ausgang 0-4	4000	Sollwert 2
		2000	Sollwert 1

- Immer, wenn der Istwert im Bereich des Sollwerts n bis n+1 (Vorwärtszählen) oder n+1 bis n (Rückwärtszählen) liegt, wird der zugehörige Ausgangsmerker n auf TRUE gesetzt.
- Im Beispiel oben wurde die Maximalwertsteuerung aktiviert. Wenn der Istwert den maximalen Sollwert erreicht, wird der Istwert auf 0 zurückgesetzt und die Zählung beginnt erneut.
- Geben Sie die Wortadresse der Ausgangsmerker in einem überlappenden strukturierten Datentyp an, z.B. BOOL32_OVERLAPPING_DUT, und legen Sie diesen SDT an dutBitOutputs an.
- Es können maximal 31 Sollwerte angegeben werden.

- Die Sollwerte müssen in aufsteigender Folge angegeben werden. Kein Wert darf zweimal vorkommen.
- Sobald die Ausführung beginnt, sind alle Ausgangsmerker auf FALSE gesetzt, ausgenommen Ausgangsmerker 0, der auf TRUE schaltet, sofern der Istwert kleiner ist als Sollwert 1. Ansonsten schaltet der Ausgangsmerker, der dem Sollwertbereich entspricht, auf TRUE. Beispiel: Wenn der Istwert zwischen Sollwert 2 = -4000 und Sollwert 3 = +4000 liegt, ist der Ausgangsmerker 2 TRUE. Im folgenden Beispiel, wurde die Maximalwertsteuerung deaktiviert. Wenn der Istwert den letzten Sollwert erreicht, zählt der Zähler weiter und der Istwert wird nicht auf 0 zurückgesetzt.



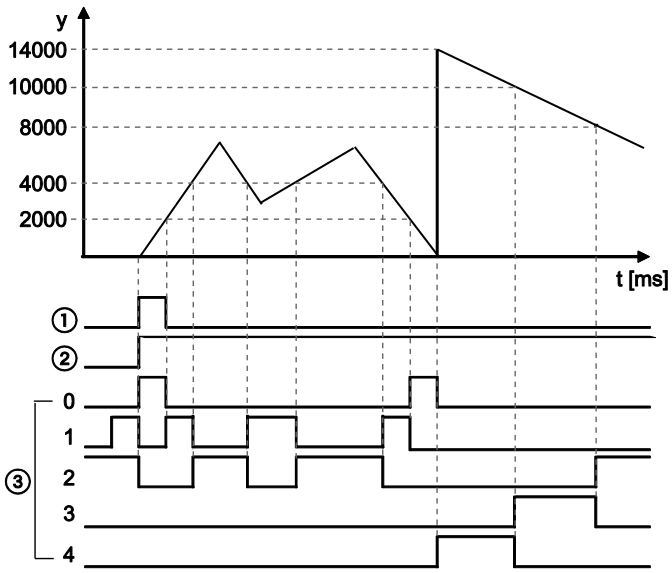
y	Istwert des schnellen Zählers	8000	Sollwert 4
①	Ausführungsbedingung	4000	Sollwert 3
②	Kontrollmerker des schnellen Zählers	-4000	Sollwert 2
③	Ausgang 0-4	-10000	Sollwert 1
INT0	Interrupt-Programm 0		

Maximalwertsteuerung

Wenn der Befehl mit Maximalwertsteuerung ausgeführt wird, wird der Istwert auf 0 zurückgesetzt, sobald der maximale Sollwert erreicht ist. Die Maximalwertsteuerung kann im Steuercode des Befehls F165_HighSpeedCounter_Cam_8_Values_DUT aktiviert werden. Anstatt der Maximalwertsteuerung kann auch der Rücksetzeingang oder ein Software-Reset (s. S. 865) verwendet werden, um den Istwert zurückzusetzen.

Die Maximalwertsteuerung kann nur verwendet werden, wenn für die Sollwerte positive Ganzzahlen gewählt werden.

Vorwärts-/Rückwärtszählung mit Maximalwertkontrolle:



y	Istwert des schnellen Zählers	14000	Maximaler Sollwert
①	Ausführungsbedingung	10000	Sollwert 4
②	Kontrollmerker des schnellen Zählers	8000	Sollwert 3
③	Ausgang 0-4	4000	Sollwert 2
		2000	Sollwert 1

Überblick:

Maximalwertsteuerung:	**Aktiviert**	**Deaktiviert (siehe Hinweis)**
Vorwärtszählen: Der Zeiger der Datentabelle bewegt sich vom Sollwert 1 zum letzten Sollwert.	Wenn der Istwert den maximalen Sollwert erreicht: ▪ Der Zeiger kehrt zu Sollwert 1 zurück ▪ Ausgangsmerker 0 schaltet auf TRUE ▪ Der Istwert wird auf 0 gesetzt	Wenn der Istwert den letzten Sollwert erreicht: ▪ Der Zeiger kehrt zu Sollwert 1 zurück ▪ Ausgangsmerker 0 schaltet auf TRUE ▪ Der Istwertzähler zählt weiter vorwärts und beginnt erneut beim Minimalwert des Ringzählers
Rückwärtszählen: Der Zeiger der Datentabelle bewegt sich vom letzten Sollwert zu Sollwert 1.	Wenn der Istwert den Wert -1 erreicht: ▪ Der Zeiger kehrt zum letzten Sollwert zurück ▪ Der Ausgangsmerker, der dem letzten Sollwert entspricht, schaltet auf TRUE. ▪ Der Istwert wird auf den maximalen Sollwert gesetzt	Wenn der Istwert den Wert -1 erreicht: ▪ Der Zeiger kehrt zu Sollwert n zurück ▪ Der Ausgangsmerker, der dem letzten Sollwert entspricht, schaltet auf TRUE. ▪ Der Istwertzähler zählt weiter rückwärts und beginnt erneut beim Maximalwert des Ringzählers.

☞ **Voraussetzung ist, dass weder ein Rücksetzeingang noch ein Software-Reset verwendet wird.**

Hardware-Reset

Kanal	Hardware-Reset-Eingang
0	X2
1	
2	X5
3	

Interrupt-Betrieb

Das Interrupt-Programm wird ausgeführt, wenn der Istwert dem Sollwert entspricht. Jedes Interrupt-Programm, das in die Taskliste eingegeben wird, ist automatisch aktiviert. Jeder Kanalnummer wird ein spezielles Interrupt-Programm zugewiesen.

Kanal	0	1	2	3	4	5
Interrupt-Programm	0	1	3	4	6	7

Allgemeine Programmierinformation

- Wählen Sie den Eingang des schnellen Zählers für den gewünschten Kanal in den Systemregistern aus.
- Wenn dieser Befehl ausgeführt wird, schaltet der Kontrollmerker für den benutzten Kanal (z.B. sys_bIsHscChannel0ControlActive) auf TRUE. Andere schnelle Zählerbefehle, die denselben Kanal nutzen, können nicht ausgeführt werden, solange der Kontrollmerker auf TRUE steht.
- Um die Ausführung eines Befehls abzubrechen, setzen Sie Bit 3 des Datenregisters, in dem der Steuercode für den schnellen Zähler (sys_wHscOrPulseControlCode) gespeichert wird, auf TRUE. Der Kontrollmerker "Schneller Zähler" wechselt dann zu FALSE. Um die Ausführung des schnellen Zählerbefehls wieder zu aktivieren, setzen Sie Bit 3 auf FALSE zurück.
- Wenn der Istwert des Kanals während der Befehlsausführung überschrieben wird, kann es zu unerwartetem Verhalten kommen.
- Stellen Sie sicher, dass die Zeitspanne zwischen benachbarten Sollwerten größer ist als 1ms.
- Wenn der Befehl im Hauptprogramm ausgeführt wird, muss die minimale Zeitspanne zwischen benachbarten Sollwerten größer sein als die Zykluszeit.
- Wenn der Befehl in einem Interrupt-Programm ausgeführt wird, muss die minimale Zeitspanne zwischen benachbarten Sollwerten größer sein als die maximale Ausführungszeit des Interrupt-Programms.
- Dieser Befehl kann auf maximal zwei Kanälen gleichzeitig ausgeführt werden.
- Wenn ein Rücksetzeingang oder Software-Reset verwendet wird, muss Sollwert 1 eine Ganzzahl und ≥ 1 sein.
- Wenn die Maximalwertsteuerung zusammen mit einem Rücksetzeingang oder Software-Reset verwendet wird, verwenden sie diese nicht gleichzeitig.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F165_HighSpeedCounter_Cam (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
iHscChannel*	INT	Kanal des schnellen Zählers: 0–5
s_dutDataTable	ANY_DUT	Startadresse des Bereichs, der die Datentabelle enthält Beispiel: F165_HighSpeedCounter_Cam_8_Values_DUT
dutBitOutputs	ANY_DUT	Startadresse (WR) des Bereichs, der die Wortadresse für die Merker enthält, z.B. BOOL32_OVERLAPPING_DUT. Wählen Sie die Größe (16 oder 32 Bits) entsprechend der mit diNumberOfTargetValuesAndOutputRelays festgelegten Anzahl aus.

Operanden

Für	Merker				T/C		Register			Konstante
iHscChannel*	-	-	-	-	-	-	DT	-	-	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ die Pulsausgabe nicht in den Systemregistern gesetzt wurde ▪ Sollwert > maximaler Sollwert. ▪ Sollwert= 0: ▪ Die Sollwerte werden nicht in aufsteigender Folge sortiert
R9008	%MX0.900.8	kurzzeitig	

Beispiel

Beispiel 1: Mit Maximalwertsteuerung

In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ
0	VAR_GLOBAL	WR0_bits_F165_CAM_Examples	WR1	%MW0.1	BOOL16_OVERLAPPING_DUT

SDT Der SDT F165_HighSpeedCounter_Cam_8_Values_DUT ist in der FP Library enthalten und lässt sich als Muster nutzen.

F165_Cam_Example1_4_Values_DUT [SDT]

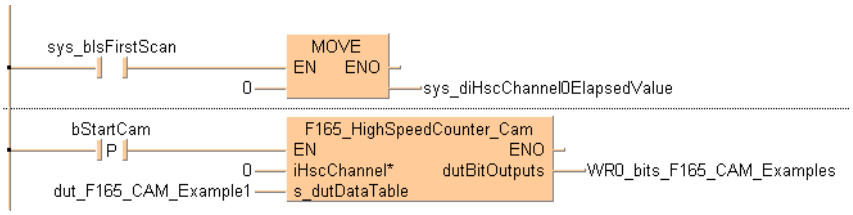
	Bezeichner	Typ	Initial	Kommentar
0	dwCamControlCode	DWORD	16#0010	(without maximum target value: 16#0000, with maximum target value: 16#0010, ...
1	diAddressOffsetInWR	DINT	0	filled with the address information of the argument applied at 'dutBitOutputs'
2	diNumberOfTargetValuesAndOutputRelays	DINT	4	from 1 to 31
3	diTargetValue_1	DINT	2000	
4	diTargetValue_2	DINT	4000	
5	diTargetValue_3	DINT	8000	
6	diTargetValue_4	DINT	10000	
7	diMaximumTargetValue	DINT	14000	

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bStartCam	BOOL	FALSE
1	VAR	dut_F165_CAM_Example1	F165_Cam_Example1_4_Values_DUT	
2	VAR_EXTERNAL	WR0_bits_F165_CAM_Examples	BOOL16_OVERLAPPING_DUT	

Rumpf Wenn die Variable **bStartCam** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF (sys_bIsFirstScan) THEN
      sys_diHscChannel0ElapsedValue:=0;
END_IF;
IF DF(bStartCam) THEN
      F165_HighSpeedCounter_Cam(iHscChannel := 0,
      s_dutDataTable := dut_F165_CAM_Example1,
      dutBitOutputs => WR0_bits_F165_CAM_Examples);
END_IF;
```

Beispiel 2: Ohne Maximalwertsteuerung

In diesem Beispiel wird die Funktion im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert.

DUT Der SDT F165_HighSpeedCounter_Cam_8_Values_DUT ist in der FP Library enthalten und lässt sich als Muster nutzen.

F165_Cam_Example2_4_Values_DUT [SDT]

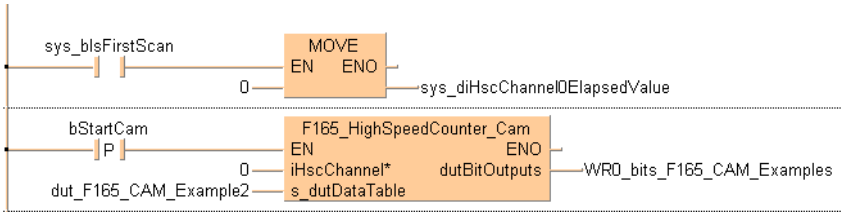
	Bezeichner	Typ	Initial	Kommentar
0	dwCamControlCode	DWORD	16#0000	(without maximum target value: 16#0000, with maximum target value: 16#0010, ...
1	diAddressOffsetInWR	DINT	0	filled with the address information of the argument applied at 'dutBitOutputs'
2	diNumberOfTargetValuesAndOutputRelays	DINT	4	from 1 to 31
3	diTargetValue_1	DINT	-10000	
4	diTargetValue_2	DINT	-4000	
5	diTargetValue_3	DINT	4000	
6	diTargetValue_4	DINT	8000	
7	diMaximumTargetValue	DINT	0	

POU header Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung dieser Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bStartCam	BOOL	FALSE
1	VAR	dut_F165_CAM_Example2	F165_Cam_Example2_4_Values_DUT	
2	VAR_EXTERNAL	WR0_bits_F165_CAM_Examples	BOOL16_OVERLAPPING_DUT	

Body Wenn die Variable **bStartCam** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

LD



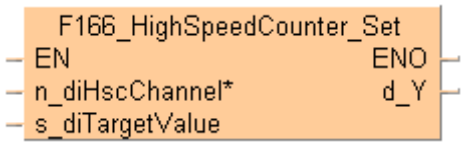
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF (sys_bIsFirstScan) THEN
      sys_diHscChannel0ElapsedValue:=0;
END_IF;
IF DF(bStartCam) THEN
      F165_HighSpeedCounter_Cam(iHscChannel := 0,
      s_dutDataTable := dut_F165_CAM_Example2,
      dutBitOutputs => WR0_bits_F165_CAM_Examples);
END_IF;
```

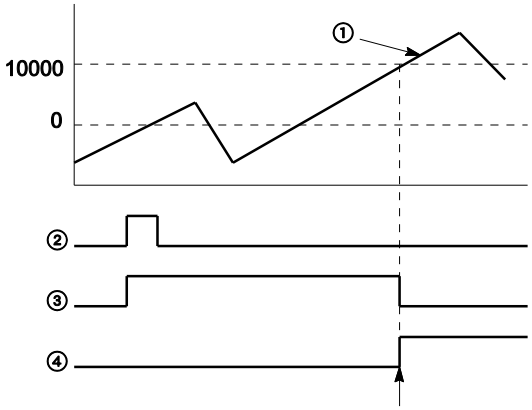
F166_HighSpeedCounter_Set

Zählervergleichsausgang setzen

Erklärung Wenn der Istwert eines schnellen Zählers dem Sollwert entspricht, setzt ein Interrupt-Prozess den angegebenen Ausgang sofort auf TRUE.



Merkmale der Funktion "Zählervergleichsausgang setzen"



10000	Sollwert
①	Istwert des schnellen Zählers
②	Ausführungsbedingung
③	Kontrollmerker des schnellen Zählers
④	SPS-Ausgang

Der SPS-Ausgang schaltet auf TRUE, wenn der Istwert dem Sollwert entspricht. Außerdem wird der Kontrollmerker des schnellen Zählers auf FALSE gesetzt und der Befehl wird deaktiviert.

Wenn ein Ausgang angegeben ist, der nicht implementiert wurde, wird nur die WY-Adresse im Speicher gesetzt oder zurückgesetzt.

Interrupt-Betrieb

Das Interrupt-Programm wird ausgeführt, wenn der Istwert dem Sollwert entspricht. Jedes Interrupt-Programm, das in die Taskliste eingegeben wird, ist automatisch aktiviert.

Kanäle, die von Interrupt-Programmen verwendet werden:

SPS-Typ	FP0, FP-e	FPΣ	FP-X (Relaistypen)	FP-X (Transistortypen)	FP0R
Interrupt 0	Kanal 0	Kanal 0	Kanal 0	Kanal 0	Kanal 0
Interrupt 1	Kanal 1	Kanal 1	Kanal 1	Kanal 1	Kanal 1
Interrupt 2			Kanal 2	Kanal 2	
Interrupt 3	Kanal 2	Kanal 2	Kanal 3	Kanal 3	Kanal 2
Interrupt 4	Kanal 3	Kanal 3	Kanal 4	Kanal 4	Kanal 3
Interrupt 5			Kanal 5	Kanal 5	
Interrupt 6			Kanal 6	Kanal 6	Kanal 4
Interrupt 7			Kanal 7	Kanal 7	Kanal 5

SPS-Typ	FP0, FP-e	FPΣ	FP-X (Relaistypen)	FP-X (Transistortypen)	FP0R
Interrupt 8			Kanal 8		
Interrupt 9			Kanal 9		
Interrupt 10					
Interrupt 11			Kanal A		
Interrupt 12			Kanal B		

Allgemeine Programmierinformation

- Wählen Sie den Eingang des schnellen Zählers für den gewünschten Kanal in den Systemregistern aus.
- FP-X, FP0R: Wenn dieser Befehl ausgeführt wird, schaltet der Kontrollmerker für den benutzten Kanal (z.B. sys_bIsHscChannel0ControlActive) auf TRUE. Andere schnelle Zählerbefehle, die denselben Kanal nutzen, können nicht ausgeführt werden, solange der Kontrollmerker auf TRUE steht.
- FP0, FP-e, FPΣ: Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Um einen SPS-Ausgang auf FALSE zu setzen, der durch diesen Befehl auf TRUE geschaltet hat, verwenden Sie einen RST oder MOVE-Befehl.
- Um die Ausführung eines Befehls abzubrechen, setzen Sie Bit 3 des Datenregisters, in dem der Steuercode für den schnellen Zähler (sys_wHscOrPulseControlCode) gespeichert wird, auf TRUE. Der Kontrollmerker "Schneller Zähler" wechselt dann zu FALSE. Um die Ausführung des schnellen Zählerbefehls wieder zu aktivieren, setzen Sie Bit 3 auf FALSE zurück.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F166_HighSpeedCounter_Set (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
n_diHscChannel	DINT	Kanalnummer des schnellen Zählers: FP-Σ: 0–3 FP-X R: 0–11 FP-X T: 0–7 FP0: 0–3 FP-e: 0–3 FP0R: 0–5

Variable	Datentyp	Funktion
s_diTargetValue	DINT	Geben Sie für den Sollwert einen 32-Bit-Datenwert innerhalb des folgenden Bereichs ein: FP0, FP-e: -838808–+8388607 FPΣ, FP-X, FP0R: -2147483467–+2147483648
d_Y	BOOL	Ausgang, der auf TRUE schaltet, wenn der Istwert dem Sollwert entspricht: FP-Σ, FP0, FP-e: Y0–Y7 FP-Σ (V3.1 oder eine neuere Version), FP0R: Y0–Y1F FP-X: Y0–Y29F

Operanden

Für	Merker				T/C		Register			Konstante
n_diHscChannel	-	-	-	-	-	-	-	-	-	dez. oder hex.
s_diTargetValue	DWX	DWY	DWR	-	DSV	DEV	DDT	-	-	-
d_Y	-	Y	-	-	-	-	-	-	-	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	ON	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	ON	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

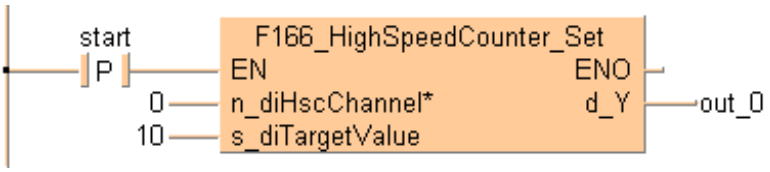
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial	Kommentar
0	VAR_GLOBAL	out_0	Y0	%QX0.0	BOOL	FALSE	output Y0 of PLC

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	out_0	BOOL	FALSE	output Y0 of PLC
1	VAR	start	BOOL	FALSE	start condition

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

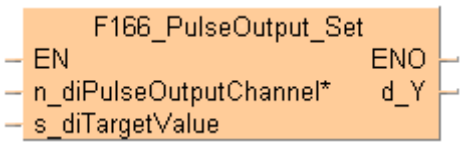
```
IF DF(start) THEN
      F166_HighSpeedCounter_Set(n_diHscChannel := 0,
      s_diTargetValue := 10,
      d_Y => out_0);
END_IF;
```

☞ **Weisen Sie den Eingangsvariablen Zahlenwerte zu (z.B. mit Monitor → Monitor Kopf, Variable anklicken, Wert eingeben, <Enter> drücken) oder ersetzen Sie die Eingangsvariablen durch Zahlen.**

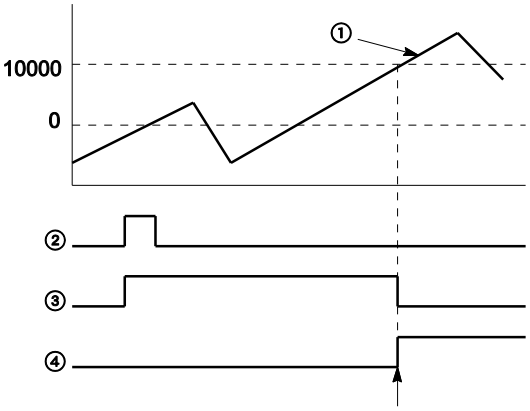
F166_PulseOutput_Set

Zählervergleichsausgang setzen (Pulsausgabe)

Erklärung Wenn der Istwert des gewählten Kanals den Sollwert erreicht, wird ein vorher bestimmter Ausgang sofort auf TRUE gesetzt.



Merkmale der Pulsausgabe



10000	Sollwert
①	Istwert der Pulsausgabe
②	Ausführungsbedingung
③	Merker "Auswertung aktiv"
④	SPS-Ausgang

Der SPS-Ausgang wird auf TRUE gesetzt, wenn der Istwert dem Sollwert entspricht. Außerdem wird der Merker "Auswertung aktiv" auf FALSE gesetzt und der Befehl deaktiviert.

Wenn ein Ausgang angegeben ist, der nicht implementiert wurde, wird nur die WY-Adresse im Speicher gesetzt oder zurückgesetzt.

Interrupt-Betrieb

Das Interrupt-Programm wird ausgeführt, wenn der Istwert dem Sollwert entspricht. Jedes Interrupt-Programm, das in die Taskliste eingegeben wird, ist automatisch aktiviert. Jeder Kanalnummer wird ein spezielles Interrupt-Programm zugewiesen.

Kanäle, die von Interrupt-Programmen verwendet werden:

Interrupt 8	Kanal 0
Interrupt 9	Kanal 1
Interrupt 10	Kanal 2
Interrupt 11	Kanal 3

■ Allgemeine Programmierinformation

- Setzen Sie den gewünschten Kanal im Systemregister auf "Pulsausgabe".
- Wenn dieser Befehl ausgeführt wird, schaltet der Merker "Auswertung aktiv" (sys_bIsPulseChannel0ControlActive) für den benutzten Kanal auf TRUE. Andere schnelle Zählerbefehle mit Ausgangssteuerung (F166_PulseOutput_Set oder

F167_PulseOutput_Reset), die denselben Kanal nutzen, können nicht ausgeführt werden, solange dieser Merker auf TRUE steht.

- Dieser Befehl kann vor oder nach einem beliebigen Pulsausgabebefehl, ausgenommen F173_PulseOutput_PWM (s. S. 925), ausgeführt werden.
- Die Benutzung eines externen Ausgangs in anderen Befehlen (OUT, SET, RST, KEEP, andere F-Befehle) wird vom System nicht überprüft (keine Erkennung einer Doppelbelegung).
- Um einen SPS-Ausgang auf FALSE zu setzen, der durch diesen Befehl auf TRUE geschaltet hat, verwenden Sie einen RST oder MOVE-Befehl.
- Wenn Sie Bit 2 des Datenregisters, in dem der Steuercode für die Pulsausgabe gespeichert wird (sys_wHscOrPulseControlCode), auf TRUE setzen, wird die Ausführung der Pulsausgabebefehle abgebrochen und der Kontrollmerker des schnellen Zählers auf FALSE gesetzt. Setzen Sie Bit 2 auf FALSE zurück, um die Ausführung der Befehle wieder zu aktivieren. Die Pulsausgabe wird jedoch fortgesetzt.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F166_PulseOutput_Set (s. S. 1188)**

Datentypen

Variable	Datentyp	Beschreibung
n_diPulseOutputChannel	DINT	Pulsausgabekanal 0–3
s_diTargetValue	DINT	Geben Sie für den Sollwert einen 32-Bit-Datenwert innerhalb des folgenden Bereichs ein: -2147483467–+2147483648
d_Y	BOOL	Ausgang, der auf TRUE schaltet, wenn der Istwert dem Sollwert entspricht: Y0–Y1F

Operanden

Für	Merker				T/C		Register			Konstante
n_diPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez. oder hex.
s_diTargetValue	DWX	DWY	DWR	-	DSV	DEV	DDT	-	-	-
d_Y	-	Y	-	-	-	-	-	-	-	-

Fehlermerker

Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	EIN	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	EIN	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

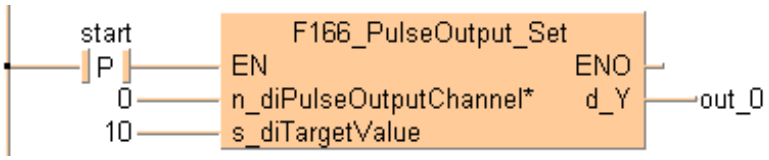
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial	Kommentar
0	VAR_GLOBAL	out_0	Y0	%QX0.0	BOOL	FALSE	output Y0 of PLC

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	out_0	BOOL	FALSE	output Y0 of PLC
1	VAR	start	BOOL	FALSE	start condition

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



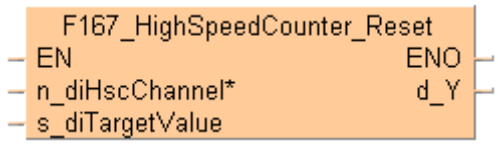
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
        F166_PulseOutput_Set(n_diPulseOutputChannel := 0,
          s_diTargetValue := 10,
          d_Y => out_0);
END_IF;
```

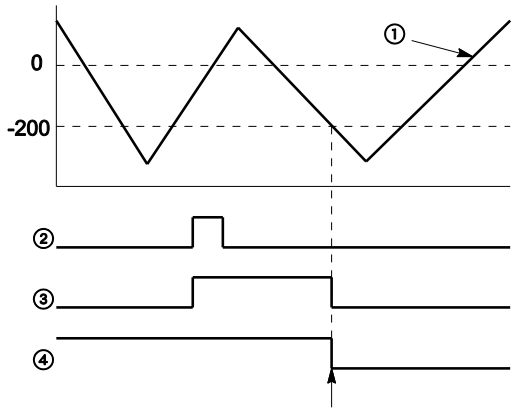
F167_HighSpeed Counter_Reset

Zählervergleichsausgang zurücksetzen (schneller Zähler)

Erklärung Wenn der Istwert eines schnellen Zählers dem Sollwert entspricht, schaltet ein Interrupt-Prozess den angegebenen Ausgang sofort auf FALSE.



Merkmale der Funktion "Zählervergleichsausgang zurücksetzen"



-200	Sollwert
①	Istwert des schnellen Zählers
②	Ausführungsbedingung
③	Kontrollmerker des schnellen Zählers
④	SPS-Ausgang

Der SPS-Ausgang schaltet auf FALSE, wenn der Istwert dem Sollwert entspricht. Außerdem wird der Kontrollmerker des schnellen Zählers auf FALSE gesetzt und der Befehl wird deaktiviert.

Wenn ein Ausgang angegeben ist, der nicht implementiert wurde, wird nur die WY-Adresse im Speicher gesetzt oder zurückgesetzt.

Interrupt-Betrieb

Das Interrupt-Programm wird ausgeführt, wenn der Istwert dem Sollwert entspricht. Jedes Interrupt-Programm, das in die Taskliste eingegeben wird, ist automatisch aktiviert. Jeder Kanalnummer wird ein spezielles Interrupt-Programm zugewiesen.

Kanäle, die von Interrupt-Programmen verwendet werden:

SPS-Typ	FP0, FP-e	FPΣ	FP-X (Relaistypen)	FP-X (Transistortypen)	FP0R
Interrupt 0	Kanal 0	Kanal 0	Kanal 0	Kanal 0	Kanal 0
Interrupt 1	Kanal 1	Kanal 1	Kanal 1	Kanal 1	Kanal 1
Interrupt 2			Kanal 2	Kanal 2	
Interrupt 3	Kanal 2	Kanal 2	Kanal 3	Kanal 3	Kanal 2
Interrupt 4	Kanal 3	Kanal 3	Kanal 4	Kanal 4	Kanal 3
Interrupt 5			Kanal 5	Kanal 5	
Interrupt 6			Kanal 6	Kanal 6	Kanal 4
Interrupt 7			Kanal 7	Kanal 7	Kanal 5
Interrupt 8			Kanal 8		
Interrupt 9			Kanal 9		
Interrupt 10					
Interrupt 11			Kanal A		
Interrupt 12			Kanal B		

Allgemeine Programmierinformation

- Wählen Sie den Eingang des schnellen Zählers für den gewünschten Kanal in den Systemregistern aus.
- FP-X, FP0R: Wenn dieser Befehl ausgeführt wird, schaltet der Kontrollmerker für den benutzten Kanal (z.B. sys_bIsHscChannel0ControlActive) auf TRUE. Andere schnelle Zählerbefehle, die denselben Kanal nutzen, können nicht ausgeführt werden, solange der Kontrollmerker auf TRUE steht.
- FP0, FP-e, FPΣ: Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Um einen SPS-Ausgang auf FALSE zu setzen, der durch diesen Befehl auf TRUE geschaltet hat, verwenden Sie einen RST oder MOVE-Befehl.
- Um die Ausführung eines Befehls abzubrechen, setzen Sie Bit 3 des Datenregisters, in dem der Steuercode für den schnellen Zähler (sys_wHscOrPulseControlCode) gespeichert wird, auf TRUE. Der Kontrollmerker "Schneller Zähler" wechselt dann zu FALSE. Um die Ausführung des schnellen Zählerbefehls wieder zu aktivieren, setzen Sie Bit 3 auf FALSE zurück.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F167_HighSpeedCounter_Reset (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
n_diHscChannel	DINT	Kanalnummer des schnellen Zählers: FP-Σ: 0–3 FP-X R: 0–11 FP-X T: 0–7 FP0: 0–3 FP-e: 0–3 FP0R: 0–5
s_diTargetValue	DINT	Geben Sie für den Sollwert einen 32-Bit-Datenwert innerhalb des folgenden Bereichs ein: FP0, FP-e: -838808–+8388607 FPΣ, FP-X, FP0R: -2147483467–+2147483648
d_Y	BOOL	Ausgang, der auf FALSE schaltet, wenn der Istwert dem Sollwert entspricht: FP-Σ, FP0, FP-e: Y0–Y7 FP-Σ (V3.1 oder eine neuere Version), FP0R: Y0–Y1F FP-X: Y0–Y29F

Operanden

Für	Merker				T/C		Register			Konstante
n_diHscChannel	-	-	-	-	-	-	-	-	-	dez. oder hex.
s_diTargetValue	DWX	DWY	DWR	-	DSV	DEV	DDT	-	-	-
d_Y	-	Y	-	-	-	-	-	-	-	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	ON	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	ON	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

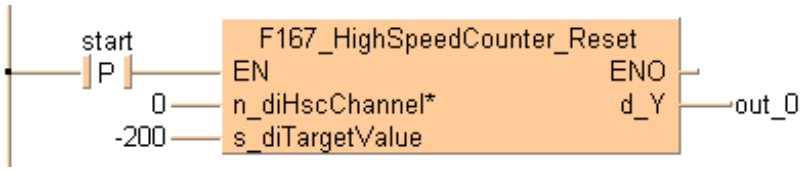
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial	Kommentar
0	VAR_GLOBAL	out_0	Y0	%QX0.0	BOOL	FALSE	output Y0 of PLC

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	out_0	BOOL	FALSE	output Y0 of PLC
1	VAR	start	BOOL	FALSE	start condition

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

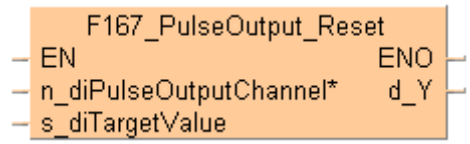
```
IF DF(start) THEN
      F167_HighSpeedCounter_Reset(n_diHscChannel := 0,
    s_diTargetValue := -200,
    d_Y => out_0);
END_IF;
```

☞ **Weisen Sie den Eingangsvariablen Zahlenwerte zu (z.B. mit Monitor → Monitor Kopf, Variable anklicken, Wert eingeben, <Enter> drücken) oder ersetzen Sie die Eingangsvariablen durch Zahlen.**

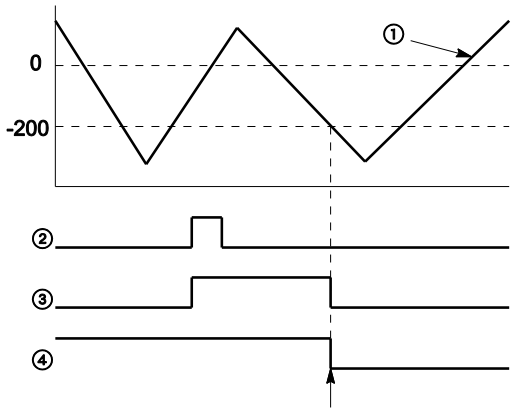
F167_PulseOutput_Reset

Zählervergleichsausgang zurücksetzen (Pulsausgabe)

Erklärung Wenn der Istwert des gewählten Kanals den Sollwert erreicht, wird ein vorher bestimmter Ausgang sofort auf FALSE gesetzt.



Merkmale der Pulsausgabe



-200	Sollwert
①	Istwert der Pulsausgabe
②	Ausführungsbedingung
③	Merker "Auswertung aktiv"
④	SPS-Ausgang

Der SPS-Ausgang schaltet auf FALSE, wenn der Istwert dem Sollwert entspricht. Außerdem wird der Merker "Auswertung aktiv" auf FALSE gesetzt und der Befehl deaktiviert.

Wenn ein Ausgang angegeben ist, der nicht implementiert wurde, wird nur die WY-Adresse im Speicher gesetzt oder zurückgesetzt.

Interrupt-Betrieb

Das Interrupt-Programm wird ausgeführt, wenn der Istwert dem Sollwert entspricht. Jedes Interrupt-Programm, das in die Taskliste eingegeben wird, ist automatisch aktiviert. Jeder Kanalnummer wird ein spezielles Interrupt-Programm zugewiesen.

Interrupt 8	Kanal 0
Interrupt 9	Kanal 1
Interrupt 10	Kanal 2
Interrupt 11	Kanal 3

■ Allgemeine Programmierinformation

- Setzen Sie den gewünschten Kanal im Systemregister auf "Pulsausgabe".
- Wenn dieser Befehl ausgeführt wird, schaltet der Merker "Auswertung aktiv" (sys_bIsPulseChannel0ControlActive) für den benutzten Kanal auf TRUE. Andere schnelle Zählerbefehle mit Ausgangssteuerung (F166_PulseOutput_Set oder F167_PulseOutput_Reset), die denselben Kanal nutzen, können nicht ausgeführt

werden, solange dieser Merker auf TRUE steht.

- Dieser Befehl kann vor oder nach einem beliebigen Pulsausgabebefehl, ausgenommen F173_PulseOutput_PWM (s. S. 925), ausgeführt werden.
- Die Benutzung eines externen Ausgangs in anderen Befehlen (OUT, SET, RST, KEEP, andere F-Befehle) wird vom System nicht überprüft (keine Erkennung einer Doppelbelegung).
- Wenn Sie Bit 2 des Datenregisters, in dem der Steuercode für die Pulsausgabe gespeichert wird (sys_wHscOrPulseControlCode), auf TRUE setzen, wird die Ausführung der Pulsausgabebefehle abgebrochen und der Kontrollmerker des schnellen Zählers auf FALSE gesetzt. Setzen Sie Bit 2 auf FALSE zurück, um die Ausführung der Befehle wieder zu aktivieren. Die Pulsausgabe wird jedoch fortgesetzt.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F167_PulseOutput_Reset (s. S. 1188)**

Datentypen

Variable	Datentyp	Beschreibung
n_diPulseOutputChannel	DINT	Pulsausgabekanal 0–3
s_diTargetValue	DINT	Geben Sie für den Sollwert einen 32-Bit-Datenwert innerhalb des folgenden Bereichs ein: -2147483467–+2147483648
d_Y	BOOL	Ausgang, der auf FALSE schaltet, wenn der Istwert dem Sollwert entspricht: Y0–Y1F

Operanden

Für	Merker				T/C		Register			Konstante
n_diPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez. oder hex.
s_diTargetValue	DWX	DWY	DWR	-	DSV	DEV	DDT	-	-	-
d_Y	-	Y	-	-	-	-	-	-	-	-

Fehlermerker

Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	EIN	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	EIN	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

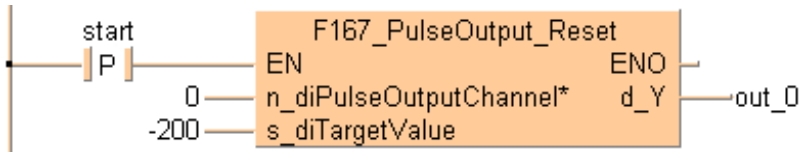
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial	Kommentar
0	VAR_GLOBAL	out_0	Y0	%QX0.0	BOOL	FALSE	output Y0 of PLC

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	out_0	BOOL	FALSE	output Y0 of PLC
1	VAR	start	BOOL	FALSE	start condition

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



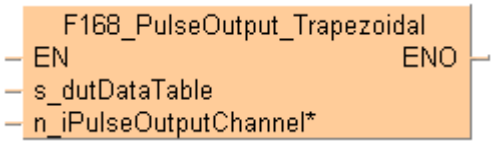
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(start) THEN
        F167_PulseOutput_Reset(n_diPulseOutputChannel := 0,
          s_diTargetValue := -200,
          d_Y => out_0);
END_IF;
```

F168_PulseOutput_Trapezoidal

AUTO-TRAPEZ-Funktion

Erklärung Anhand der Parameter im angegebenen strukturierten Datentyp wird eine AUTO-TRAPEZ-Steuerung durchgeführt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.

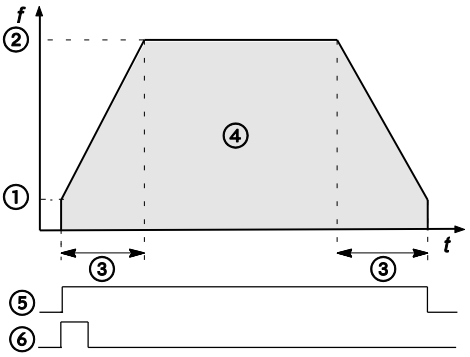


Verwenden Sie folgenden, vordefinierten strukturierten Datentyp:

F168_PulseOutput_Trapezoidal_DUT

- Steuercode
- Anfangs- und Endgeschwindigkeit
- Sollgeschwindigkeit
- Beschleunigungs-/Bremszeit
- Sollwert
- Ende der Pulsausgabe (fest):

Merkmale der Pulsausgabe



①	Anfangs- und Endgeschwindigkeit	④	Sollwert
②	Sollgeschwindigkeit	⑤	Kontrollmerker für Pulsausgabe
③	Beschleunigungs-/Bremszeit	⑥	Ausführungsbedingung

Die Pulsausgabefrequenz ändert sich entsprechend der festgelegten Beschleunigungs- und Bremszeit.

Der Unterschied zwischen der Soll- und Anfangsgeschwindigkeit bestimmt die Steigung der Rampen.

Allgemeine Programmierinformation

- Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- Die Pulsausgabe wird während des Online-Editierens gestoppt und erst nach der Übertragung der Programmänderungen fortgesetzt.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive)

und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.

- Bleibt die Drehrichtung immer gleich, kann es passieren, dass die Zählerobergrenze erreicht wird. Die Pulsausgabe wird dann gestoppt. Als Gegenmaßnahme setzen Sie den Istwert auf 0 zurück, ehe Sie diesen Befehl ausführen. Die Pulsausgabe stoppt nicht, wenn die FP0R im FP0-Kompatibilitätsmodus verwendet wird, da der Datenbereich für den Istwert ein vorzeichenbehafteter 32-Bit-Wert ist.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

■ Verwendung des FP0-Kompatibilitätsmodus der FP0R

Um die FP0R im FP0-Kompatibilitätsmodus auszuführen, können Sie ein FP0-Programm auf die FP0R übertragen. Bitte beachten Sie folgende Beschränkungen:

- Die FP0R unterstützt vorzeichenbehaftete 32-Bit-Daten für Ist- und Sollwerte; die FP0 unterstützt vorzeichenbehaftete 24-Bit-Daten. Auf der FP0 zählen Zähler und Pulsausgabe auch über den FP0-Bereich hinaus weiter.
- Das Puls-Pausenverhältnis beträgt immer 25%, unabhängig von den Befehlseinstellungen. Bei der Verwendung der Methode Pulsausgabe/Richtungsanzeige werden die Pulse ca. 300µs nach der Ausgabe des Richtungsanzeigesignals ausgegeben; wobei das Verhalten eines Motorantriebs berücksichtigt wird.
- Die FP0R unterstützt die Einstellung "Nicht gezählt" nicht. Bestimmte FP0-Befehle bieten die Einstellmöglichkeit "Nicht zählen". Die FP0R unterstützt diese Einstellung nicht. Sie wird auf der FP0R als "Vorwärtszählen" interpretiert.
- Die maximale Frequenz für die Pulsausgabe ist 10000Hz.
- Stellen Sie sicher, dass der Pulsausgabebefehl keinen Ausgang verwendet, der als normaler Ausgang genutzt wird.
- Um ein FP0-Programm im FP0-Kompatibilitätsmodus auszuführen, müssen die SPS-Typen (C10, C14, C16, C32, T32) exakt übereinstimmen. Der FP0-Kompatibilitätsmodus steht für die FP0R, Typ F32 nicht zur Verfügung

SPS-Typen **Verfügbarkeit von F168_PulseOutput_Trapezoidal (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_dutDataTable	F168_PulseOutput_Trapezoidal_DUT	Startadresse des Bereichs, der die Datentabelle enthält
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgang: 0 oder 1

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ Anfangsgeschwindigkeit < 40 ▪ Anfangsgeschwindigkeit > Maximalgeschwindigkeit
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	X0_bMotorSwitch	X0	%IX0.0	BOOL	FALSE

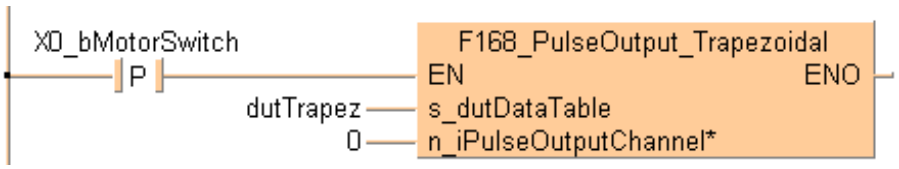
SDT Der SDT F168_PulseOutput_Trapezoidal_DUT ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	dutTrapez	F168_PulseOutput_Trapezoidal_DUT	wControlCode := 16#102, iInitialAndFinalSpeed := 1000, iTargetSpeed := 7000, iAccelerationAndDecelerationTime := 300, diTargetValue := 10000	
1	VAR_EXTERNAL	X0_bMotorSwitch	BOOL		
2	VAR				
3	VAR				

Rumpf Wenn **X0_bMotorSwitch** auf TRUE schaltet, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

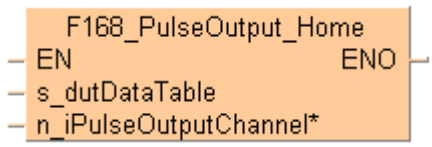
```
IF DF(X0_bMotorSwitch) THEN
        F168_PulseOutput_Trapezoidal(s_dutDataTable := dutTrapez,
        n_iPulseOutputChannel := 0);
END_IF;
```

F168_PulseOutput_Home

Referenzpunktfahrt

Erklärung

Anhand der Parameter im angegebenen strukturierten Datentyp wird eine Referenzpunktfahrt durchgeführt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Nach dem Einschalten des Antriebssystems besteht ein vorher nicht bestimmbarer Versatz zwischen dem internen Positionswert (Istwert) und der mechanischen Position der Achse. Zur Herstellung des Positionsbezuges muss der interne Wert mit dem realen Positionswert der Achse synchronisiert werden. Die Synchronisation erfolgt durch Übernahme eines Positionswertes an einem bekannten Punkt (Referenzpunkt).

Bei der Ausführung eines Referenzpunktfahrtbefehls werden so lange Pulse ausgegeben, bis der Referenzpunkteingang aktiviert wird. Die E/A-Zuweisung richtet sich nach dem verwendeten Kanal.

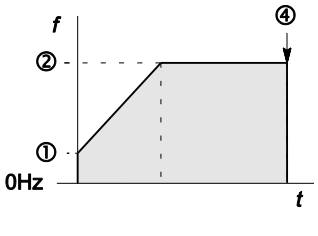
Zum Abbremsen im Referenzpunktbereich geben Sie einen Referenzpunkteingang an und setzen Bit 4 des Sonderdatenregisters, in dem der Steuercode für die Pulsausgabe gespeichert wird (sys_wHscOrPulseControlCode), auf TRUE und zurück auf FALSE.

Während der Referenzpunktfahrt unterscheiden sich der Wert im Istwert-Speicherbereich und der tatsächliche Istwert. Wenn die Referenzpunktfahrt abgeschlossen ist, springt der Istwert auf 0.

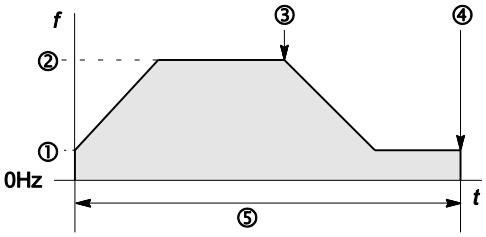
Es gibt zwei verschiedene Betriebsarten:

- Typ 1: Der Referenzpunkteingang wird aktiviert, unabhängig davon, ob ein Referenzpunkt-Sucheingang vorhanden ist, ob der Bremsvorgang bereits eingesetzt hat oder ob der Bremsvorgang abgeschlossen ist.

Ohne Referenzpunkteingang:

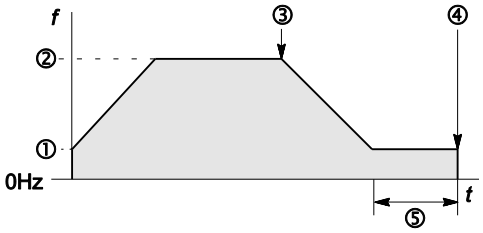


Mit Referenzpunkteingang:



①	Anfangs- und Endgeschwindigkeit	③	Referenzpunkteingang: TRUE
②	Sollgeschwindigkeit	④	Referenzpunkteingang: TRUE
⑤	Referenzpunkteingang jederzeit aktivierbar		

- Typ 2: Der Referenzpunkteingang kann nur aktiviert werden, nachdem der Bremsvorgang (ausgelöst durch einen Referenzpunkt-Sucheingang) abgeschlossen ist.



①	Anfangs- und Endgeschwindigkeit	③	Referenzpunkteingang: TRUE

②	Sollgeschwindigkeit	④	Referenzpunkteingang: TRUE
⑤	Referenzpunktfahrt erst aktivierbar, wenn Bremsvorgang abgeschlossen		

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: F168_PulseOutput_Home_DUT

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Anfangs- und Endgeschwindigkeit
- Sollgeschwindigkeit
- Beschleunigungs-/Bremszeit
- Ende der Pulsausgabe (fest)

Merkmale der Pulsausgabe

- Die Pulsausgabefrequenz ändert sich entsprechend der festgelegten Beschleunigungs- und Bremszeit.
- Der Unterschied zwischen der Soll- und Anfangsgeschwindigkeit bestimmt die Steigung der Rampen.

Allgemeine Programmierinformation

- Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- Die Pulsausgabe wird während des Online-Editierens gestoppt und erst nach der Übertragung der Programmänderungen fortgesetzt.
- Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- Auch wenn der Referenzpunkt erreicht ist, werden durch die Ausführung dieses Befehls Pulse ausgegeben.
- Wenn der Referenzpunkteingang während der Beschleunigung aktiviert wird, beginnt die Abbremsung.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

■ Verwendung des FP0-Kompatibilitätsmodus der FP0R

Um die FP0R im FP0-Kompatibilitätsmodus auszuführen, können Sie ein FP0-Programm auf die FP0R übertragen. Bitte beachten Sie folgende Beschränkungen:

- Die FP0R unterstützt vorzeichenbehaftete 32-Bit-Daten für Ist- und Sollwerte; die FP0 unterstützt vorzeichenbehaftete 24-Bit-Daten. Auf der FP0 zählen Zähler und Pulsausgabe auch über den FP0-Bereich hinaus weiter.

- Das Puls-Pausenverhältnis beträgt immer 25%, unabhängig von den Befehlseinstellungen. Bei der Verwendung der Methode Pulsausgabe/Richtungsanzeige werden die Pulse ca. 300µs nach der Ausgabe des Richtungsanzeigesignals ausgegeben; wobei das Verhalten eines Motorantriebs berücksichtigt wird.
- Die FP0R unterstützt die Einstellung "Nicht gezählt" nicht. Bestimmte FP0-Befehle bieten die Einstellmöglichkeit "Nicht zählen". Die FP0R unterstützt diese Einstellung nicht. Sie wird auf der FP0R als "Vorwärtszählen" interpretiert.
- Die maximale Frequenz für die Pulsausgabe ist 10000Hz.
- Stellen Sie sicher, dass der Pulsausgabebefehl keinen Ausgang verwendet, der als normaler Ausgang genutzt wird.
- Um ein FP0-Programm im FP0-Kompatibilitätsmodus auszuführen, müssen die SPS-Typen (C10, C14, C16, C32, T32) exakt übereinstimmen. Der FP0-Kompatibilitätsmodus steht für die FP0R, Typ F32 nicht zur Verfügung

SPS-Typen **Verfügbarkeit von F168_PulseOutput_Home (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_dutDataTable	F168_PulseOutput_Home_DUT	Startadresse des Bereichs, der die Datentabelle enthält
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgang: 0 oder 1

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ Anfangsgeschwindigkeit < 40 ▪ Anfangsgeschwindigkeit > Maximalgeschwindigkeit
R9008	%MX0.900.8	kurzzeitig	

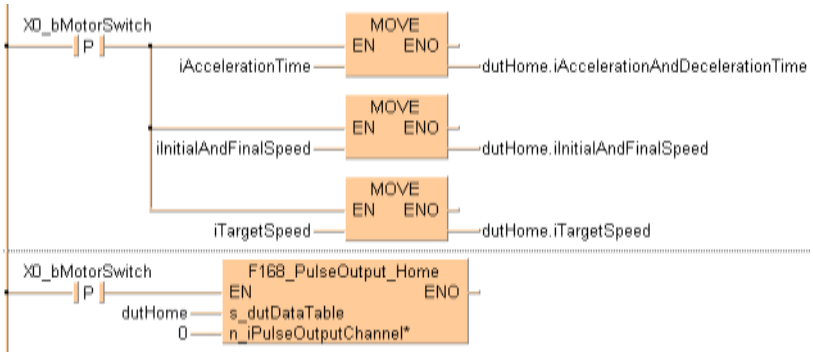
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT Der SDT F168_PulseOutput_Home_DUT ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	dutHome	F168_PulseOutput_Home_DUT	wControlCode := 16#102,
1	VAR	iInitialAndFinalSpeed	INT	3000
2	VAR	iTargetSpeed	INT	7000
3	VAR	iAccelerationTime	INT	300

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

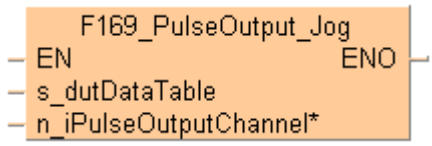
```
IF DF(X0_bMotorSwitch) THEN
    dutHome.iInitialAndFinalSpeed:=iInitialAndFinalSpeed
    dutHome.iTargetSpeed:=iTargetSpeed
    dutHome.iAccelerationAndDecelerationTime:=iAccelerationTime
END_IF;

IF DF(X0_bMotorSwitch) THEN
F168_PulseOutput_Home(s_dutDataTable:=dutHome,
    n_iPulseOutputChannel:=0);
END_IF;
```

F169_PulseOutput_Jog

Tipp-Betrieb

Erklärung

Dieser Befehl wird für den Tipp-Betrieb verwendet. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: F169_PulseOutput_Jog_DUT

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Geschwindigkeit

Merkmale der Pulsausgabe

Die Frequenz und das Puls-Pausenverhältnis können in jedem SPS-Zyklus geändert werden. (Die Änderung wird bei der nächsten Pulsausgabe wirksam.)

Allgemeine Programmierinformation

Achtung!

Wenn Sie Programme, die diesen Befehl verwenden, im RUN-Modus bearbeiten (Online-Editieren), wird die Pulsausgabe beendet.

- Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- Bleibt die Drehrichtung immer gleich, kann es passieren, dass die Zählerobergrenze erreicht wird. Die Pulsausgabe wird dann gestoppt. Als Gegenmaßnahme setzen Sie den Istwert auf 0 zurück, ehe Sie diesen Befehl ausführen. Die Pulsausgabe stoppt nicht, wenn die FP0R im FP0-Kompatibilitätsmodus verwendet wird, da der Datenbereich für den Istwert ein vorzeichenbehafteter 32-Bit-Wert ist.
- Beim Vorwärtszählen wird die Pulsausgabe beendet, wenn der Istwert den Wert 2147483647 übersteigt.
- Beim Rückwärtszählen wird die Pulsausgabe beendet, wenn der Istwert den Wert -2147483648 unterschreitet.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s.

S. 865) in Ihrem Positionierprogramm bereit zu stellen.

- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

■ Verwendung des FP0-Kompatibilitätsmodus der FP0R

Um die FP0R im FP0-Kompatibilitätsmodus auszuführen, können Sie ein FP0-Programm auf die FP0R übertragen. Bitte beachten Sie folgende Beschränkungen:

- Die FP0R unterstützt vorzeichenbehaftete 32-Bit-Daten für Ist- und Sollwerte; die FP0 unterstützt vorzeichenbehaftete 24-Bit-Daten. Auf der FP0 zählen Zähler und Pulsausgabe auch über den FP0-Bereich hinaus weiter.
- Das Puls-Pausenverhältnis beträgt immer 25%, unabhängig von den Befehlseinstellungen. Bei der Verwendung der Methode Pulsausgabe/Richtungsanzeige werden die Pulse ca. 300µs nach der Ausgabe des Richtungsanzeigesignals ausgegeben; wobei das Verhalten eines Motorantriebs berücksichtigt wird.
- Die FP0R unterstützt die Einstellung "Nicht gezählt" nicht. Bestimmte FP0-Befehle bieten die Einstellmöglichkeit "Nicht zählen". Die FP0R unterstützt diese Einstellung nicht. Sie wird auf der FP0R als "Vorwärtszählen" interpretiert.
- Die maximale Frequenz für die Pulsausgabe ist 10000Hz.
- Stellen Sie sicher, dass der Pulsausgabebefehl keinen Ausgang verwendet, der als normaler Ausgang genutzt wird.
- Um ein FP0-Programm im FP0-Kompatibilitätsmodus auszuführen, müssen die SPS-Typen (C10, C14, C16, C32, T32) exakt übereinstimmen. Der FP0-Kompatibilitätsmodus steht für die FP0R, Typ F32 nicht zur Verfügung

SPS-Typen **Verfügbarkeit von F169_PulseOutput_Jog (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_dutDataTable	F169_PulseOutput_Jog_DUT	Startadresse des Bereichs, der die Datentabelle enthält
n_iPulseOutputChannel	INT	Pulsausgang: 0 oder 1

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	X0_bMotorSwitch	X0	%IX0.0	BOOL	FALSE

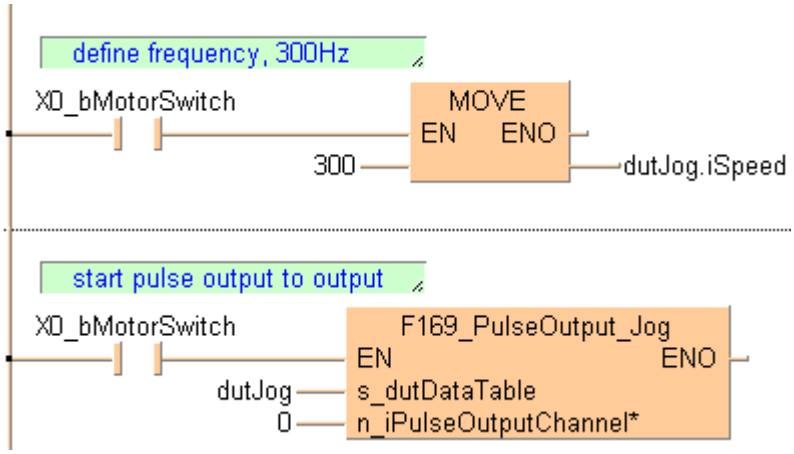
SDT Der SDT F169_PulseOutput_Jog_DUT ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	X0_bMotorSwitch	BOOL	FALSE	
1	VAR	dut_Jog	F169_PulseOutput_Jog_DUT	wControlCode := 16#110,	Digit2: 1= Duty 10%
2	VAR			iSpeed := 0	Digit1: 1=Incremental counting Digit0: 0=No direction output

Rumpf Die Kommentarfelder erläutern die Funktion dieses Beispiels.

KOP



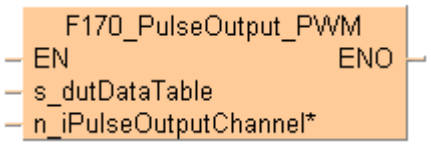
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF (X0_bMotorSwitch) THEN
      dutJog.ispeed := 300;
END_IF;
IF (X0_bMotorSwitch) THEN
      F169_PulseOutput_Jog(s_dutDataTable := dutJog,
  n_iPulseOutputChannel := 0);
END_IF;
```

F170_PulseOutput_PWM

Pulsausgabe mit Angabe des Kanals

Erklärung Dieser Befehl liefert ein pulsweitenmoduliertes Ausgangssignal. Die Parameter für die Pulsausgabe werden in einem SDT festgelegt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: F170_PulseOutput_PWM_DUT

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Ungefähre Frequenz
- Puls-Pausenverhältnis (der Pulsdauer und -periode)

Allgemeine Programmierinformation

Achtung!

Wenn Sie Programme, die diesen Befehl verwenden, im RUN-Modus bearbeiten (Online-Editieren), wird die Pulsausgabe beendet.

- Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- Bei einem Puls-Pausenverhältnis, das nahe am Minimal- oder Maximalwert liegt, wird die Pulsausgabe verzögert. Durch diese Verzögerung kann das Puls-Pausenverhältnis vom festgelegten Wert abweichen.
- Das Puls-Pausenverhältnis kann in jedem SPS-Zyklus geändert werden. (Die Änderung wird bei der nächsten Pulsfrequenzperiode wirksam.) Der Frequenzwert wird nur zu Beginn der Befehlsausführung gelesen.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

■ FP0-Kompatibilitätsmodus der FP0R verwenden

Um die FP0R im FP0-Kompatibilitätsmodus auszuführen, können Sie ein FP0-Programm auf die FP0R übertragen

SPS-Typen **Verfügbarkeit von F170_PulseOutput_PWM (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_dutDataTable	F170_PulseOutput_PWM_DUT	Startadresse des Bereichs, der die Datentabelle enthält
n_iPulseOutputChannel	INT	Pulsausgabekanal: 0 oder 1

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

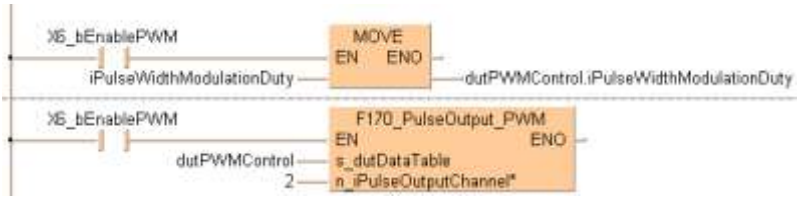
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	X6_bEnablePWM	X6	%IX0.6	BOOL	FALSE

SDT Der SDT F170_PulseOutput_PWM_DUT ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	X6_bEnablePWM	BOOL	FALSE	
1	VAR	dutPWMControl	F170_PulseOutput_PWM_DUT		iFrequencyValue := 1: f=2.0Hz, T=502.5m
2	VAR	iPulseWidthModulationDuty	INT	500	500=50% duty

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

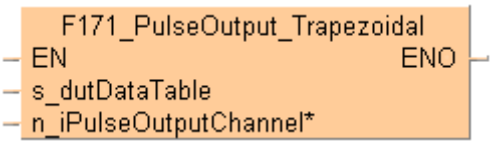
```
IF (X6_bEnablePWM) THEN
      dutPWMControl.iPulseWidthModulationDuty:=iPulseWidthModulationDuty;
END_IF;
IF (X6_bEnablePWM) THEN
      F170_PulseOutput_PWM(s_dutDataTable := dutPWMControl,
    n_iPulseOutputChannel := 2);
END_IF;
```

F171_PulseOutput_ Trapezoidal

AUTO-TRAPEZ-Funktion

Erklärung

Anhand der Parameter im angegebenen strukturierten Datentyp wird eine AUTO-TRAPEZ-Steuerung durchgeführt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



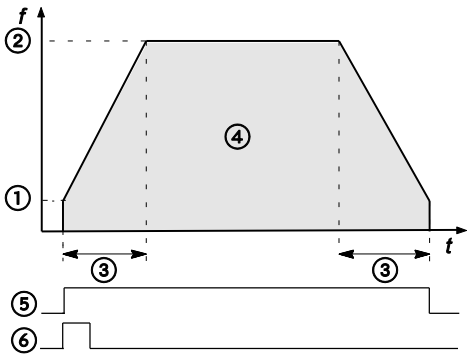
■ Beschreibung für FP-Sigma, FP-X (für die FP0R siehe auf Seite 906)

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: F171_PulseOutput_Trapezoidal_DUT.

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Anfangs- und Endgeschwindigkeit
- Sollgeschwindigkeit
- Beschleunigungs-/Bremszeit
- Sollwert
- Ende der Pulsausgabe

Merkmale der Pulsausgabe



①	Anfangs- und Endgeschwindigkeit	④	Sollwert
②	Sollgeschwindigkeit	⑤	Kontrollmerker für Pulsausgabe
③	Beschleunigungs-/Bremszeit	⑥	Ausführungsbedingung

Die Pulsausgabefrequenz ändert sich entsprechend der festgelegten Beschleunigungs- und Bremszeit.

Der Unterschied zwischen der Soll- und Anfangsgeschwindigkeit bestimmt die Steigung der Rampen.

Allgemeine Programmierinformation

- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- FP-X: Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- FPΣ: Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- FPΣ: Wenn der Kreisinterpolationsbefehl **F176** ausgeführt wird, wird der Kontrollmerker (sys_bIsCircularInterpolationActive) auf TRUE gesetzt. Der Status dieses Merkers bleibt erhalten, bis der Sollwert erreicht ist (auch wenn die Ausführungsbedingung nicht mehr TRUE ist). In dieser Zeit lassen sich keine anderen Pulsausgabebefehle ausführen.
- Die Pulsausgabe wird während des Online-Editierens gestoppt und erst nach der Übertragung der Programmänderungen fortgesetzt.
- FPΣ: Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- FP-X: Setzen Sie den gewünschten Kanal im Systemregister auf "Pulsausgabe".
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

◆REFERENZ

Hinweise zur Verwendung von Systemvariablen finden Sie in der Online-Hilfe von FPWIN Pro.

- Ausgänge und Systemvariablen für FP-Sigma
- Ausgänge und Systemvariablen für FP-X-Relaistypen
- Ausgänge und Systemvariablen für FP-X-Transistortypen

■ Beschreibung für FP0R

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp:
F171_PulseOutput_Trapezoidal_Type0_DUT (Maximalgeschwindigkeit = erste Sollgeschwindigkeit) oder
F171_PulseOutput_Trapezoidal_Type1_DUT (Maximalgeschwindigkeit = 50kHz).

Die Sollgeschwindigkeit lässt sich während der Pulsausgabe ändern. Es gibt zwei verschiedene Betriebsarten:

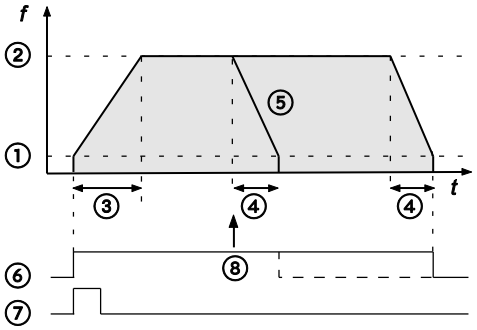
- Typ 0: Die Geschwindigkeit lässt sich nur innerhalb des zuvor festgelegten Bereichs für die Sollgeschwindigkeit ändern.
- Typ 1: Die Geschwindigkeit lässt sich nur innerhalb des zuvor festgelegten

Bereichs für die Sollgeschwindigkeit ändern (50kHz).

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Anfangs- und Endgeschwindigkeit
- Sollgeschwindigkeit
- Beschleunigungszeit
- Bremszeit
- Sollwert

Merkmale der Pulsausgabe



①	Anfangs- und Endgeschwindigkeit	⑤	Sollwert
②	Sollgeschwindigkeit	⑥	Kontrollmerker für Pulsausgabe
③	Beschleunigungszeit	⑦	Ausführungsbedingung
④	Bremszeit	⑧	Signal für gebremsten Halt

Typ 0: Die Differenz zwischen Soll- und Anfangsgeschwindigkeit bestimmt die Steigung der Bremsrampe. Die Differenz zwischen Soll- und Endgeschwindigkeit bestimmt die Steigung der Bremsrampe.

Typ 1: Die Differenz zwischen der Maximalgeschwindigkeit von 50kHz und der Anfangsgeschwindigkeit bestimmt die Steigung der Beschleunigungsrampe. Die Differenz zwischen der Maximalgeschwindigkeit von 50kHz und der Endgeschwindigkeit bestimmt die Steigung der Bremsrampe.

Die Pulse werden mit einem Puls-Pausenverhältnis von 25% ausgegeben.

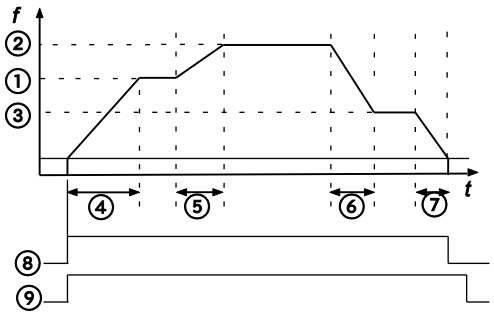
Bei der Verwendung der Methode Pulsausgabe/Richtungsanzeige werden die Pulse ca. 300µs nach der Ausgabe des Richtungsanzeigesignals ausgegeben; wobei das Verhalten eines Motorantriebs berücksichtigt wird.

Gebremster Halt

Um einen abgebremsten Stopp auszuführen, setzen Sie Bit 5 des Datenregisters, in dem der Steuercode für die Pulsausgabe gespeichert wird auf FALSE → TRUE → FALSE (z.B. `MOVE(16#120,sys_wHscOrPulseControlCode);`).

Wenn während der Beschleunigung das Signal für einen gebremsten Halt ausgelöst wird, wird der Bremsvorgang mit derselben Geschwindigkeit ausgeführt wie der normale Bremsvorgang, der von der Sollgeschwindigkeit ausgeht.

Sollgeschwindigkeit während der Pulsausgabe ändern



Typ 1: Die Geschwindigkeit lässt sich nur innerhalb des zuvor festgelegten Bereichs für die Sollgeschwindigkeit ändern (50kHz).

①	Sollgeschwindigkeit	⑥	Abbremsung
②	1. Änderung der Sollgeschwindigkeit	⑦	Bremszeit
③	2. Änderung der Sollgeschwindigkeit	⑧	Kontrollmerker für Pulsausgabe
④	Beschleunigungszeit	⑨	Ausführungsbedingung
⑤	Beschleunigung		

Die Geschwindigkeit kann nur geändert werden, wenn die Ausführungsbedingung TRUE ist.

Typ 0: Wenn zu Beginn ein höherer Wert als die Sollgeschwindigkeit angegeben ist, wird diese Angabe korrigiert und stattdessen die Sollgeschwindigkeit verwendet.
Typ 1: Wenn für die Sollgeschwindigkeit ein höherer Wert als 50kHz festgelegt wurde, erfolgt eine Korrektur auf 50kHz.

Wenn der Istwert zu Beginn der Beschleunigung im nicht zu lässigen Bereich liegt (z.B. sys_diPulseChannel0AccelerationForbiddenAreaStartingPosition), kann die Beschleunigung nicht ausgeführt werden.

Die Bremsgeschwindigkeit kann nicht unter der korrigierten Restgeschwindigkeit liegen.

Allgemeine Programmierinformation

Achtung!

Wenn Sie Programme, die diesen Befehl verwenden, im RUN-Modus bearbeiten (Online-Editieren), wird die Pulsausgabe beendet.

- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- Der Befehl kann nicht initiiert werden, wenn ein abgebremster Stopp angefordert wurde.
- Um nach dem Stopp des Betriebs einen Neustart auszuführen, muss die Ausführungsbedingung zunächst auf FALSE und dann wieder auf TRUE gesetzt werden.
- Die Ausführung der Anweisung erfolgt beim zweiten Mal nach dem Start schneller

wenn die Positionierungsparameter unverändert sind. Dieses Verhalten bleibt bestehen, auch wenn Sie die Freigabe der Ausgänge (Pulsausgabe oder Simulation) ändern.

- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F171_PulseOutput_Trapezoidal (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_dutDataTable	**FP-Σ, FP-X:** F171_PulseOutput_Trapezoidal_DUT **FP0R:** F171_PulseOutput_Trapezoidal_Type0_DUT F171_PulseOutput_Trapezoidal_Type1_DUT	Startadresse des Bereichs, der die Datentabelle enthält
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgabekanal FP-Σ: 0, 2 FP-X R: 0, 1 FP-X C14T: 0, 1, 2 FP-X C30T/C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ die Anfangsgeschwindigkeit > Sollgeschwindigkeit ist ▪ FP0R/FP-X:die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	kurzzeitig	

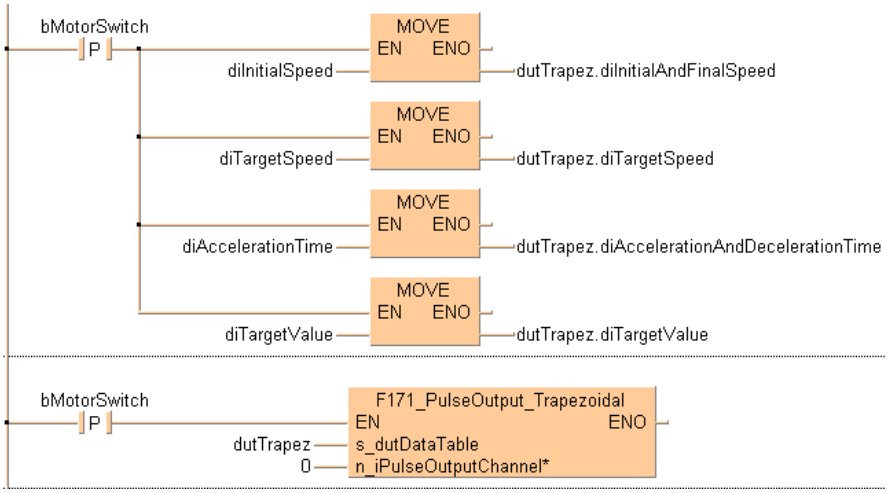
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT Der SDT F171_PulseOutput_Trapezoidal_DUT ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	X0_bMotorSwitch	BOOL	FALSE	at X0
1	VAR	bMotorSwitch	BOOL	FALSE	
2	VAR	dutTrapez	F171_PulseOutput_Trapezoidal_DUT	dwControlCode := 16#1100	Control code 16#1100= 1 = 25% Duty 1 = 48Hz to 100kHz 00 = Incremental CW/CCW
3	VAR	diInitialSpeed	DINT	100	
4	VAR	diTargetSpeed	DINT	2000	
5	VAR	diAccelerationTime	DINT	300	
6	VAR	diTargetValue	DINT	10000	

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

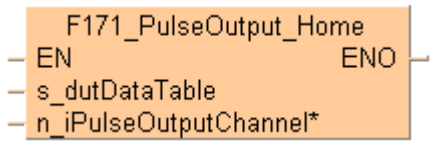
```
IF DF(bMotorSwitch) THEN
  dutTrapez.diInitialAndFinalSpeed:=diInitialSpeed;
  dutTrapez.diTargetSpeed:=diTargetSpeed;
  dutTrapez.diAccelerationDecelerationTime:=diAccelerationTime;
  dutTrapez.diDeviationCounterClearSignalOutputTime:=10;
END_IF;

IF DF(bMotorSwitch) THEN
  F171_PulseOutput_Trapezoidal(s_dutDataTable := dutTrapez,
        n_iPulseOutputChannel := 0);
END_IF;
```

F171_PulseOutput_Home

Referenzpunktfahrt

Erklärung

Anhand der Parameter im angegebenen strukturierten Datentyp wird eine Referenzpunktfahrt durchgeführt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Es gibt zwei verschiedene Betriebsarten:

Nach dem Einschalten des Antriebssystems besteht ein vorher nicht bestimmbarer Versatz zwischen dem internen Positionswert (Istwert) und der mechanischen Position der Achse. Zur Herstellung des Positionsbezuges muss der interne Wert mit dem realen Positionswert der Achse synchronisiert werden. Die Synchronisation erfolgt durch Übernahme eines Positionswertes an einem bekannten Punkt (Referenzpunkt).

Bei der Ausführung eines Referenzpunktfahrtbefehls werden so lange Pulse ausgegeben, bis der Referenzpunkteingang aktiviert wird. Die E/A-Zuweisung richtet sich nach dem verwendeten Kanal.

Zum Abbremsen im Referenzpunktbereich geben Sie einen Referenzpunkteingang an und setzen Bit 4 des Sonderdatenregisters, in dem der Steuercode für die Pulsausgabe gespeichert wird (sys_wHscOrPulseControlCode), auf TRUE und zurück auf FALSE.

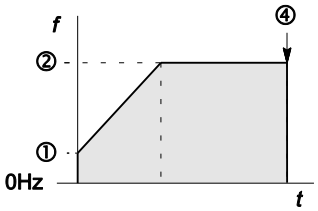
Der Referenzpunktausgang kann auf TRUE gesetzt werden, wenn die Referenzpunktfahrt beendet ist.

Während der Referenzpunktfahrt unterscheiden sich der Wert im Istwert-Speicherbereich und der tatsächliche Istwert. Wenn die Referenzpunktfahrt abgeschlossen ist, springt der Istwert auf 0.

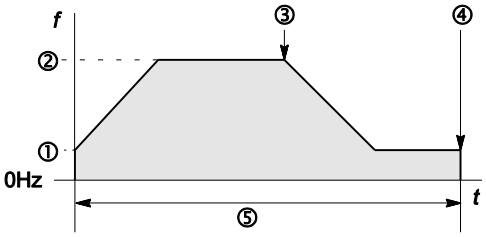
Es gibt zwei verschiedene Betriebsarten:

- Typ 1: Der Referenzpunkteingang wird aktiviert, unabhängig davon, ob ein Referenzpunkt-Sucheingang vorhanden ist, ob der Bremsvorgang bereits eingesetzt hat oder ob der Bremsvorgang abgeschlossen ist.

Ohne Referenzpunkteingang:

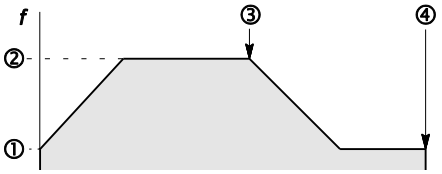


Mit Referenzpunkteingang:



①	Anfangs- und Endgeschwindigkeit	③	Referenzpunkteingang: TRUE
②	Sollgeschwindigkeit	④	Referenzpunkteingang: TRUE
⑤	Referenzpunkteingang jederzeit aktivierbar		

- Typ 2: Der Referenzpunkteingang kann nur aktiviert werden, nachdem der Bremsvorgang (ausgelöst durch einen Referenzpunkt-Sucheingang) abgeschlossen ist.



①	Anfangs- und Endgeschwindigkeit	③	Referenzpunkteingang: TRUE
②	Sollgeschwindigkeit	④	Referenzpunkteingang: TRUE
⑤	Referenzpunktfahrt erst aktivierbar, wenn Bremsvorgang abgeschlossen		

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: F171_PulseOutput_Home_DUT

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Anfangs- und Endgeschwindigkeit
- Sollgeschwindigkeit
- Beschleunigungs-/Bremszeit
- Abweichungszähler zurückgesetzt

Merkmale der Pulsausgabe

- Die Pulsausgabefrequenz ändert sich entsprechend der festgelegten Beschleunigungs- und Bremszeit.
- Der Unterschied zwischen der Soll- und Anfangsgeschwindigkeit bestimmt die Steigung der Rampen.

Allgemeine Programmierinformation

- Auch wenn der Referenzpunkt erreicht ist, werden durch die Ausführung dieses Befehls Pulse ausgegeben.
- Wenn der Referenzpunkteingang während der Beschleunigung aktiviert wird, beginnt die Abbremsung.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Das Referenzpunktausgangssignal wird bestimmten Ausgangsnummern zugewiesen, die jeweils einem SPS-Typ entsprechen.
- Die Pulsausgabe wird während des Online-Editierens gestoppt und erst nach der Übertragung der Programmänderungen fortgesetzt.
- FP-X: Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- FPΣ: Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- FPΣ: Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- FP-X: Setzen Sie den gewünschten Kanal im Systemregister auf "Pulsausgabe".
- FPΣ: Wenn der Kreisinterpolationsbefehl **F176** ausgeführt wird, wird der Kontrollmerker (sys_bIsCircularInterpolationActive) auf TRUE gesetzt. Der Status dieses Merkers bleibt erhalten, bis der Sollwert erreicht ist (auch wenn die Ausführungsbedingung nicht mehr TRUE ist). In dieser Zeit lassen sich keine anderen Pulsausgabebefehle ausführen.

- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F171_PulseOutput_Home (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_dutDataTable	F171_PulseOutput_Home_DUT	Startadresse des Bereichs, der die Datentabelle enthält
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgabekanal FP-Σ: 0, 2 FP-X R: 0, 1 FP-X C14T: 0, 1, 2 FP-X C30T/C60T: 0, 1, 2, 3

Operanden

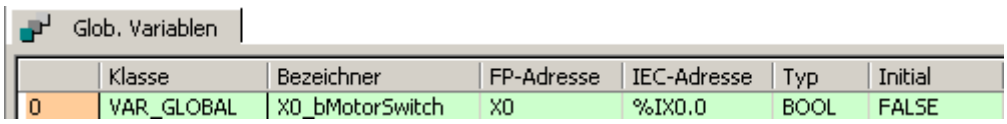
Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ die Anfangsgeschwindigkeit > Sollgeschwindigkeit ist ▪ FP-X: die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	kurzzeitig	

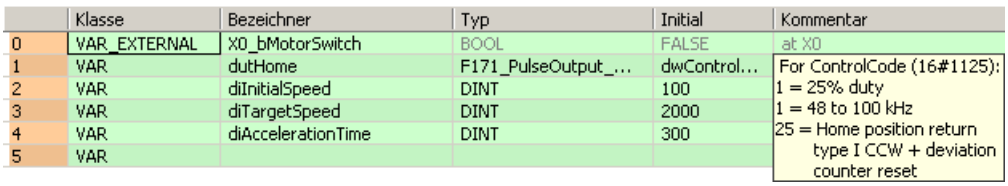
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

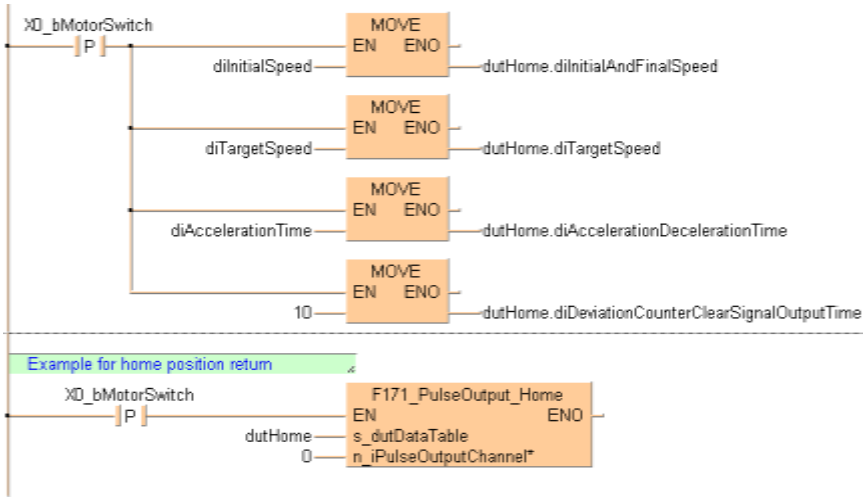


SDT Der SDT F171_PulseOutput_Home_DUT ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

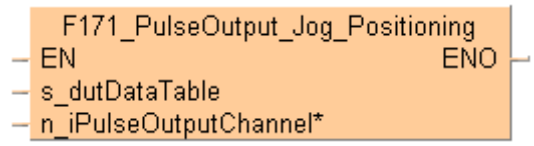
```
IF DF(X0_bMotorSwitch) THEN
        dutHome.diInitialAndFinalSpeed:=diInitialSpeed;
        dutHome.diTargetSpeed:=diTargetSpeed;
        dutHome.diAccelerationDecelerationTime:=diAccelerationTime;
        dutHome.diDeviationCounterClearSignalOutputTime:=10;
END_IF;
(*Example for home position return*)
IF DF(X0_bMotorSwitch) THEN
   F171_PulseOutput_Home(s_dutDataTable :=dutHome,
     n_iPulseOutputChannel := 0);
END_IF;
```

F171_PulseOutput_ Jog_Positioning

Tipp-Betrieb und Positionierung

Erklärung

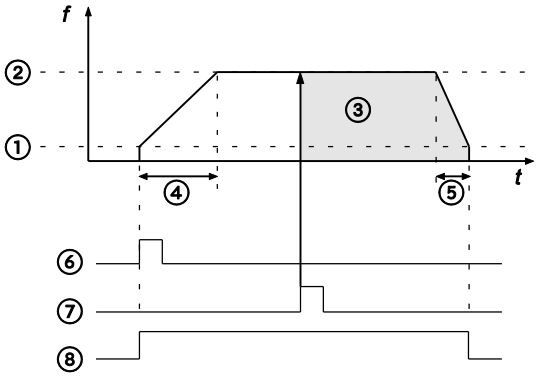
Wenn der Positionierungstrigger-Eingang auf TRUE gesetzt wird, wird die zuvor festgelegte Zahl von Pulsen ausgegeben. Ehe der Sollwert erreicht ist und die Pulsausgabe endet, wird eine Abbremsung ausgeführt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Es gibt zwei verschiedene Betriebsarten:

Typ 0: Die Geschwindigkeit lässt sich nur innerhalb des für die Sollgeschwindigkeit festgelegten Bereichs ändern.
Typ 1: Die Sollgeschwindigkeit lässt sich einmal ändern, wenn der Positionierungstrigger-Eingang auf TRUE gesetzt wird.

Merkmale der Pulsausgabe



①	Anfangs- und Endgeschwindigkeit	⑤	Bremszeit
②	Sollgeschwindigkeit	⑥	Ausführungsbedingung
③	Sollwert	⑦	Positionierungstrigger-Eingang
④	Beschleunigungszeit	⑧	Kontrollmerker für Pulsausgabe

- Die Pulsausgabefrequenz ändert sich mit der angegebenen Beschleunigungs- und Bremszeit.
- Die Differenz zwischen Soll- und Anfangsgeschwindigkeit bestimmt die Steigung der Bremsrampe.
- Die Differenz zwischen Soll- und Endgeschwindigkeit bestimmt die Steigung der Bremsrampe.
- Nachdem der Positionierungstrigger-Eingang auf TRUE geschaltet hat, wird die Pulsausgabe bis zum Sollwert fortgesetzt; dann setzt der Bremsvorgang ein und es erfolgt der Stopp.

Pulsausgabe beenden

Die Pulsausgabe wird durch einen der folgenden Vorgänge beendet:

- Positionierungstrigger-Eingang auf TRUE setzen (die Pulsausgabe wird solange fortgesetzt, bis der Sollwert erreicht und der Bremsvorgang abgeschlossen ist): Der Positionierungstrigger lässt sich starten, indem Sie den Positionierungstrigger-Eingang auf TRUE setzen oder Bit 6 des Datenregisters, in

dem der Steuercode für die Pulsausgabe gespeichert wird, von FALSE auf TRUE setzen (z.B. `MOVE(16#140, sys_wHscOrPulseControlCode);`).

- Signal für gebremsten Halt: Um einen abgebremsten Stopp auszuführen, setzen Sie Bit 5 des Datenregisters, in dem der Steuercode für die Pulsausgabe gespeichert wird auf FALSE → TRUE → FALSE (z.B. `MOVE(16#120, sys_wHscOrPulseControlCode);`). Wenn während der Beschleunigung das Signal für einen gebremsten Halt ausgelöst wird, wird der Bremsvorgang mit derselben Geschwindigkeit ausgeführt wie der normale Bremsvorgang, der von der Sollgeschwindigkeit ausgeht.
- Not-Aus-Stopp ausführen: Um einen Not-Aus-Stopp auszuführen, schalten Sie Bit 3 des Datenregisters, in dem der Steuercode für die Pulsausgabe gespeichert wird von FALSE auf TRUE (z.B. `MOVE(16#108, sys_wHscOrPulseControlCode);`).

Hinweis: Während des Not-Aus-Stopps müssen sämtliche Pulsausgabefunktionen des verwendeten Kanals deaktiviert werden.

■ Tipp-Betrieb Typ 0

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp:
F171_PulseOutput_Jog_Positioning_Type0_DUT

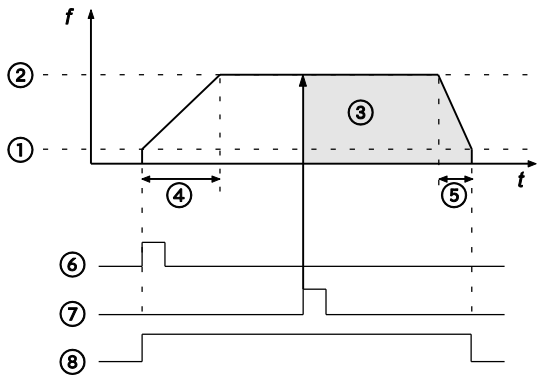
Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Anfangs- und Endgeschwindigkeit
- Sollgeschwindigkeit
- Beschleunigungszeit
- Bremszeit
- Sollwert

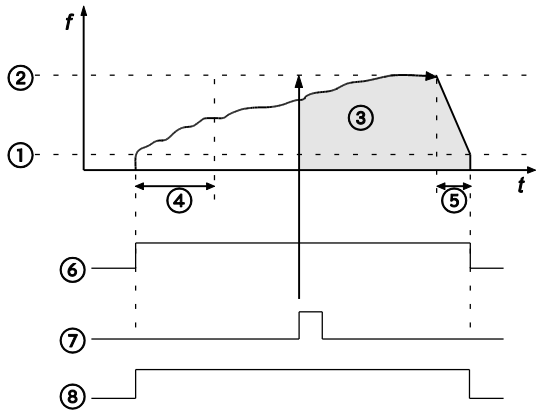
Die Sollgeschwindigkeit lässt sich während der Pulsausgabe ändern.

Sollgeschwindigkeit während der Pulsausgabe ändern

Sollgeschwindigkeit wird nicht geändert:



Sollgeschwindigkeit wird geändert:



①	Anfangs- und Endgeschwindigkeit	⑤	Bremszeit
②	Sollgeschwindigkeit	⑥	Ausführungsbedingung
③	Sollwert	⑦	Positionierungstrigger-Eingang
④	Beschleunigungszeit	⑧	Kontrollmerker für Pulsausgabe

- Die Geschwindigkeit kann nur geändert werden, wenn die Ausführungsbedingung TRUE ist.
- Wenn für die Sollgeschwindigkeit ein höherer Wert als 50kHz festgelegt wurde,

erfolgt eine Korrektur auf 50kHz.

- Wenn der Istwert zu Beginn der Beschleunigung im nicht zu lässigen Bereich liegt (z.B. sys_diPulseChannel0AccelerationForbiddenAreaStartingPosition), kann die Beschleunigung nicht ausgeführt werden.
- Die Bremsgeschwindigkeit kann nicht unter der korrigierten Restgeschwindigkeit liegen.
- Eine Änderung der Sollgeschwindigkeit ist nicht möglich, wenn der Befehl innerhalb eines Interrupt-Programms ausgeführt wird.

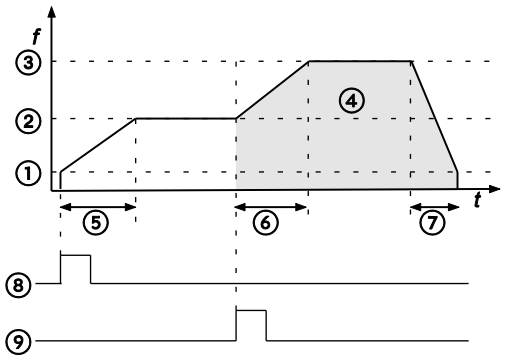
■ Tipp-Betrieb Typ 1

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp:
F171_PulseOutput_Jog_Positioning_Type1_DUT

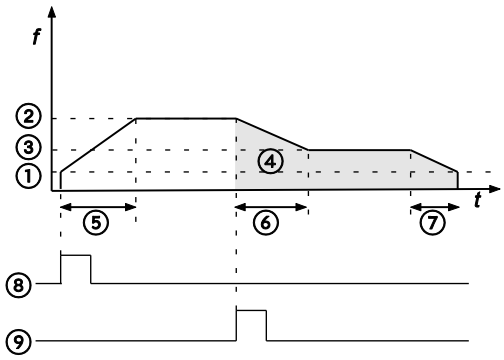
Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Anfangs- und Endgeschwindigkeit
- Sollgeschwindigkeit 1
- Beschleunigungszeit
- Sollgeschwindigkeit 2
- Wechselzeit
- Bremszeit
- Sollwert

Sollgeschwindigkeit 1 < Sollgeschwindigkeit 2:



Sollgeschwindigkeit 1 > Sollgeschwindigkeit 2:



①	Anfangs- und Endgeschwindigkeit	⑥	Wechselzeit
②	Sollgeschwindigkeit 1	⑦	Bremszeit
③	Sollgeschwindigkeit 2	⑧	Ausführungsbedingung
④	Sollwert	⑨	Positionierungstrigger-Eingang
⑤	Beschleunigungszeit		

Nachdem der Positionierungstrigger-Eingang auf TRUE schaltet, steigt oder sinkt die Pulsausgabefrequenz innerhalb der festgelegten Wechselzeit auf den Sollwert 2. Weitere Sollwertänderungen sind nicht möglich. Der Positionierungstrigger-Eingang wird nicht berücksichtigt, wenn er während der Beschleunigung eingeschaltet wird.

Allgemeine Programmierinformation

Achtung!

Wenn Sie Programme, die diesen Befehl verwenden, im RUN-Modus bearbeiten (Online-Editieren), wird die Pulsausgabe beendet.

- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- Positionierungstrigger-Eingang (X0, X1, X2, X3) im Systemregister setzen 402
- Der Positionierungstrigger-Eingang wird nur erkannt, wenn eine steigende Flanke (TRUE) anliegt.
- Der Befehl kann nicht initiiert werden, wenn ein abgebremster Stopp angefordert wurde.
- Um nach dem Stopp des Betriebs einen Neustart auszuführen, muss die Ausführungsbedingung zunächst auf FALSE und dann wieder auf TRUE gesetzt werden.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F171_PulseOutput_Jog_Positioning (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_DUT_DataTable	F171_PulseOutput_Jog_Positioning_Type0_DUT oder F171_PulseOutput_Jog_Positioning_Type1_DUT	Startadresse des Bereichs, der die Datentabelle enthält
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgabekanal 0–3

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ derselbe Kanal zweimal gestartet wurde ▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ die Anfangsgeschwindigkeit > Sollgeschwindigkeit ist ▪ die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	kurzzeitig	

Beispiel

In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT

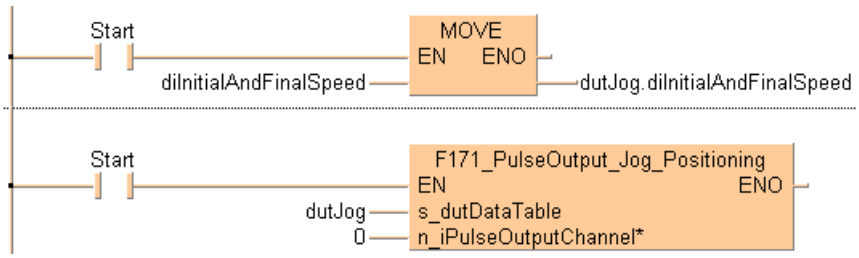
Der SDT F171_PulseOutput_Jog_Positioning_Type0_DUT ist in der FP Library enthalten.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Start	BOOL	FALSE	
1	VAR	dutJog	F171_PulseOutput_Jog_Positioning_Type0_DUT	dwControlCode := 16#010, diInitialAndFinalSpeed := 1000, diTargetSpeed := 7000, diAccelerationTime := 300, diDecelerationTime := 450, diTargetValue := 10000	Digit3: 0=Pulse output Digit2: 1=Fixed Digit0: 0=CW/CCW2
2	VAR	diInitialAndFinalSpeed	DINT		
3	VAR				

KOP



ST

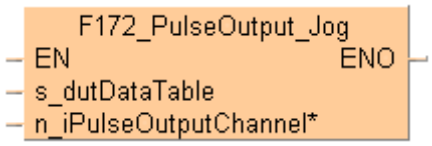
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF (Start) THEN
      dutJog.diInitialAndFinalSpeed:=diInitialAndFinalSpeed;
END_IF;
IF (Start) THEN
  F171_PulseOutput_Jog_Positioning(s_dutDataTable := dutJog, 0);
END_IF;
```

F172_PulseOutput_Jog

Tipp-Betrieb

Erklärung Dieser Befehl wird für den Tipp-Betrieb verwendet. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



■ Beschreibung für FP-Sigma, FP-X

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp:
F172_PulseOutput_Jog_Type0_DUT_0 (Ohne Sollwertvergleich) oder
F172_PulseOutput_Jog_Type1_DUT_0 (Mit Sollwertvergleich)

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

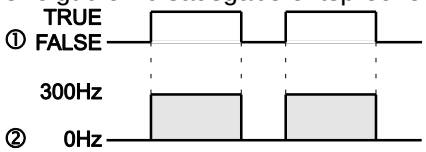
- Steuercode
- Frequenz
- Sollwert

Merkmale der Pulsausgabe

Die Frequenz und der Sollwert können in jedem Zyklus geändert werden. Der Steuercode kann jedoch nicht während der Befehlsausführung geändert werden.

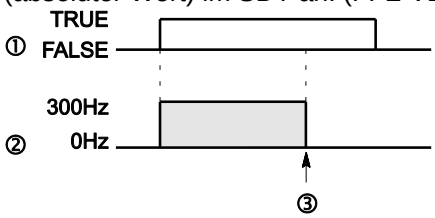
Es gibt zwei verschiedene Betriebsarten:

- Ohne Sollwertvergleich (Typ 0): Solange die Ausführungsbedingung TRUE ist, erfolgt die Pulsausgabe entsprechend den Werten im strukturierten Datentyp.



①	Ausführungsbedingung
②	Pulsausgabe Rechtslauf

- Mit Sollwertvergleich (Typ 1): Die Pulsausgabe stoppt, wenn der Sollwert erreicht ist. Setzen Sie diesen Modus im Steuercode und geben Sie den Sollwert (absoluter Wert) im SDT an. (FPΣ V1.4 oder neuer, FP-X)



①	Ausführungsbedingung
②	Pulsausgabe Rechtslauf
③	Sollwert erreicht (Pulsausgabe stoppt)

Allgemeine Programmierinformation

Achtung!

Wenn Sie Programme, die diesen Befehl verwenden, im RUN-Modus bearbeiten (Online-Editieren), wird die Pulsausgabe beendet.

- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- FP-X: Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- FPΣ: Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- FPΣ: Wenn der Kreisinterpolationsbefehl **F176** ausgeführt wird, wird der Kontrollmerker (sys_bIsCircularInterpolationActive) auf TRUE gesetzt. Der Status dieses Merkers bleibt erhalten, bis der Sollwert erreicht ist (auch wenn die Ausführungsbedingung nicht mehr TRUE ist). In dieser Zeit lassen sich keine anderen Pulsausgabebefehle ausführen.
- FPΣ: Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- FP-X: Setzen Sie den gewünschten Kanal im Systemregister auf "Pulsausgabe".
- Wird die Befehlsausführung mit einem ungültigen Frequenzwert gestartet, tritt ein Operationsfehler auf. Erhält die Frequenz während der Befehlsausführung einen ungültigen Wert, wird sie auf den untersten oder obersten Wert des zulässigen Bereichs eingestellt.
- Eine Änderung des Steuercodes während der Befehlsausführung hat keine Auswirkungen.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

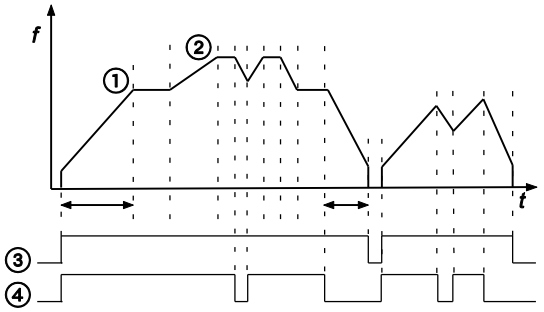
Beschreibung für FP0R

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp:
F172_PulseOutput_Jog_Type0_DUT_1 (Ohne Sollwertvergleich) oder
F172_PulseOutput_Jog_Type1_DUT_1 (Mit Sollwertvergleich)

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Anfangs- und Endgeschwindigkeit
- Sollgeschwindigkeit
- Beschleunigungszeit
- Bremszeit
- Sollwert

Merkmale der Pulsausgabe

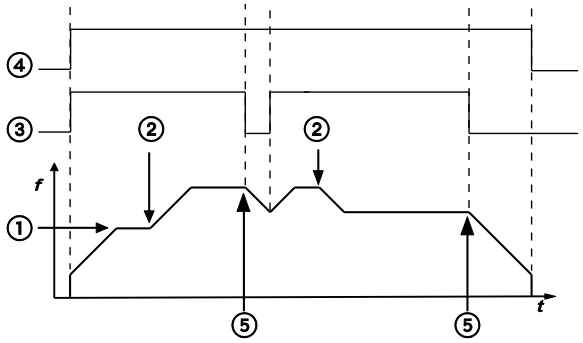


①	Sollgeschwindigkeit 1	③	Kontrollmerker für Pulsausgabe
②	Sollgeschwindigkeit 2	④	Ausführungsbedingung

- Die Pulsausgabefrequenz ändert sich mit der angegebenen Beschleunigungs- und Bremszeit.
- Die Differenz zwischen der Maximalgeschwindigkeit von 50kHz und der Anfangsgeschwindigkeit bestimmt die Steigung der Beschleunigungsrampe.
- Die Differenz zwischen der Maximalgeschwindigkeit von 50kHz und der Endgeschwindigkeit bestimmt die Steigung der Bremsrampe.
- Wenn die Ausführungsbedingung nach dem Befehlsstart auf FALSE schaltet, wird ein abgebremster Stopp ausgeführt.
- Wenn die Ausführungsbedingung während des Bremsvorgangs auf TRUE schaltet, wird wieder beschleunigt.
- Die Sollgeschwindigkeit lässt sich während der Pulsausgabe ändern.
- Die Pulse werden mit einem Puls-Pausenverhältnis von 25% ausgegeben.
- Bei der Verwendung der Methode Pulsausgabe/Richtungsanzeige werden die Pulse ca. 300μs nach der Ausgabe des Richtungsanzeigesignals ausgegeben; wobei das Verhalten eines Motorantriebs berücksichtigt wird.
- Wenn während der Beschleunigung das Signal für einen gebremsten Halt ausgelöst wird, wird der Bremsvorgang mit derselben Geschwindigkeit ausgeführt wie der normale Bremsvorgang, der von der Sollgeschwindigkeit ausgeht.
- Die Beschleunigungs- und Bremszeit hat Priorität gegenüber der Anfangs- und Endgeschwindigkeit. Das heißt, die Werte für die Beschleunigungs- und Bremszeit werden nicht geändert, während die Werte für die Anfangs- und Endgeschwindigkeit vom Pulsausgabebefehl korrigiert werden können, um die Beschleunigung und den Bremsvorgang in der festgelegten Zeit zu erreichen. Die geänderten Werte werden in die Datenregister geschrieben, auf die Sie über die Systemvariablen sys_iPulseChannelxCorrectedInitialSpeed und sys_iPulseChannelxCorrectedFinalSpeed zugreifen können (wobei x=Kanalnummer).

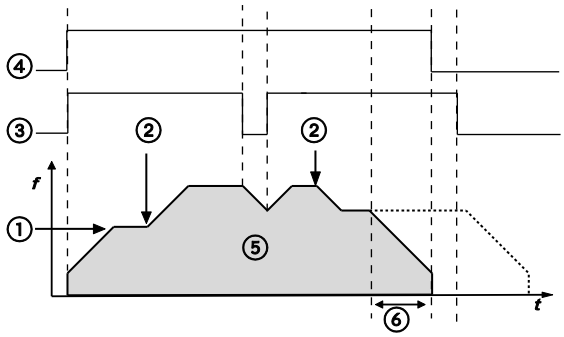
Es gibt zwei verschiedene Betriebsarten:

- Ohne Sollwertvergleich (Typ 0): Solange die Ausführungsbedingung TRUE ist, erfolgt die Pulsausgabe entsprechend den Werten im strukturierten Datentyp. Ein gebremster Halt beginnt, sobald die Ausführungsbedingung FALSE ist.



①	Anfangs- und Endgeschwindigkeit	④	Kontrollmerker für Pulsausgabe
②	Änderung der Sollgeschwindigkeit	⑤	Gebremster Halt
③	Ausführungsbedingung		

- Mit Sollwertvergleich (Typ 1): Die Pulsausgabe stoppt, wenn der Sollwert erreicht ist. Setzen Sie diesen Modus im Steuercode und geben Sie den Sollwert (absoluter Wert) im SDT an. Wenn der Sollwert erreicht ist, wird ein gebremster Halt ausgeführt. Der Bremsvorgang wird in der angegebenen Bremszeit durchgeführt.



①	Anfangs- und Endgeschwindigkeit	④	Kontrollmerker für Pulsausgabe
②	Änderung der Sollgeschwindigkeit	⑤	Sollwert
③	Ausführungsbedingung	⑥	Bremszeit

Sollgeschwindigkeit während der Pulsausgabe ändern

- Wenn der Istwert zu Beginn der Beschleunigung im nicht zu lässigen Bereich liegt (z.B. sys_diPulseChannel0AccelerationForbiddenAreaStartingPosition), kann die Beschleunigung nicht ausgeführt werden.
- Die Bremsgeschwindigkeit kann nicht unter der korrigierten Restgeschwindigkeit liegen.

Allgemeine Programmierinformation

Achtung!

Wenn Sie Programme, die diesen Befehl verwenden, im RUN-Modus bearbeiten (Online-Editieren), wird die Pulsausgabe beendet.

- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- Eine Änderung des Steuercodes während der Befehlsausführung hat keine Auswirkungen.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F172_PulseOutput_Jog (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_dutDataTable	**FP-Σ, FP-X:** F172_PulseOutput_Jog_Type0_DUT_0 F172_PulseOutput_Jog_Type1_DUT_0 **FP0R:** F172_PulseOutput_Jog_Type0_DUT_1 F172_PulseOutput_Jog_Type1_DUT_1	Startadresse des Bereichs, der die Datentabelle enthält
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgabekanal FP-Σ: 0, 2 FP-X R: 0, 1 FP-X C14T: 0, 1, 2 FP-X C30T/C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ FP0R/FP-X: die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	kurzzeitig	

Beispiel

In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT

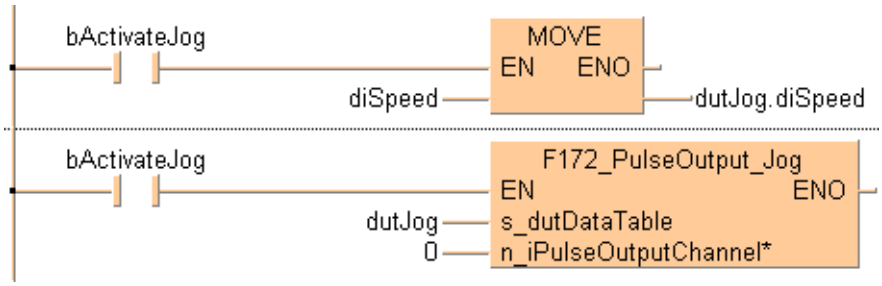
Der SDT F172_PulseOutput_Jog_Type0_DUT_0 ist in der FP Library enthalten.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	Start_X2	BOOL	FALSE	
1	VAR	dutJog	F172_PulseOutput_Jog_Type0_DUT_0	dwControlCode := 16#1110	ControlCode 16#1110= 1 = 25% duty 1= 48 to 100 kHz 10 = incremental counting CW Frequency = 300 Hz
2	VAR	diSpeed	DINT	300	
3	VAR				

KOP



ST

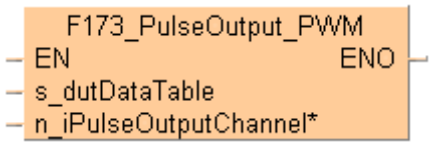
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF (bActivateJog) THEN
  dutJog.diSpeed:=diSpeed;
END_IF;
IF (bActivateJog) THEN
  F172_PulseOutput_Jog(s_dutDataTable := dutJog, 0);
END_IF;
```

F173_PulseOutput_PWM

Pulsausgabe mit Angabe des Kanals

Erklärung Dieser Befehl liefert ein pulsweitenmoduliertes Ausgangssignal. Die Parameter für die Pulsausgabe werden in einem SDT festgelegt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: F173_PulseOutput_PWM_DUT

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Ungefähre Frequenz
- Puls-Pausenverhältnis (der Pulsdauer und -periode)

Allgemeine Programmierinformation

Achtung!

Wenn Sie Programme, die diesen Befehl verwenden, im RUN-Modus bearbeiten (Online-Editieren), wird die Pulsausgabe beendet.

- Das Puls-Pausenverhältnis kann vom festgelegten Quotienten abweichen, wenn es nahe am Minimal- oder Maximalwert liegt, denn es ist abhängig von Lastspannung und Laststrom.
- Das Puls-Pausenverhältnis kann in jedem Zyklus geändert werden.
- Die Frequenzkonstante K kann jedoch nicht während der Befehlsausführung geändert werden. Wird der Wert verändert, hat dies nur Auswirkungen auf die Auflösung des Puls-Pausenverhältnisses, nicht auf die Frequenz.
- Wenn das Puls-Pausenverhältnis einen Wert erhält, der außerhalb des zulässigen Bereichs liegt, während der Befehl ausgeführt wird, wird der Maximalwert für das Puls-Pausenverhältnis eingestellt. Ein Operationsfehler wird ausgegeben, sobald die Befehlsausführung beginnt.
- Wenn während der Befehlsausführung ein ungültiger Frequenzwert eingestellt wird, wird eine Auflösung von 100 gewählt. Zu Beginn der Befehlsausführung wird kein Operationsfehler angezeigt.
- Wenn das Puls-Pausenverhältnis während der Befehlsausführung zu 100% oder mehr geändert wird, wird der für diese Auflösung maximal mögliche Frequenzwert eingestellt. Zu Beginn der Befehlsausführung wird kein Operationsfehler angezeigt.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- FP-X, FP0R: Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- FPΣ: Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B.

sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.

- FPΣ: Wenn der Kreisinterpolationsbefehl **F176** ausgeführt wird, wird der Kontrollmerker (sys_bIsCircularInterpolationActive) auf TRUE gesetzt. Der Status dieses Merkers bleibt erhalten, bis der Sollwert erreicht ist (auch wenn die Ausführungsbedingung nicht mehr TRUE ist). In dieser Zeit lassen sich keine anderen Pulsausgabebefehle ausführen.
- FPΣ: Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- FP-X, FP0R: Wählen Sie für den gewünschten Kanal in den Systemregistern die Einstellung "PWM-Ausgang".
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F173_PulseOutput_PWM (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_dutDataTable	F173_PulseOutput_PWM_DUT	Startadresse des Bereichs, der die Datentabelle enthält
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgabekanal FP-Σ: 0, 2 FP-X R: 0, 1 FP-X C14T: 0, 1, 2 FP-X C30T/C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen (bei erstmaliger Ausführung des Befehls) ▪ FP0R/FP-X: die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

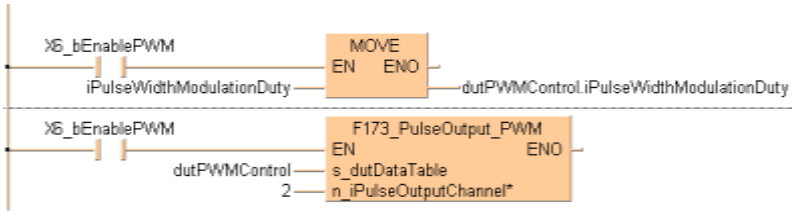
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	X6_bEnablePWM	X6	%IX0.6	BOOL	FALSE

SDT Der SDT F173_PulseOutput_PWM_DUT ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	X6_bEnablePWM	BOOL	FALSE	
1	VAR	dutPWMControl	F173_PulseOutput_PWM_DUT	iFrequencyValue := 1	iFrequencyValue := 1: f=2.0 Hz, T=502.5 ms;
2	VAR	iPulseWidthModulationDuty	INT	500	500 = 50% duty

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

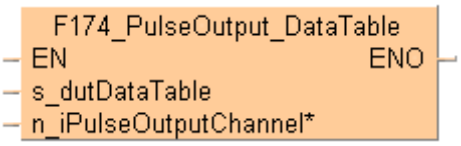
```
IF (X6_bEnablePWM) THEN
        dutPWMControl.iPulseWidthModulationDuty:=iPulseWidthModulationDuty;
END_IF;
IF (X6_bEnablePWM) THEN
        F173_PulseOutput_PWM(s_dutDataTable := dutPWMControl,
        n_iPulseOutputChannel := 2);
END_IF;
```

F174_PulseOutput_DataTable

Positionierprofil ohne Rampen

Erklärung

Der Befehl führt eine Rechtecksteuerung gemäß den Parametern des strukturierten Datentyps durch. Es können beliebig viele verschiedene Geschwindigkeiten und Sollwerte festgelegt werden. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.

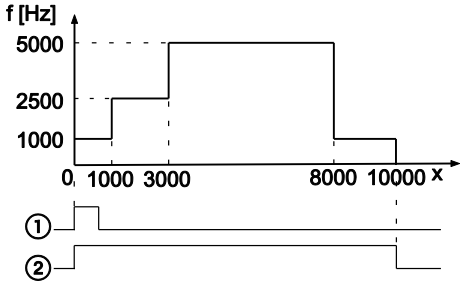


Erstellen Sie auf der Basis des folgenden Musters ihren eigenen SDT:
F174_PulseOutput_DataTable_8_Values_DUT

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Frequenz 1
- Sollwert 1
- Frequenz 2
- Sollwert 2
- ...
- Frequenz n
- Sollwert n
- Ende der Pulsausgabe

Merkmale der Pulsausgabe



x	Istwert des schnellen Zählers (Verfahrstrecke)
①	Ausführungsbedingung
②	Kontrollmerker für Pulsausgabe

- Die Pulse werden mit der festgelegten Frequenz ausgegeben, bis der Sollwert erreicht ist. Dann wird die Pulsausgabe mit dem zweiten Frequenzwert fortgesetzt, wieder bis der Sollwert erreicht ist usw.
- Die Pulsausgabe stoppt, wenn der letzte Sollwert erreicht ist.
- Die Frequenz 0 gibt die letzte Frequenz an und hält die Pulsausgabe an.

Allgemeine Programmierinformation

- FP-X, FP0R: Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- FPΣ: Der Kontrollmerker "Schneller Zähler" (z.B.

sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.

- Liegt der erste angegebene Frequenzwert außerhalb des zulässigen Bereichs, tritt ein Operationsfehler auf. (Beträgt der erste Frequenzwert 0, wird der Betrieb sofort angehalten; es erfolgt dann keine Pulsausgabe.)
- Beträgt der zweite angegebene Frequenzwert 0 oder liegt er außerhalb des zulässigen Bereichs, wird die Pulsausgabe gestoppt.
- Liegt der Sollwert außerhalb des zulässigen Bereichs, weicht die Anzahl der ausgegebenen Pulse eventuell vom festgelegten Wert ab.
- FPΣ: Wenn der Kreisinterpolationsbefehl **F176** ausgeführt wird, wird der Kontrollmerker (sys_bIsCircularInterpolationActive) auf TRUE gesetzt. Der Status dieses Merkers bleibt erhalten, bis der Sollwert erreicht ist (auch wenn die Ausführungsbedingung nicht mehr TRUE ist). In dieser Zeit lassen sich keine anderen Pulsausgabebefehle ausführen.
- FPΣ: Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- FPΣ, FP-X: Wird der Befehl zusammen mit einem schnellen Zähler, einem Zeit-Interrupt oder der SPS-Kopplung verwendet, benutzen Sie eine Frequenz von max. 80kHz.
- FP-X: Setzen Sie den gewünschten Kanal im Systemregister auf "Pulsausgabe".
- Die Pulsausgabe wird während des Online-Editierens gestoppt und erst nach der Übertragung der Programmänderungen fortgesetzt.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F174_PulseOutput_DataTable (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_dutDataTable	ANY_DUT	Startadresse des Bereichs, der die Datentabelle enthält Beispiel: F174_PulseOutput_DataTable_8_Values_DUT
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgabekanal FP-Σ: 0, 2 FP-X R: 0, 1 FP-X C14T: 0, 1, 2 FP-X C30T/C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ Frequenz 1 außerhalb des zulässigen Bereichs liegt ▪ FP0R/FP-X: die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	kurzzeitig	

Beispiel

In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL

In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	X0_bMotorSwitch	X0	%IX0.0	BOOL	FALSE

SDT

Der SDT F174_PulseOutput_DataTable_8_Values_DUT ist in der FP Library enthalten und kann als Beispiel genutzt werden.

F174_DUT [SDT]

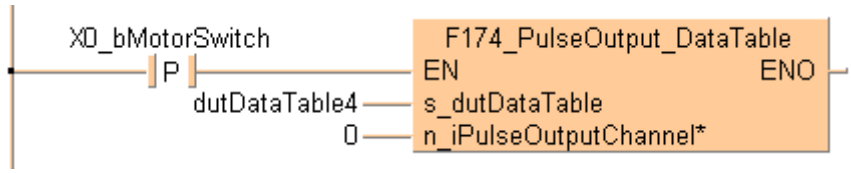
	Bezeichner	Typ	Ini...	Kommentar
0	ControlCode	DWORD	0	Highest word fixed to 0000
1	Frequency1	DINT	0	
2	TargetValue1	DINT	0	
3	Frequency2	DINT	0	
4	TargetValue2	DINT	0	
5	Frequency3	DINT	0	
6	TargetValue3	DINT	0	
7	Termination	DINT	0	End of data table

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	X0_bMotorSwitch	BOOL	FALSE	at X0
1	VAR	dutDataTable4	F174_DUT	ControlCode := 16#1200, Frequency1 := 1000, TargetValue1 := 1000, Frequency2 := 2500, TargetValue2 := 2000, Frequency3 := 5000, TargetValue3 := 5000, Frequency4 := 1000, TargetValue4 := 2000, Termination := 0	For ControlCode (16#1200): 1 = 25% duty 2 = 191 Hz to 100 kHz 0 = Relative value control 0 = CW (incremental counting)CC
2	VAR				

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

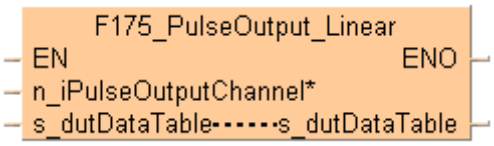
```
IF DF(X0_bMotorSwitch) THEN
  F174_PulseOutput_DataTable(s_dutDataTable:=dutDataTable4, 4);
END_IF;
```

F175_PulseOutput_Linear

Linearinterpolation

Erklärung

Durch eine zweikanalige Pulsausgabe wird eine geradlinige Verfahrstrecke erzeugt. Die Parameter für die Pulsausgabe werden in einem SDT festgelegt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



■ Beschreibung für FP-Sigma, FP-X (für die FP0R siehe auf Seite 934)

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp:
F175_PulseOutput_Linear_DUT_0

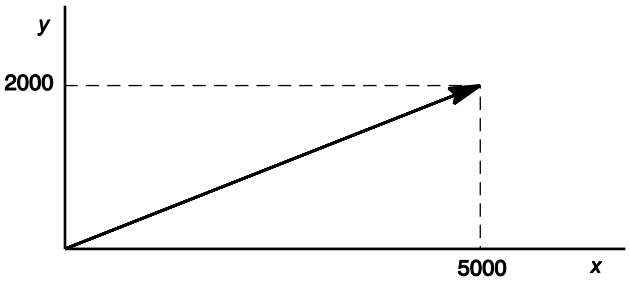
Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Anfangs- und Endgeschwindigkeit
- Sollgeschwindigkeit
- Beschleunigungs-/Bremszeit
- Sollwert (x-Achse)
- Sollwert (y-Achse)

Die folgenden Parameter für jede Achse werden bei der Befehlsausführung berechnet und im Speicherbereich für Rechenergebnisse des SDT gespeichert.

- Anfangs- und Restgeschwindigkeit (x-Achse)
- Sollwert (x-Achse)
- Anfangs- und Restgeschwindigkeit (y-Achse)
- Sollwert (y-Achse)
- Frequenzbereich (x-Achse)
- Frequenzbereich (y-Achse)
- Anzahl Beschleunigungs-/Bremsschritte (x-Achse)
- Anzahl Beschleunigungs-/Bremsschritte (y-Achse)

Merkmale der Pulsausgabe



5000	Sollwert (x-Achse) (Kanal 0)
2000	Sollwert (y-Achse) (Kanal 2) (FP-X: Kanal 1)

Beide Achsen werden so gesteuert, dass eine lineare Bewegung bis zur Sollposition erzielt wird.

Allgemeine Programmierinformation

- Der Sollwert für jede Achse muss innerhalb des Bereichs -8388608–8388607 liegen. Wenn dieser Befehl zusammen mit anderen Pulsausgabebefehlen verwendet wird, z.B. F171_PulseOutput_Trapezoidal (s. S. 904), muss der Sollwert in diesen Befehlen ebenfalls innerhalb desselben Bereichs liegen.
- Prüfen Sie bei Präzisionsanwendungen, ob die Mechanik der Maschine die geforderte Genauigkeit unterstützt.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- FP-X: Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- FPΣ: Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- FPΣ: Wenn der Kreisinterpolationsbefehl **F176** ausgeführt wird, wird der Kontrollmerker (sys_bIsCircularInterpolationActive) auf TRUE gesetzt. Der Status dieses Merkers bleibt erhalten, bis der Sollwert erreicht ist (auch wenn die Ausführungsbedingung nicht mehr TRUE ist). In dieser Zeit lassen sich keine anderen Pulsausgabebefehle ausführen.
- FPΣ: Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- FP-X: Setzen Sie den gewünschten Kanal im Systemregister auf "Pulsausgabe".
- Die Pulsausgabe wird während des Online-Editierens gestoppt und erst nach der Übertragung der Programmänderungen fortgesetzt.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

■ Beschreibung für FP0R

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp:
F175_PulseOutput_Linear_DUT_1

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

Allgemeine Programmierinformation

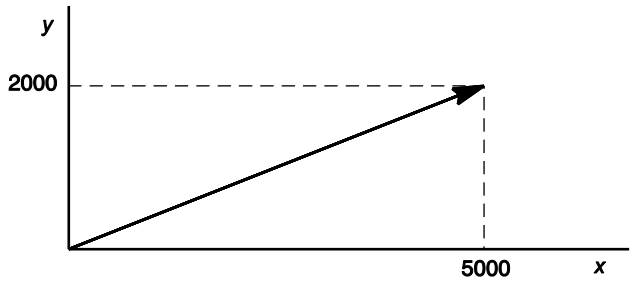
- Steuercode
- Anfangs- und Endgeschwindigkeit

- Sollgeschwindigkeit
- Beschleunigungszeit
- Bremszeit
- Sollwert (x-Achse)
- Sollwert (y-Achse)

Die folgenden Parameter für jede Achse werden bei der Befehlsausführung berechnet und im Speicherbereich für Rechenergebnisse des SDT gespeichert.

- Anfangs- und Restgeschwindigkeit (x-Achse)
- Sollwert (x-Achse)
- Anfangs- und Restgeschwindigkeit (y-Achse)
- Sollwert (y-Achse)

Merkmale der Pulsausgabe



5000	Sollwert (x-Achse) (Kanal 0)
2000	Sollwert (y-Achse) (Kanal 1)

Die Pulse werden von Kanal 0 (x-Achse) und Kanal 1 (y-Achse) so ausgegeben, dass die Anfangsgeschwindigkeit 500Hz, die Sollgeschwindigkeit 5kHz, und die Beschleunigungs-/Bremszeit 300ms beträgt. Beide Achsen werden über lineare Streckensteuerung zur Sollposition geführt.

Die Pulse werden mit einem Puls-Pausenverhältnis von 25% ausgegeben.

Bei der Verwendung der Methode Pulsausgabe/Richtungsanzeige werden die Pulse ca. 300µs nach der Ausgabe des Richtungsanzeigesignals ausgegeben; wobei das Verhalten eines Motorantriebs berücksichtigt wird.

Allgemeine Programmierinformation

- Die Pulsausgabe stoppt, wenn der letzte Sollwert erreicht ist.
- Der Sollwert für jede Achse muss innerhalb des Bereichs -8388608–8388607 liegen. Wenn dieser Befehl zusammen mit anderen Pulsausgabebefehlen verwendet wird, z.B. F171_PulseOutput_Trapezoidal (s. S. 904), muss der Sollwert in diesen Befehlen ebenfalls innerhalb desselben Bereichs liegen.
- Prüfen Sie bei Präzisionsanwendungen, ob die Mechanik der Maschine die geforderte Genauigkeit unterstützt.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- Die Pulsausgabe wird während des Online-Editierens gestoppt und erst nach der Übertragung der Programmänderungen fortgesetzt.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.

- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F175_PulseOutput_Linear (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
n_iPulseOutputChannel	Konstante	Pulsausgabekanal FP-Σ: 0, 2 FP-X R: 0, 1 FP-X C14T: 0, 1, 2 FP-X C30T/C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 Für die Interpolation verwenden Sie paarweise Kanal 0 und 1 oder Kanal 2 und 3. Sie können nur 0 oder 2 angeben (für C14T: nur 0).
s_dutDataTable	**FP-Σ, FP-X:** F175_PulseOutput_Linear_DUT_0 **FP0R:** F175_PulseOutput_Linear_DUT_1	Startadresse des Bereichs, der die Datentabelle enthält

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

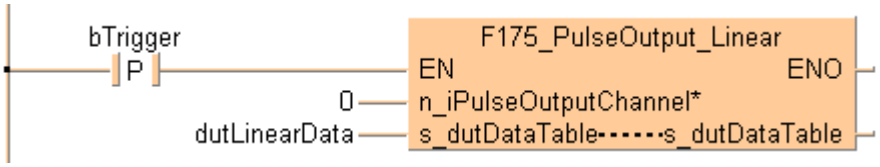
Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ Fmin > Fmax ▪ Fmax > 100kHz ▪ FP-X C14T, C30/60T (Verwendung von Kanal 2 und 3): Fmax > 20kHz ▪ Relativwertpositionierung: [Istwert + Sollwert] liegt außerhalb des Bereichs von -8388608 bis +8388607 ▪ Absolutwertpositionierung: Sollwert liegt außerhalb des Bereichs von -8388608 bis +8388607 ▪ FP-X: die Pulsausgabe nicht in den Systemregistern gesetzt wurde
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet. Dieses Beispiel wurde für die FP-Σ programmiert. Die Parameter für die FP0R unterscheiden sich geringfügig.

SDT Der SDT F175_PulseOutput_Linear_DUT_0 ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

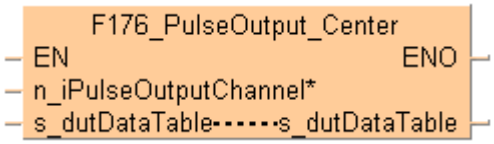
```
IF DF(bTrigger) THEN
      F175_PulseOutput_Linear(n_iPulseOutputChannel := 0,
      s_dutDataTable := dutLinearData);
END_IF;
```

F176_PulseOutput_Center

Kreisinterpolation (Mittelpunktverfahren)

Erklärung

Durch eine zweikanalige Pulsausgabe wird eine bogenförmige Verfahrstrecke erzeugt. Die Parameter für die Pulsausgabe werden in einem SDT festgelegt. Der Radius des Kreisbogens wird aus dem angegebenen Mittelpunkt und der Endposition errechnet. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Verwenden Sie folgenden, vordefinierten strukturierten Datentyp:
F176_PulseOutput_Center_DUT

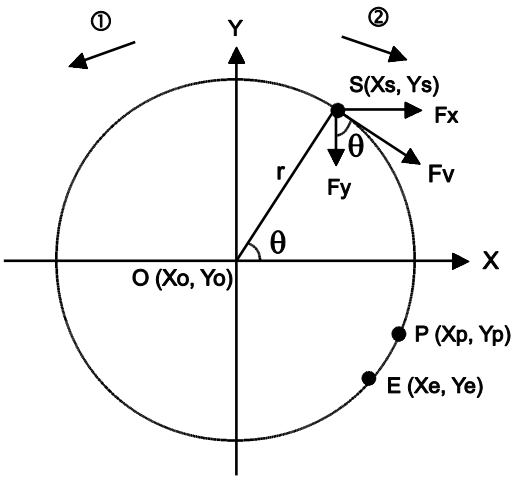
Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Resultierende Geschwindigkeit
- Sollwert (x-Achse)
- Sollwert (y-Achse)
- Mittelpunkt (x-Achse)
- Mittelpunkt (y-Achse)

Die folgenden Parameter für jede Achse werden bei der Befehlsausführung berechnet und im Speicherbereich für Rechenergebnisse des SDT gespeichert.

- Radius

Merkmale der Pulsausgabe



①	Drehrichtung: Rückwärts	②	Drehrichtung: Vorwärts
F_v:	Resultierende Geschwindigkeit	**O (Xo,Yo):**	Mittelpunkt
F_x:	Geschwindigkeit (x-Achse)	**S (Xs,Ys):**	Istposition (Start)
F_y:	Geschwindigkeit (y-Achse)	**P (Xp,Yp)**	Position auf Kreisbogen
r:	Radius	**E (Xe,Ye)**	Sollposition (Ende)

$$Fx = Fv\sin\theta = Fv\frac{|Ye-Yo|}{r} \qquad Fy = Fv\cos\theta = Fv\frac{|Xe-Xo|}{r}$$

Beispiel: Kanal 0 sei die x-Achse, Kanal 2 die y-Achse. Die Positionierung soll als

Absolutwertpositionierung durchgeführt werden.

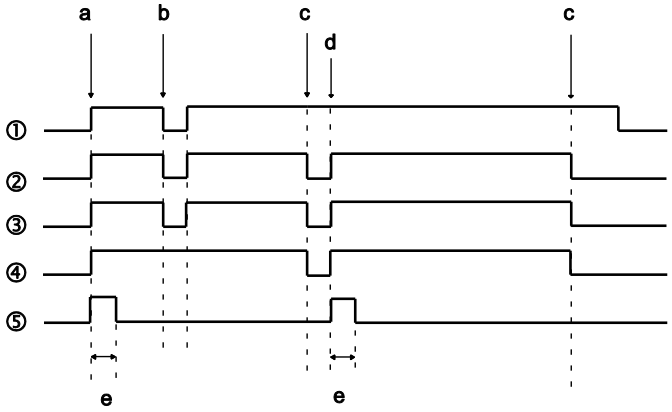
Die Istposition ist (θ=60°, Xs=5000, Ys=8660). Der Mittelpunkt O (Xo=0, Yo=0) wird als Referenzpunkt verwendet. Die Pulse werden von Kanal 0 (x-Achse) und Kanal 2 (y-Achse) mit einer Geschwindigkeit von Fv=2000Hz ausgegeben, bis die Endposition (θ=-30°, Xe=8660, Ye=-5000) erreicht ist.

Allgemeine Programmierinformation

- Die Ausführungsbedingung für diesen Befehl muss kontinuierlich TRUE sein. Wenn die Ausführungsbedingung FALSE ist, wird die Pulsausgabe beendet.
- Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- Wenn der Kreisinterpolationsbefehl **F176** ausgeführt wird, wird der Kontrollmerker (sys_bIsCircularInterpolationActive) auf TRUE gesetzt. Der Status dieses Merkers bleibt erhalten, bis der Sollwert erreicht ist (auch wenn die Ausführungsbedingung nicht mehr TRUE ist). In dieser Zeit lassen sich keine anderen Pulsausgabebefehle ausführen. Für einen Neustart der Kreisinterpolation führen Sie einen erzwungenen Stopp (Pulsausgabe beenden (s. S. 865)) aus. Der Kontrollmerker Kreisinterpolation (sys_bIsCircularInterpolationActive) wird hiermit auf FALSE gesetzt.
- Wurde für den Verkettungsmodus "Fortsetzen" gewählt, verwenden Sie einen Sondermerker (sys_bIsCircularInterpolationOverwritingPossible), um das Überschreiben des Sollwerts zuzulassen. Dieser Merker ist einen Zyklus lang TRUE, nachdem der Kreisinterpolationsbefehl ausgeführt wurde.
- Der Sollwert für jede Achse muss innerhalb des Bereichs -8388608–8388607 liegen. Wenn dieser Befehl zusammen mit anderen Pulsausgabebefehlen verwendet wird, z.B. F171_PulseOutput_Trapezoidal (s. S. 904), muss der Sollwert in diesen Befehlen ebenfalls innerhalb desselben Bereichs liegen.
- Die Genauigkeit der Kreisinterpolation kann abnehmen, wenn die Zykluszeit zu lange dauert.
- Programmänderungen im RUN-Modus (Online-Editieren) sind bei diesem Befehl nicht möglich.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Wenn Sie für die Istposition und die Sollposition denselben Wert eingeben, wird ein Kreis gefahren.
- Da es für Referenzpunktfahrten keine Interpolationsfunktion gibt, müssen Sie die Referenzpunktfahrt für jeden Kanal einzeln durchführen.
- Prüfen Sie bei Präzisionsanwendungen, ob die Mechanik der Maschine die geforderte Genauigkeit unterstützt.
- Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des

Sondermerkers in eine Variable am Programmanfang kopieren.

Verhalten der Merker während der Befehlsausführung



①	Ausführungsbedingung X0
②	Kontrollmerker für Pulsausgabe, Kanal 0 (sys_bIsPulseChannel0Active)
③	Kontrollmerker für Pulsausgabe, Kanal 2 (sys_bIsPulseChannel2Active)
④	Kontrollmerker "Kreisinterpolation" (sys_bIsCircularInterpolationActive)
⑤	Merker "Sollwertänderung" (sys_bIsCircularInterpolationOverwritingPossible)
a	Start
b	Ausführungsbedingung FALSE
c	Sollwert erreicht
d	Verkettungsmodus "Fortsetzen" starten
e	1 Zyklus

SPS-Typen **Verfügbarkeit von F176_PulseOutput_Center (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgabekanal 0, 2
s_dutDataTable	F176_PulseOutput_Center_DUT	Startadresse des Bereichs, der die Datentabelle enthält

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

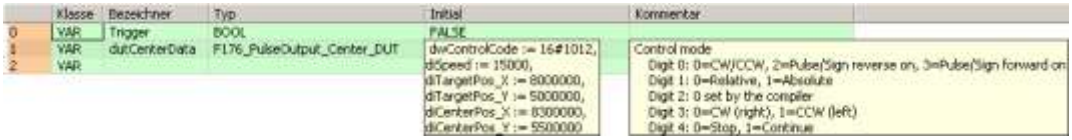
Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ Relativwertpositionierung: [Istwert + Sollwert] liegt außerhalb des Bereichs von -8388608 bis +8388607 ▪ Absolutwertpositionierung: Sollwert liegt außerhalb des Bereichs von -8388608 bis +8388607 ▪ Mittelpunkt **O** = Endposition **E** ▪ Mittelpunkt **O** = Startposition **S**
R9008	%MX0.900.8	kurzzeitig	

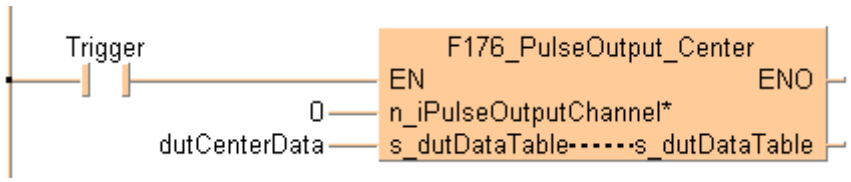
Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT Der SDT F176_PulseOutput_Center_DUT ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

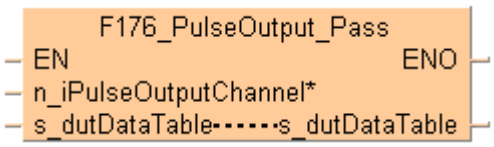
```
IF DF(Trigger) THEN
      F176_PulseOutput_Center(n_iPulseOutputChannel := 0,
      s_dutDataTable := dutCenterData);
END_IF;
```

F176_PulseOutput_Pass

Kreisinterpolation (Drei-Punkte-Verfahren)

Erklärung

Durch eine zweikanalige Pulsausgabe wird eine bogenförmige Verfahrstrecke erzeugt. Die Parameter für die Pulsausgabe werden in einem SDT festgelegt. Mittelpunkt und Radius des Kreisbogens werden aus dem Punkt auf dem Kreisbogen, der durchfahren werden muss, und der Endposition errechnet. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: **F176_PulseOutput_Pass_DUT**

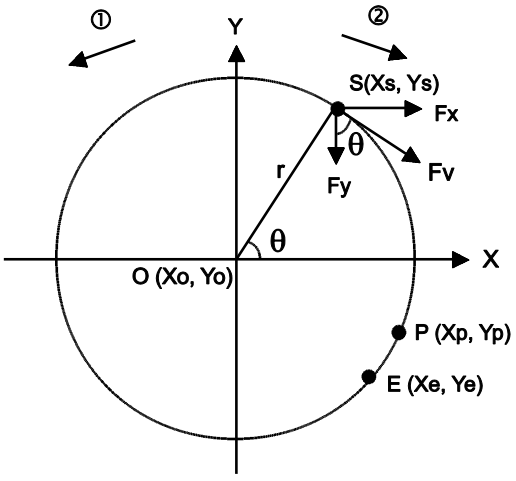
Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Resultierende Geschwindigkeit
- Sollwert (x-Achse)
- Sollwert (y-Achse)
- Position auf Kreisbogen (x-Achse)
- Position auf Kreisbogen (y-Achse)

Die folgenden Parameter für jede Achse werden bei der Befehlsausführung berechnet und im Speicherbereich für Rechenergebnisse des SDT gespeichert.

- Radius
- Mittelpunkt (x-Achse)
- Mittelpunkt (y-Achse)

Merkmale der Pulsausgabe



①	Drehrichtung: Rückwärts	②	Drehrichtung: Vorwärts
F_v:	Resultierende Geschwindigkeit	**O (Xo,Yo):**	Mittelpunkt
F_x:	Geschwindigkeit (x-Achse)	**S (Xs,Ys):**	Istposition (Start)
F_y:	Geschwindigkeit (y-Achse)	**P (Xp,Yp)**	Position auf Kreisbogen
r:	Radius	**E (Xe,Ye)**	Sollposition (Ende)

$$Fx = Fv\sin\theta = Fv\frac{|Ye-Yo|}{r} \qquad Fy = Fv\cos\theta = Fv\frac{|Xe-Xo|}{r}$$

Beispiel: Kanal 0 sei die x-Achse, Kanal 2 die y-Achse. Die Positionierung soll als

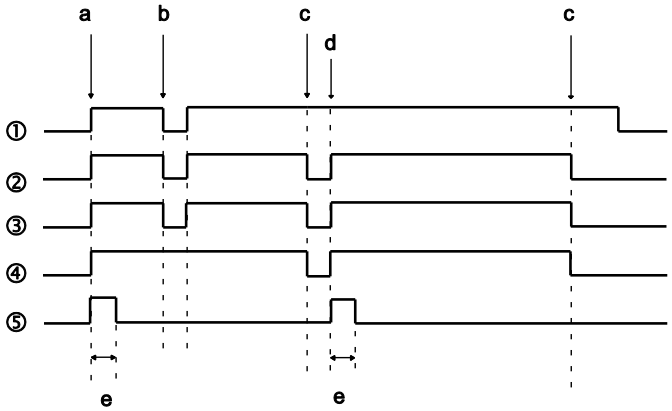
Absolutwertpositionierung durchgeführt werden.

Die Istposition ist (θ=60°, Xs=5000, Ys=8660). Die Pulse werden von Kanal 0 (x-Achse) und Kanal 2 (y-Achse) mit einer Geschwindigkeit von Fv=2000Hz ausgegeben. Wenn der Punkt auf dem Kreisbogen (θ=-20°, Xp=9396, Yp=3420) durchfahren ist und die Endposition erreicht wurde, stoppt die Pulsausgabe (θ=-30°, Xe=8660, Ye=-5000).

Allgemeine Programmierinformation

- Die Ausführungsbedingung für diesen Befehl muss kontinuierlich TRUE sein. Wenn die Ausführungsbedingung FALSE ist, wird die Pulsausgabe beendet.
- Der Kontrollmerker "Schneller Zähler" (z.B. sys_bIsHscChannel0ControlActive) und der Kontrollmerker "Pulsausgabe" (z.B. sys_bIsPulseChannel0Active) sind demselben Sondermerker zugeordnet (z.B. R903A). Daher ist sowohl der Kontrollmerker für den schnellen Zähler (z.B. sys_bIsHscChannel0ControlActive) als auch der Kontrollmerker für die Pulsausgabe (z.B. sys_bIsPulseChannel0Active) für den betreffenden Kanal TRUE, wenn ein schneller Zählerbefehl oder ein Pulsausgabebefehl ausgeführt wird. Solange dieser Merker auf TRUE steht, kann kein anderer schneller Zählerbefehl oder Pulsausgabebefehl ausgeführt werden.
- Wenn der Kreisinterpolationsbefehl **F176** ausgeführt wird, wird der Kontrollmerker (sys_bIsCircularInterpolationActive) auf TRUE gesetzt. Der Status dieses Merkers bleibt erhalten, bis der Sollwert erreicht ist (auch wenn die Ausführungsbedingung nicht mehr TRUE ist). In dieser Zeit lassen sich keine anderen Pulsausgabebefehle ausführen. Für einen Neustart der Kreisinterpolation führen Sie einen erzwungenen Stopp (Pulsausgabe beenden (s. S. 865)) aus. Der Kontrollmerker Kreisinterpolation (sys_bIsCircularInterpolationActive) wird hiermit auf FALSE gesetzt.
- Wurde für den Verkettungsmodus "Fortsetzen" gewählt, verwenden Sie einen Sondermerker (sys_bIsCircularInterpolationOverwritingPossible), um das Überschreiben des Sollwerts zuzulassen. Dieser Merker ist einen Zyklus lang TRUE, nachdem der Kreisinterpolationsbefehl ausgeführt wurde.
- Der Sollwert für jede Achse muss innerhalb des Bereichs -8388608–8388607 liegen. Wenn dieser Befehl zusammen mit anderen Pulsausgabebefehlen verwendet wird, z.B. F171_PulseOutput_Trapezoidal (s. S. 904), muss der Sollwert in diesen Befehlen ebenfalls innerhalb desselben Bereichs liegen.
- Die Genauigkeit der Kreisinterpolation kann abnehmen, wenn die Zykluszeit zu lange dauert.
- Programmänderungen im RUN-Modus (Online-Editieren) sind bei diesem Befehl nicht möglich.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Da es für Referenzpunktfahrten keine Interpolationsfunktion gibt, müssen Sie die Referenzpunktfahrt für jeden Kanal einzeln durchführen.
- Prüfen Sie bei Präzisionsanwendungen, ob die Mechanik der Maschine die geforderte Genauigkeit unterstützt.
- Setzen Sie einen schnellen Zähler, der einem Pulsausgabekanal zugeordnet ist, in den Systemregistern auf "Nicht genutzt".
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

Verhalten der Merker während der Befehlsausführung



①	Ausführungsbedingung X0
②	Kontrollmerker für Pulsausgabe, Kanal 0 (sys_bIsPulseChannel0Active)
③	Kontrollmerker für Pulsausgabe, Kanal 2 (sys_bIsPulseChannel2Active)
④	Kontrollmerker "Kreisinterpolation" (sys_bIsCircularInterpolationActive)
⑤	Merker "Sollwertänderung" (sys_bIsCircularInterpolationOverwritingPossible)
a	Start
b	Ausführungsbedingung FALSE
c	Sollwert erreicht
d	Verkettungsmodus "Fortsetzen" starten
e	1 Zyklus

SPS-Typen **Verfügbarkeit von F176_PulseOutput_Pass (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgabekanal 0, 2
s_dutDataTable	F176_PulseOutput_Pass_DUT	Startadresse des Bereichs, der die Datentabelle enthält

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

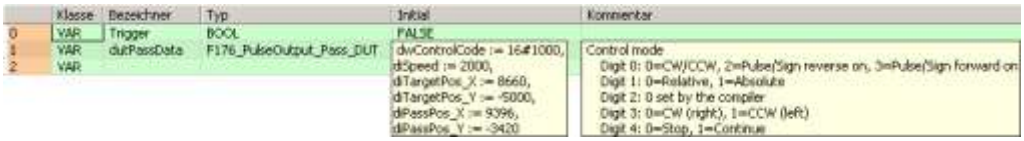
Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ Relativwertpositionierung: [Istwert + Sollwert] liegt außerhalb des Bereichs von -8388608 bis +8388607 ▪ Absolutwertpositionierung: Sollwert liegt außerhalb des Bereichs von -8388608 bis +8388607 ▪ Startposition S = Endposition E ▪ Startposition S = Position auf dem Kreisbogen P ▪ Position auf dem Kreisbogen P =Endposition E ▪ Startposition S, Position auf dem Kreisbogen P, und Endposition E ergeben eine nahezu gerade Linie.
R9008	%MX0.900.8	kurzzeitig	

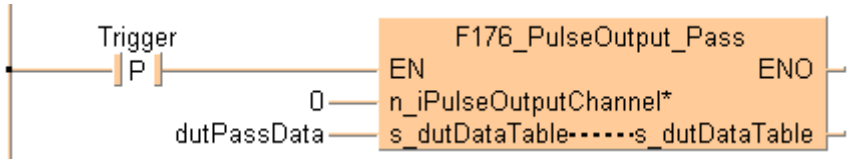
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT Der SDT F176_PulseOutput_Pass_DUT ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

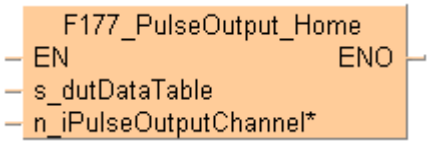
```
IF DF(Trigger) THEN
        F176_PulseOutput_Pass(n_iPulseOutputChannel := 0,
        s_dutDataTable := dutPassData);
END_IF;
```

F177_PulseOutput_Home

Referenzpunktfahrt

Erklärung

Anhand der Parameter im angegebenen strukturierten Datentyp wird eine Referenzpunktfahrt durchgeführt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Nach dem Einschalten des Antriebssystems besteht ein vorher nicht bestimmbarer Versatz zwischen dem internen Positionswert (Istwert) und der mechanischen Position der Achse. Zur Herstellung des Positionsbezuges muss der interne Wert mit dem realen Positionswert der Achse synchronisiert werden. Die Synchronisation erfolgt durch Übernahme eines Positionswertes an einem bekannten Punkt (Referenzpunkt).

Bei der Ausführung eines Referenzpunktfahrtbefehls werden so lange Pulse ausgegeben, bis der Referenzpunkteingang aktiviert wird. Die E/A-Zuweisung richtet sich nach dem verwendeten Kanal.

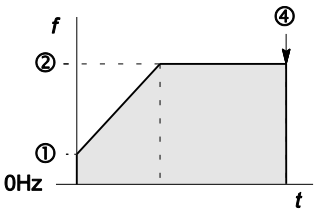
Zum Abbremsen im Referenzpunktbereich geben Sie einen Referenzpunkteingang an und setzen Bit 4 des Sonderdatenregisters, in dem der Steuercode für die Pulsausgabe gespeichert wird (sys_wHscOrPulseControlCode), auf TRUE und zurück auf FALSE.

Der Referenzpunktausgang kann auf TRUE gesetzt werden, wenn die Referenzpunktfahrt beendet ist.

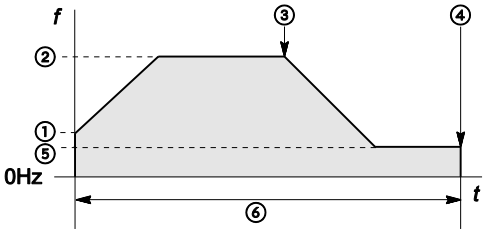
Es gibt zwei verschiedene Betriebsarten:

- Typ 0: Der Referenzpunkteingang wird aktiviert, unabhängig davon, ob ein Referenzpunkt-Sucheingang vorhanden ist, ob der Bremsvorgang bereits eingesetzt hat oder ob der Bremsvorgang abgeschlossen ist.

Ohne Referenzpunkteingang:

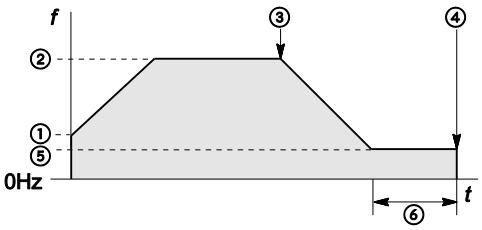


Mit Referenzpunkteingang:



① Anfangsgeschwindigkeit
② Sollgeschwindigkeit
③ Referenzpunkteingang: TRUE
④ Referenzpunkteingang: TRUE
⑤ Suchgeschwindigkeit
⑥ Referenzpunkteingang jederzeit aktivierbar

- Typ 1: Der Referenzpunkteingang kann nur aktiviert werden, nachdem der Bremsvorgang (ausgelöst durch einen Referenzpunkt-Sucheingang) abgeschlossen ist.



① Anfangsgeschwindigkeit
② Sollgeschwindigkeit
③ Referenzpunkteingang: TRUE
④ Referenzpunkteingang: TRUE
⑤ Suchgeschwindigkeit
⑥ Referenzpunktfahrt erst aktivierbar, wenn Bremsvorgang abgeschlossen

Verwenden Sie folgenden, vordefinierten strukturierten Datentyp:
F177_PulseOutput_Home_Type0_DUT oder F177_PulseOutput_Home_Type1_DUT

Folgende Parameter lassen sich im strukturierten Datentyp festlegen:

- Steuercode
- Anfangsgeschwindigkeit
- Sollgeschwindigkeit
- Beschleunigungszeit
- Bremszeit
- Suchgeschwindigkeit
- Abweichungszähler zurückgesetzt (Ausgabezeit)

Merkmale der Pulsausgabe

- Die Pulsausgabefrequenz ändert sich mit der angegebenen Beschleunigungs- und Bremszeit.
- Der Unterschied zwischen der Soll- und Anfangsgeschwindigkeit bestimmt die Steigung der Rampen.
- Die Pulse werden mit einem Puls-Pausenverhältnis von 25% ausgegeben.
- Bei der Verwendung der Methode Pulsausgabe/Richtungsanzeige werden die Pulse ca. 300µs nach der Ausgabe des Richtungsanzeigesignals ausgegeben; wobei das Verhalten eines Motorantriebs berücksichtigt wird.

■ Allgemeine Programmierinformation

- Setzen Sie den gewünschten Kanal im Systemregister auf "Pulsausgabe".
- Auch wenn der Referenzpunkt erreicht ist, werden durch die Ausführung dieses Befehls Pulse ausgegeben.
- Wenn der Referenzpunkteingang während der Beschleunigung aktiviert wird, beginnt die Abbremsung.
- Das Referenzpunktausgangssignal wird bestimmten Ausgangsnummern zugewiesen, die jeweils einem SPS-Typ entsprechen.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Während der Ausführung eines Pulsausgabebefehls und der Ausgabe von Pulsen steht der Kontrollmerker "Pulsausgabe" des entsprechenden Kanals (z.B. sys_bIsPulseChannel0Active) auf TRUE. Solange dieser Merker auf TRUE steht, kann kein anderer Pulsausgabebefehl ausgeführt werden.
- Die Pulsausgabe wird während des Online-Editierens gestoppt und erst nach der Übertragung der Programmänderungen fortgesetzt.
- Es ist unbedingt empfehlenswert, die Möglichkeit für einen erzwungenen Stop (s. S. 865) in Ihrem Positionierprogramm bereit zu stellen.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F177_PulseOutput_Home (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s_dutDataTable	F177_PulseOutput_Home_Type0_DUT oder F177_PulseOutput_Home_Type1_DUT	Startadresse des Bereichs, der die Datentabelle enthält
n_iPulseOutputChannel	Dezimalkonstante	Pulsausgabekanal 0–3

Operanden

Für	Merker				T/C		Register			Konstante
s_dutDataTable	-	-	-	-	-	-	DT	-	-	-
n_iPulseOutputChannel	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ die Anfangsgeschwindigkeit > Sollgeschwindigkeit ist
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Glob. Variablen

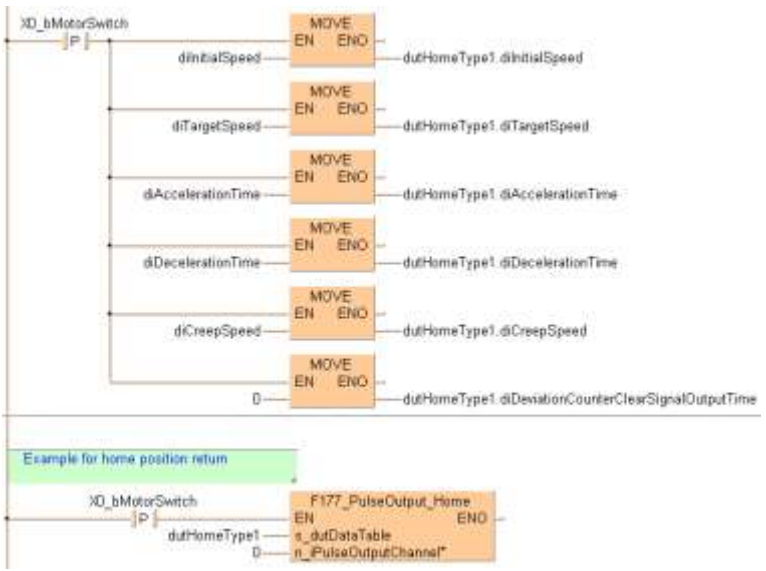
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	X0_bMotorSwitch	X0	%IX0.0	BOOL	FALSE

SDT Der SDT F177_PulseOutput_Home_Type1_DUT ist in der FP Library enthalten.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	X0_bMotorSwitch	BOOL	FALSE	at X0
1	VAR	diInitialSpeed	DINT	1000	
2	VAR	diTargetSpeed	DINT	5000	
3	VAR	diAccelerationTime	DINT	3000	
4	VAR	diDecelerationTime	DINT	3000	
5	VAR	diCreepSpeed	DINT	5000	
6	VAR	dutHomeType1	F177_PulseOutput_Home_Type1_DUT	dwControlCode := 16#0012, diInitialSpeed := 0, diTargetSpeed := 0, diAccelerationTime := 0, diDecelerationTime := 0, diCreepSpeed := 0	For ControlCode (16#0012): 1 = Forward 2 = Pulse/Sign forward on
7	VAR				

KOP



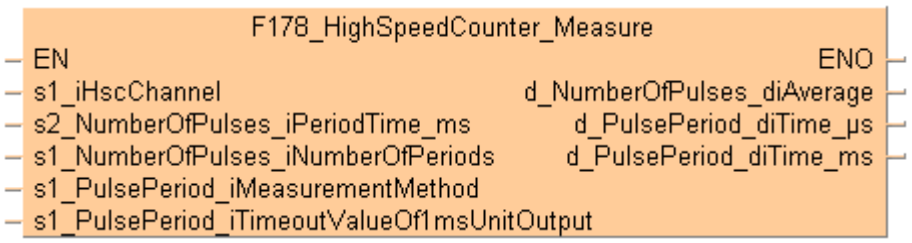
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(X0_bMotorSwitch) THEN
  dutHomeType1.diInitialSpeed:=diInitialSpeed;
  dutHomeType1.diTargetSpeed:=diTargetSpeed;
  dutHomeType1.diAccelerationTime:=diAccelerationTime;
  dutHomeType1.diDecelerationTime:=diDecelerationTime;
  dutHomeType1.diCreepSpeed:=diCreepSpeed;
  dutHomeType1.diDeviationCounterClearSignalOutputTime:=0;
END_IF;

(*Example for home position return*)
IF DF(X0_bMotorSwitch) THEN
  F177_PulseOutput_Home(s_dutDataTable := dutHomeType1,
    n_iPulseOutputChannel := 0);
END_IF;
```

F178_HighSpeedCounter_Measure

Eingangspulsmessung

Erklärung Diese Befehle messen die Anzahl der Eingangspulse innerhalb einer angegebenen Zähl- und Pulsperiode.



Merkmale der Eingangsimpulsmessung

- Für die Eingangsimpulsmessung müssen die Kanalnummer, die Zählperiode (1ms–5s) und die Anzahl der Zählperioden (1–5) angegeben werden. Diese Parameter werden zur Berechnung der durchschnittlichen Anzahl der Eingangspulse pro Zählperiode verwendet.
- Die Einheit für die Messung der Pulsperiode ([µs] ,[ms] oder beide) lässt sich angeben.
- Wenn die Messung in µs erfolgt, wird die Pulsperiode gemessen und sofort bei der Ausführung des Befehls ausgegeben. Ein Maximalwert von ca. 174,4ms ist messbar.
- Wenn die Messung in ms erfolgt, wird der Wert der Pulsperiode nach jeder Messung aktualisiert. Ein Maximalwert von ca. 49,7 Tagen ist messbar. Wenn die Messung noch nicht abgeschlossen ist, kann die Wartezeit eingestellt werden, nachdem die gemessene Pulsperiode auf -1 gesetzt wurde.
- Während der ersten Zählperioden nach dem Befehlsstart, wird -1 ausgegeben, bis die angegebene Anzahl von Zählperioden erreicht ist.
- Wenn die Pulsperiode den messbaren Bereich übersteigt oder wenn die Messung noch nicht abgeschlossen ist, wird die gemessene Pulsperiode auf -1 gesetzt.

■ Allgemeine Programmierinformation

- Wählen Sie den Eingang des schnellen Zählers für den gewünschten Kanal in den Systemregistern aus.
- Um die Pulsmessung mit diesem Befehl auszuführen, muss die Ausführungsbedingung auf TRUE gesetzt bleiben.
- Um die Messung zu stoppen, setzen Sie die Ausführungsbedingung auf FALSE.
- Wenn dieser Befehl ausgeführt wird, schaltet der Kontrollmerker für den benutzten Kanal (z.B. sys_bIsHscChannel0ControlActive) auf TRUE. Andere schnelle Zählerbefehle, die denselben Kanal nutzen, können nicht ausgeführt werden, solange der Kontrollmerker auf TRUE steht.
- Dieser Befehl kann gleichzeitig auf maximal zwei Kanälen ausgeführt werden.
- Wenn sowohl das Hauptprogramm als auch das Interrupt-Programm Code für denselben Kanal enthalten, dürfen die Programme nicht gleichzeitig ausgeführt werden.
- Der Zustand des Kontrollmerkers für den schnellen Zähler oder den Pulsausgang kann sich innerhalb eines Zyklus ändern. Zum Beispiel: Wenn der Merker mehr als einmal als Eingangsbedingung verwendet wurde, existieren eventuell verschiedene Merkerzustände innerhalb eines Zyklus. Damit das Programm ordnungsgemäß ausgeführt wird, sollten Sie daher den Zustand des Sondermerkers in eine Variable am Programmanfang kopieren.

SPS-Typen **Verfügbarkeit von F178_HighSpeedCounter_Measure (s. S. 1188)**

Datentypen

Variable	Datentyp	Unterstützte Funktionen
s1_iHscChannel	INT	Kanal des schnellen Zählers: 0–5
s2_NumberOfPulses_iPeriodTime_ms	INT	Zählperiode [ms]: 1–5000 (1ms–5s).
s1_NumberOfPulses_iNumberOfPeriods	INT	Anzahl der Zählperioden: 1–5
s1_PulsePeriod_iMeasurementMethod	INT	Einheit der Pulsperiodenmessung 0: Pulsperiode wird nicht gemessen 1: Die Pulsperiode wird in µs gemessen 2: Die Pulsperiode wird in ms gemessen 3: Die Pulsperiode wird in µs und ms gemessen:
s1_PulsePeriod_iTimeoutValueOf1msUnitOutput	INT	Wert für Zeitüberschreitung der Pulsperiodenmessung [ms]: 0: Keine Zeitüberschreitung 1: 100ms 6: 1s 2: 200ms 7: 2s 3: 300ms 8: 10s 4: 500ms 9: 60s
d_NumberOfPulses_diAverage	DINT	Durchschnittsanzahl der Pulse pro Zählperiode (Anzahl der Pulse pro Zählperiode/Anzahl der Zählperioden)
d_PulsePeriod_diTime_µs	DINT	Pulsperiode [µs]
d_PulsePeriod_diTime_ms	DINT	Pulsperiode [ms]

Operanden

Für	Merker				T/C		Register			Konstante
s1_iHscChannel	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez. oder hex.
s1/s2 Eingänge:	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez. oder hex.
d Ausgänge	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ Kanalnummer oder Werte der Datentabelle außerhalb des zulässigen Bereichs liegen ▪ die Pulsausgabe nicht in den Systemregistern gesetzt wurde ▪ der Kanal des schnellen Zählers bereits von einem anderen schnellen Zähler oder einem Pulsausgabebefehl verwendet wird ▪ die Anzahl der Kanäle 3 oder mehr beträgt
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bStartMeasurement	BOOL	FALSE
1	VAR	diNumberOfPulses	DINT	0
2	VAR	diPulsePeriodTime_µs	DINT	0
3	VAR	diPulsePeriodTime_ms	DINT	0

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF (bStartMeasurement) THEN
  F178_HighSpeedCounter_Measure(s1_iHscChannel := 0,
              s2_NumberOfPulses_iPeriodTime_ms := 10,
              s1_NumberOfPulses_iNumberOfPeriods := 5,
              s1_PulsePeriod_iMeasurementMethod :=
SYS_F178_HSC_MEASUREMENT_µs_ms,
              s1_PulsePeriod_iTimeoutValueOf1msUnitOutput := 0,
              d_NumberOfPulses_diAverage => diNumberOfPulses,
              d_PulsePeriod_diTime_µs => diPulsePeriodTime_µs,
              d_PulsePeriod_diTime_ms => diPulsePeriodTime_ms);
END_IF;
```

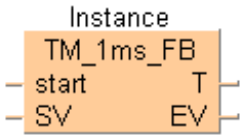
Kapitel 29

Zeitgeber

TM_1ms_FB

Zeitgeber zur Zeitbasis 0,001s einschaltverzögert (0 bis 32,767s)

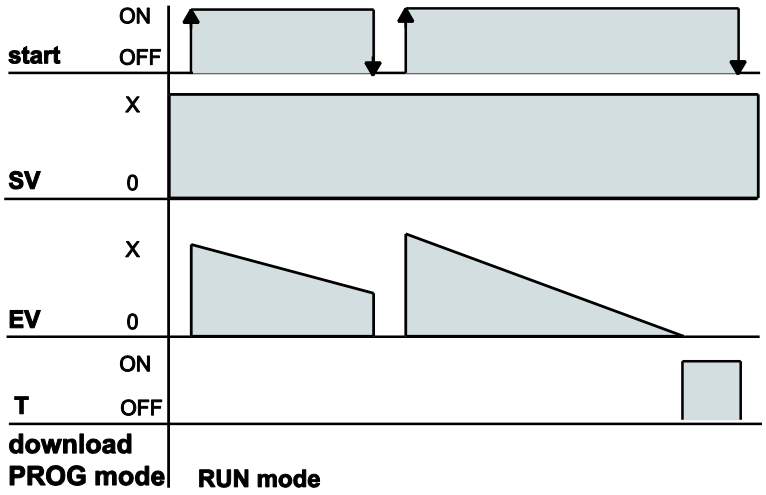
Erklärung Dieser Zeitgeber mit der Zeitbasis von 0,001s arbeitet als Einschaltverzögerung (0 bis 32,767s). Ist die Eingangsbedingung **start** des Zeitgeberbausteins erfüllt, wird die programmierte Zeit **SV** (Set value = Sollwert) gestartet. Nach Ablauf der Zeit schaltet der zugehörige Zeitgeberkontakt **T** ein.



Für den Funktionsbaustein TM_1ms_FB deklarieren Sie:

start	**Startsignal** bei jeder steigenden Flanke wird der Zielwert SV auf den Istwert EV kopiert und der Zeitgeber gestartet
SV	**Zielwert** hier definieren Sie die Einschaltverzögerungszeit (0 bis 32,767s)
T	**Signalausgang** wird gesetzt, wenn die an SV definierte Zeitspanne abgelaufen ist, d.h., wenn der Istwert EV den Wert 0 erreicht hat
EV	**Istwert** ist der Zählwert, der bei laufendem Zeitgeber alle 0,001s um 1 verringert wird

Zeitdia-gramm



☞

- **Die Anzahl der verfügbaren Zeitgeber ist von den Einstellungen in den Systemregistern 5 und 6 abhängig.**
- **Die System-Zeitgeber-Funktionen (TM_1s, TM_100ms, TM_10ms und TM_1ms) verwenden denselben Num*-Adressbereich (Num*-Eingang) wie die System-Zeitgeber-Funktionsbausteine (TM_1s_FB, TM_100ms_FB, TM_10ms_FB und TM_1ms_FB). Für die Zeitgeber-Funktionsbausteine vergibt der Compiler an jede FB-Instanz automatisch eine NUM*-Adresse. Die Vergabe erfolgt abwärtszählend, beginnend mit der höchstmöglichen Adresse. Um Fehlern (Adresskonflikten) vorzubeugen, sollten Sie diese Zeitgeber-Funktionen und -Funktionsbausteine nicht gleichzeitig in einem Projekt verwenden.**

SPS-Typen **Verfügbarkeit von TM_1ms_FB (s. S. 1197)**

Datentypen

Variable	Datentyp	Funktion
start	BOOL	Startsignal
SV	INT, WORD	Zielwert
T	BOOL	Signalausgang
EV	INT, WORD	Istwert

Operanden

Für	Merker				T/C		Register			Konstante
start	X	Y	R	L	T	C	-	-	-	-
T	-	Y	R	L	-	-	-	-	-	-
SV, EV	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins TOF verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **Alarm_control** abgespeichert und ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Alarm_control	TM_1ms_FB	
1	VAR	Start_contact	BOOL	FALSE
2	VAR	Alarm_Relay_1	BOOL	FALSE
3	VAR	Alarm_Relay_2	BOOL	FALSE

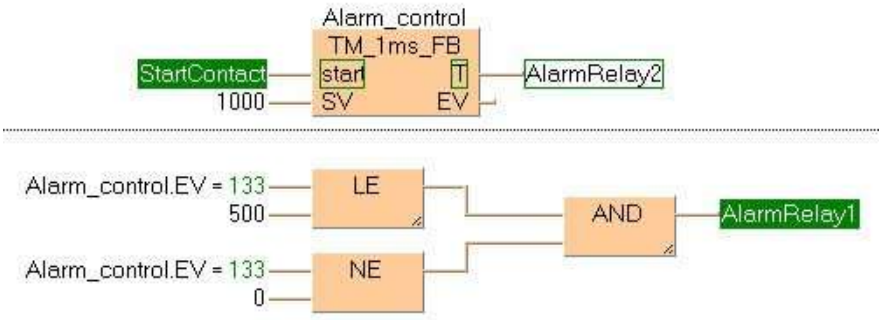
In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf

Sobald die Variable **Start_contact** gesetzt wird (Status = TRUE), wird der Zeitgeber **Alarm_control** gestartet. Die Variable **EV** des Zeitgebers wird auf den Wert von **SV** gesetzt. Solange **Start_contact** den Zustand TRUE behält, wird alle 1ms der Wert 1 von **EV** abgezogen. Erreicht **EV** den Wert 0 (nach 1 Sekunden, da SV = 1000 mit Zeitgeber-Typ TM_1ms_FB), wird **Alarm_Relay_2** auf TRUE gesetzt.

Sobald der Wert der Variable **EV** des Zeitgebers kleiner oder gleich 500 ist (nach 0,5 Sekunden) und **EV** nicht 0 ist, wird **Alarm_Relay_1** auf TRUE gesetzt.

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

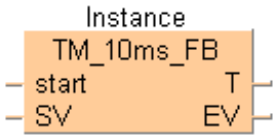
```
Alarm_Control( start:= Start_Contact ,
        sv:= 1000,
        T=> Alarm_Relay_2 ,
        EV=> Alarm_Control.EV );
(*The ON-delay time is 1000ms*)
Alarm_Relay_1:= Alarm_Control.EV <= 500 & Alarm_Control.EV <> 0;
(*Alarm_Relay_1 is set to TRUE after 500ms*)
```

TM_10ms_FB

Zeitgeber zur Zeitbasis 0,01s einschaltverzögert (0 bis 327,67s)

Erklärung

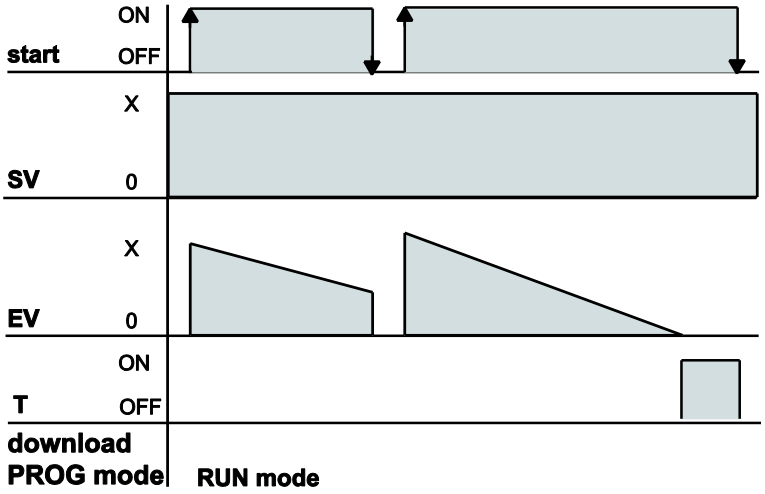
Dieser Zeitgeber mit der Zeitbasis von 0,01s arbeitet als Einschaltverzögerung (0 bis 32,767s). Ist die Eingangsbedingung **start** des Zeitgeberbausteins erfüllt, wird die programmierte Zeit **SV** (Set value = Sollwert) gestartet. Nach Ablauf der Zeit schaltet der zugehörige Zeitgeberkontakt **T** ein.



Für den Funktionsbaustein TM_10ms_FB deklarieren Sie:

start	**Startsignal** bei jeder steigenden Flanke wird der Zielwert **SV** auf den Istwert **EV** kopiert und der Zeitgeber gestartet
SV	**Zielwert** hier definieren Sie die Einschaltverzögerungszeit (0 bis 327,67s)
T	**Signalausgang** wird gesetzt, wenn die an **SV** definierte Zeitspanne abgelaufen ist, d.h., wenn der Istwert **EV** den Wert 0 erreicht hat
EV	**Istwert** ist der Zählwert, der bei laufendem Zeitgeber alle 0,01s um 1 verringert wird

Zeitdia-gramm



- **Die Anzahl der verfügbaren Zeitgeber ist von den Einstellungen in den Systemregistern 5 und 6 abhängig.**
- **Die System-Zeitgeber-Funktionen (TM_1s, TM_100ms, TM_10ms und TM_1ms) verwenden denselben Num*-Adressbereich (Num*-Eingang) wie die System-Zeitgeber-Funktionsbausteine (TM_1s_FB, TM_100ms_FB, TM_10ms_FB und TM_1ms_FB). Für die Zeitgeber-Funktionsbausteine vergibt der Compiler an jede FB-Instanz automatisch eine NUM*-Adresse. Die Vergabe erfolgt abwärtszählend, beginnend mit der höchstmöglichen Adresse. Um Fehlern (Adresskonflikten) vorzubeugen, sollten Sie diese Zeitgeber-Funktionen und -Funktionsbausteine nicht gleichzeitig in einem Projekt verwenden.**

SPS-Typen **Verfügbarkeit von TM_10ms_FB (s. S. 1197)**

Datentypen

Variable	Datentyp	Funktion
start	BOOL	Startsignal
SV	INT, WORD	Zielwert
T	BOOL	Signalausgang
EV	INT, WORD	Istwert

Operanden

Für	Merker				T/C		Register			Konstante
start	X	Y	R	L	T	C	-	-	-	-
T	-	Y	R	L	-	-	-	-	-	-
SV, EV	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins TM_10ms_FB verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **Alarm_control** abgespeichert und ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Alarm_control	TM_10ms_FB	
1	VAR	Start_contact	BOOL	FALSE
2	VAR	Alarm_Relay_1	BOOL	FALSE
3	VAR	Alarm_Relay_2	BOOL	FALSE

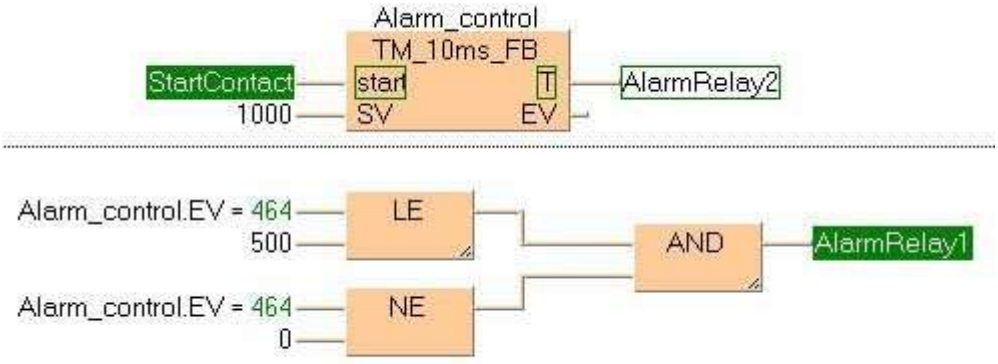
In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf

Sobald die Variable **Start_contact** gesetzt wird (Status = TRUE), wird der Zeitgeber **Alarm_control** gestartet. Die Variable **EV** des Zeitgebers wird auf den Wert von **SV** gesetzt. Solange **Start_contact** den Zustand TRUE behält, wird alle 10ms der Wert 1 von **EV** abgezogen. Erreicht **EV** den Wert 0 (nach 10 Sekunden, da SV = 1000 mit Zeitgeber-Typ TM_10ms_FB), wird **Alarm_Relay_2** auf TRUE gesetzt.

Sobald der Wert der Variable **EV** des Zeitgebers kleiner oder gleich 500 ist (nach 5 Sekunden) und **EV** nicht 0 ist, wird **Alarm_Relay_1** auf TRUE gesetzt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

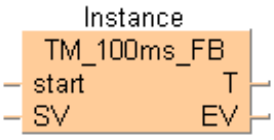
```
Alarm_Control( start:= Start_Contact ,
        SV:= 1000,
        T=> Alarm_Relay_2 ,
        EV=> Alarm_Control.EV );
(*The ON-delay time is 10s*)

Alarm_Relay_1:= Alarm_Control.EV <= 500 & Alarm_Control.EV <> 0;
(*Alarm_Relay_1 is set to TRUE after 5s*)
```

TM_100ms_FB

Zeitgeber zur Zeitbasis 0,1s; einschaltverzögert (0 bis 3276,7s)

Erklärung

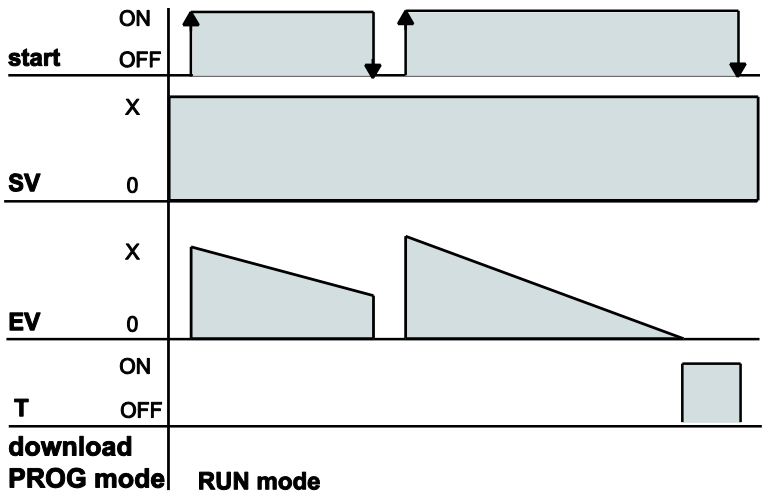
Dieser Zeitgeber mit der Zeitbasis von 0,1s arbeitet als Einschaltverzögerung (0 bis 3276,7s). Ist die Eingangsbedingung **start** des Zeitgeberbausteins erfüllt, wird die programmierte Zeit **SV** (Set value = Sollwert) gestartet. Nach Ablauf der Zeit schaltet der zugehörige Zeitgeberkontakt **T** ein.



Für den Funktionsbaustein TM_100ms_FB deklarieren Sie:

start	**Startsignal** bei jeder steigenden Flanke wird der Zielwert **SV** auf den Istwert **EV** kopiert und der Zeitgeber gestartet
SV	**Zielwert** hier definieren Sie die Einschaltverzögerungszeit (0 bis 3276,7s)
T	**Signalausgang** wird gesetzt, wenn die an **SV** definierte Zeitspanne abgelaufen ist, d.h., wenn der Istwert **EV** den Wert 0 erreicht hat
EV	**Istwert** ist der Zählwert, der bei laufendem Zeitgeber alle 0,1s um 1 verringert wird

Zeitdia-gramm:



- **Die Anzahl der verfügbaren Zeitgeber ist von den Einstellungen in den Systemregistern 5 und 6 abhängig.**
- **Die System-Zeitgeber-Funktionen (TM_1s, TM_100ms, TM_10ms und TM_1ms) verwenden denselben Num*-Adressbereich (Num*-Eingang) wie die System-Zeitgeber-Funktionsbausteine (TM_1s_FB, TM_100ms_FB, TM_10ms_FB und TM_1ms_FB). Für die Zeitgeber-Funktionsbausteine vergibt der Compiler an jede FB-Instanz automatisch eine NUM*-Adresse. Die Vergabe erfolgt abwärtszählend, beginnend mit der höchstmöglichen Adresse. Um Fehlern (Adresskonflikten) vorzubeugen, sollten Sie diese Zeitgeber-Funktionen und -Funktionsbausteine nicht gleichzeitig in einem Projekt verwenden.**

SPS-Typen **Verfügbarkeit von TM_100ms_FB (s. S. 1197)**

Datentypen

Variable	Datentyp	Funktion
start	BOOL	Startsignal
SV	INT, WORD	Zielwert
T	BOOL	Signalausgang
EV	INT, WORD	Istwert

Operanden

Für	Merker				T/C		Register			Konstante
start	X	Y	R	L	T	C	-	-	-	-
T	-	Y	R	L	-	-	-	-	-	-
SV, EV	-	WY	WR	WL	SV	EV	DT	LD	FL	-

▪

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins TM_100ms_FB verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **Alarm_control** abgespeichert und ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Alarm_Control	TM_100ms_FB	
1	VAR	Start_Contact	BOOL	FALSE
2	VAR	Alarm_Relay_1	BOOL	FALSE
3	VAR	Alarm_Relay_2	BOOL	FALSE

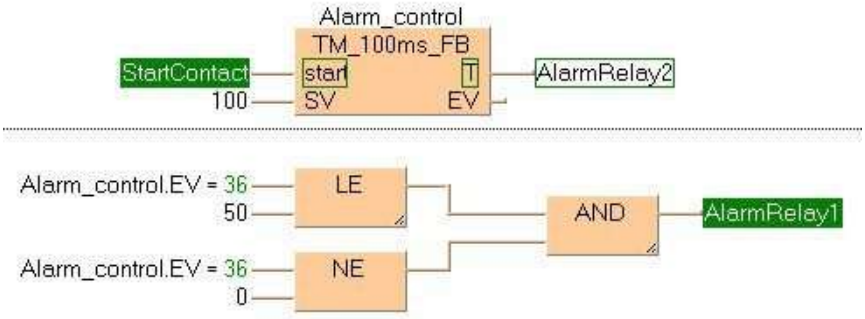
In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf

Sobald die Variable **Start_contact** gesetzt wird (Status = TRUE), wird der Zeitgeber **Alarm_control** gestartet. Die Variable **EV** des Zeitgebers wird auf den Wert von **SV** gesetzt. Solange **Start_contact** den Zustand TRUE behält, wird alle 100ms der Wert 1 von **EV** abgezogen. Erreicht **EV** den Wert 0 (nach 10 Sekunden, da **SV** = 100 mit Zeitgeber-Typ TM_100ms_FB), wird **Alarm_Relay_2** auf TRUE gesetzt.

Sobald der Wert der Variable **EV** des Zeitgebers kleiner oder gleich 50 ist (nach 5 Sekunden) und **EV** nicht 0 ist, wird **Alarm_Relay_1** auf TRUE gesetzt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

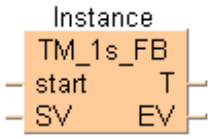
```
Alarm_Control( start:= Start_Contact ,
        SV:= 100,
        T=> Alarm_Relay_2 ,
        EV=> Alarm_Control.EV );
(*The ON-delay time is 10s*)
Alarm_Relay_1:= Alarm_Control.EV <= 50 & Alarm_Control.EV <> 0;
(*Alarm_Relay_1 is set to TRUE after 5s*)
```

TM_1s_FB

Zeitgeber zur Zeitbasis 1s einschaltverzögert (0 bis 32767s)

Erklärung

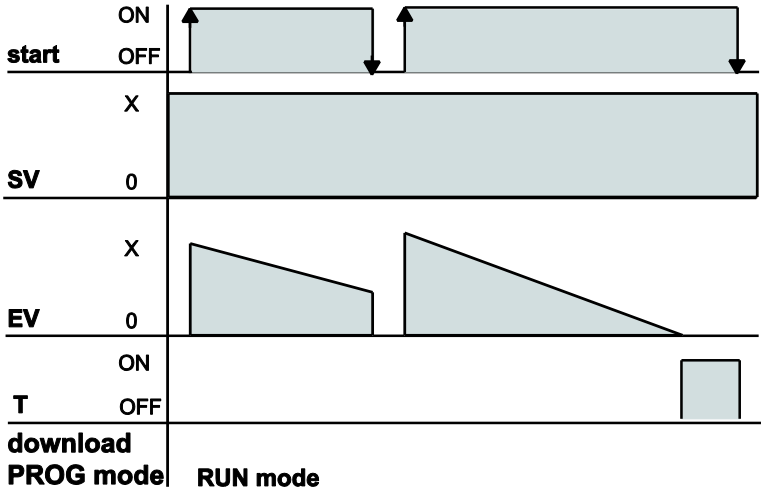
Dieser Zeitgeber mit der Zeitbasis von 1s arbeitet als Einschaltverzögerung (0 bis 32767s). Ist die Eingangsbedingung **start** des Zeitgeberbausteins erfüllt, wird die programmierte Zeit **SV** (Set value = Sollwert) gestartet. Nach Ablauf der Zeit schaltet der zugehörige Zeitgeberkontakt **T** ein.



Für den Funktionsbaustein TM_1s_FB deklarieren Sie:

start	**Startsignal** bei jeder steigenden Flanke wird der Zielwert **SV** auf den Istwert **EV** kopiert und der Zeitgeber gestartet
SV	**Zielwert** hier definieren Sie die Einschaltverzögerungszeit (0 bis 32767s)
T	**Signalausgang** wird gesetzt, wenn die an SV definierte Zeitspanne abgelaufen ist, d.h., wenn der Istwert **EV** den Wert 0 erreicht hat
EV	**Istwert** ist der Zählwert, der bei laufendem Zeitgeber alle 1s um 1 verringert wird

Zeit-diagramm



- **Die Anzahl der verfügbaren Zeitgeber ist von den Einstellungen in den Systemregistern 5 und 6 abhängig.**
- **Die System-Zeitgeber-Funktionen (TM_1s, TM_100ms, TM_10ms und TM_1ms) verwenden denselben Num*-Adressbereich (Num*-Eingang) wie die System-Zeitgeber-Funktionsbausteine (TM_1s_FB, TM_100ms_FB, TM_10ms_FB und TM_1ms_FB). Für die Zeitgeber-Funktionsbausteine vergibt der Compiler an jede FB-Instanz automatisch eine NUM*-Adresse. Die Vergabe erfolgt abwärtszählend, beginnend mit der höchstmöglichen Adresse. Um Fehlern (Adresskonflikten) vorzubeugen, sollten Sie diese Zeitgeber-Funktionen und -Funktionsbausteine nicht gleichzeitig in einem Projekt verwenden.**

SPS-Typen **Verfügbarkeit von TM_1s_FB (s. S. 1198)**

Datentypen

Variable	Datentyp	Funktion
start	BOOL	Startsignal
SV	INT, WORD	Zielwert
T	BOOL	Signalausgang
EV	INT, WORD	Istwert

Operanden

Für	Merker				T/C		Register			Konstante
start	X	Y	R	L	T	C	-	-	-	-
T	-	Y	R	L	-	-	-	-	-	-
SV, EV	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung des Funktionsbausteins TM_1s_FB verwendet werden. Dazu zählt auch der Funktionsbaustein (FB) selbst. Mit der Deklaration des FB legen Sie eine Kopie von dem Original-FB an. Diese Kopie wird unter **Alarm_control** abgespeichert und ein eigener Datenbereich reserviert.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Alarm_control	TM_1s_FB	
1	VAR	Start_contact	BOOL	FALSE
2	VAR	Alarm_Relay_1	BOOL	FALSE
3	VAR	Alarm_Relay_2	BOOL	FALSE

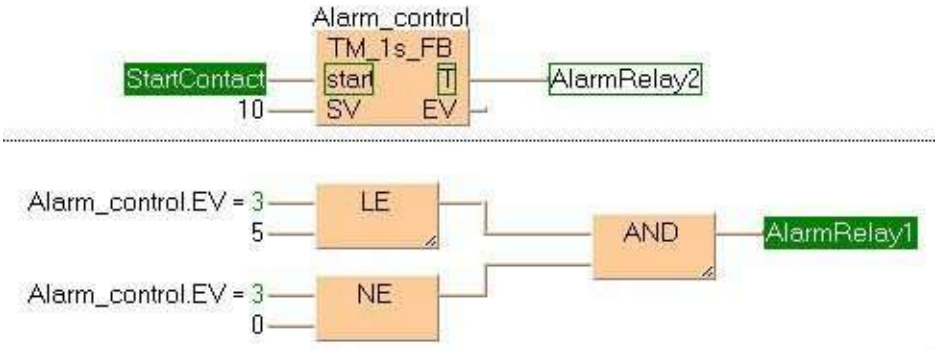
In diesem Beispiel werden Variablen verwendet. Sie können als Eingangsvariable auch eine Konstante verwenden.

Rumpf

Sobald die Variable **Start_contact** gesetzt wird (Status = TRUE), wird der Zeitgeber **Alarm_control** gestartet. Die Variable **EV** des Zeitgebers wird auf den Wert von **SV** gesetzt. Solange **Start_contact** den Zustand TRUE behält, wird alle 1s der Wert 1 von **EV** abgezogen. Erreicht **EV** den Wert 0 (nach 10 Sekunden, da **SV** = 10 mit Zeitgeber-Typ TM_1s_FB), wird **Alarm_Relay_2** auf TRUE gesetzt.

Sobald der Wert der Variable **EV** des Zeitgebers kleiner oder gleich 5 ist (nach 5 Sekunden) und **EV** nicht 0 ist, wird **Alarm_Relay_1** auf TRUE gesetzt.

KOP



ST

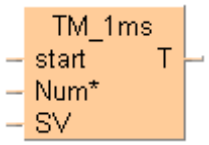
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Alarm_Control( start:= Start_Contact ,
        SV:= 10,
        T=> Alarm_Relay_2 ,
        EV=> Alarm_Control.EV );
(*The ON-delay time is 10s*)
Alarm_Relay_1:= Alarm_Control.EV <= 5 & Alarm_Control.EV <> 0;
(*Alarm_Relay_1 is set to TRUE after 5s*)
```

TM_1ms

Zeitgeber zur Zeitbasis 0,001s einschaltverzögert (0 bis 32,767s)

Erklärung Dieser Zeitgeber mit der Zeitbasis von 0,001s arbeitet als Einschaltverzögerung (0 bis 32,767s).



Die für den Befehl verwendeten Bereiche sind:

- voreingestellter Wertebereich: **SV** Preset (Set)
- tatsächlich abgelaufener Wertebereich: **EV** Count (Elapsed)

Wenn der Modus auf RUN steht, wird die voreingestellte Zeit an **SV** übertragen. Wenn der Trigger **start** des Zeitgebers auf TRUE gesetzt ist, wird die voreingestellte Zeit von **SV** nach **EV** übertragen.

Während der Zeitberechnung wird diese Zeit von **EV** abgezogen.

Die Zykluszeit wird ebenfalls im nächsten Zyklus von **EV** abgezogen.

Der Zeitgeberkontakt **T** schaltet ein, wenn **EV** 0 wird.

Berechnung der Schaltzeit:

Schaltzeit = Sollzeit - 0 bis 1/2 Zeitbasis (0,5ms) + Zykluszeit

Beispiel:

150ms Sollzeit und 8ms SPS-Zykluszeit

Obere Grenze = 150 - 0 + 8 = 158ms
Untere Grenze = 150 -0,5 +8 = 157,5ms

Dies ergibt einen Zeitbereich von 157,5ms bis 158ms.

SPS-Typen **Verfügbarkeit von TM_1ms (s. S. 1197)**

Datentypen

Variable	Datentyp	Funktion
start	BOOL	Startet Zeitgeber
Num*	ANY16	Zeitgeberkontakt muss eine Konstante sein
SV		Zeitgeberadresse im Systemregister 5 und 6
T	BOOL	Zielwert

Operanden

Für	Merker				T/C		Register			Konstante
start	X	Y	R	L	T	C	-	-	-	-
T	-	Y	R	L	-	-	-	-	-	-
Num*	-	-	-	-	-	-	-	-	-	dez., hex.
SV	-	-	-	-	SV	-	-	-	-	dez., hex.

☞

- **Diese Funktion kann in einem Funktionsblock nicht verwendet werden.**
- **Um die Richtigkeit der Ergebnisse zu gewährleisten, muss sichergestellt sein, dass jede Zeitgeber-Funktion und jeder Zeitgeber-Funktionsbaustein in jedem Zyklus nur genau einmal ausgeführt wird. Aus diesem Grund ist es nicht zulässig, Zeitgeber-Funktionen oder Funktionsbausteine in Interrupt-Programmen oder in Schleifen zu verwenden.**
- **Jeder der verwendeten Zeitgeber muss eine eigene Konstante Num* haben.**
  Die verfügbaren Num*-Adressen sind vom Systemregister 5 und 6 abhängig.
  Zeitgeber vom Typ TM_1s, TM_100ms, TM_10ms und TM_1ms verwenden denselben Num*-Adressbereich (Num*-Eingang).
- **Die System-Zeitgeber-Funktionen (TM_1s, TM_100ms, TM_10ms und TM_1ms) verwenden denselben Num*-Adressbereich (Num*-Eingang) wie die System-Zeitgeber-Funktionsbausteine (TM_1s_FB, TM_100ms_FB, TM_10ms_FB und TM_1ms_FB). Für die Zeitgeber-Funktionsbausteine vergibt der Compiler an jede FB-Instanz automatisch eine NUM*-Adresse. Die Vergabe erfolgt abwärts zählend beginnend mit der höchstmöglichen Adresse. Um Fehlern (Adresskonflikten) vorzubeugen, sollten Sie diese Zeitgeber-Funktionen und -Funktionsbausteine nicht gleichzeitig in einem Projekt verwenden.**
- **Diese Funktion erfordert am Ausgang "T" keine Variable.**

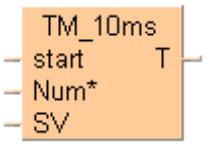
Beispiel Bitte lesen Sie hierzu die Beschreibung zu TM_1ms_FB (s. S. 953).

TM_10ms

Zeitgeber zur Zeitbasis 0,01s einschaltverzögert (0 bis 327,67s)

Erklärung

Dieser Zeitgeber mit der Zeitbasis von 0,01s arbeitet als Einschaltverzögerung (0 bis 327,67s).



Der Befehl verwendet folgende Bereiche:

- voreingestellter Wertebereich: **SV** Preset (Set)
- tatsächlich abgelaufener Wertebereich: **EV** Count (Elapsed)

Beim Wechseln in den RUN-Modus wird die eingestellte Sollzeit in **SV** übertragen. Wenn der Trigger **start** des Zeitgebers auf TRUE gesetzt ist, wird die voreingestellte Zeit von **SV** nach **EV** übertragen.

Während der Zeitberechnung wird diese Zeit von **EV** abgezogen.

Die Zykluszeit wird ebenfalls im nächsten Zyklus von **EV** abgezogen.

Der Zeitgeberkontakt **T** schaltet ein, wenn **EV** 0 wird.

Berechnung der Schaltzeit:

Schaltzeit = Sollzeit - 0 bis 1/4 Zeitbasis (2,5ms) + Zykluszeit

Beispiel:

150ms Sollzeit und 8ms SPS-Zykluszeit

Obere Grenze = 150 - 0 + 8 = 158ms
Untere Grenze = 150 - 2,5 + 8 = 155,5ms

Dies ergibt einen Zeitbereich von 155,5ms bis 158ms.

SPS-Typen

Verfügbarkeit von TM_10ms (s. S. 1197)

Datentypen

Variable	Datentyp	Funktion
start	BOOL	Startet Zeitgeber
Num*	ANY16	Zeitgeberadresse im Systemregister 5 und 6 muss eine Konstante sein
SV		Zielwert
T	BOOL	Zeitgeberkontakt

Operanden

Für	Merker				T/C		Register			Konstante
start	X	Y	R	L	T	C	-	-	-	-
T	-	Y	R	L	-	-	-	-	-	-
Num*	-	- muss eine Konstante sein	-	-	-	-	-	-	-	dez., hex.
SV	-	-	-	-	SV	-	-	-	-	dez., hex.

☞

- **Diese Funktion kann in einem Funktionsblock nicht verwendet werden.**
- **Um die Richtigkeit der Ergebnisse zu gewährleisten, muss sichergestellt sein, dass jede Zeitgeber-Funktion und jeder Zeitgeber-Funktionsbaustein in jedem Zyklus nur genau einmal ausgeführt wird. Aus diesem Grund ist es nicht zulässig, Zeitgeber-Funktionen oder Funktionsbausteine in Interrupt-Programmen oder in Schleifen zu verwenden.**
- **Jeder der verwendeten Zeitgeber muss eine eigene Konstante Num* haben.**
  Die verfügbaren Num*-Adressen sind vom Systemregister 5 und 6 abhängig.
  Zeitgeber vom Typ TM_1s, TM_100ms, TM_10ms und TM_1ms verwenden denselben Num*-Adressbereich (Num*-Eingang).
- **Die System-Zeitgeber-Funktionen (TM_1s, TM_100ms, TM_10ms und TM_1ms) verwenden denselben Num*-Adressbereich (Num*-Eingang) wie die System-Zeitgeber-Funktionsbausteine (TM_1s_FB, TM_100ms_FB, TM_10ms_FB und TM_1ms_FB). Für die Zeitgeber-Funktionsbausteine vergibt der Compiler an jede FB-Instanz automatisch eine NUM*-Adresse. Die Vergabe erfolgt abwärts zählend beginnend mit der höchstmöglichen Adresse. Um Fehlern (Adresskonflikten) vorzubeugen, sollten Sie diese Zeitgeber-Funktionen und -Funktionsbausteine nicht gleichzeitig in einem Projekt verwenden.**
- **Diese Funktion erfordert am Ausgang "T" keine Variable.**

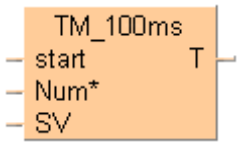
Beispiel Bitte lesen Sie hierzu die Beschreibung zu TM_10ms_FB (s. S. 955).

TM_100ms

Zeitgeber zur Zeitbasis 0,1s; einschaltverzögert (0 bis 3276,7s)

Erklärung

Dieser Zeitgeber mit der Zeitbasis von 0,1s arbeitet als Einschaltverzögerung (0 bis 3276,7s).



Der Zeitgeberbefehl **TM** ist einschaltverzögert.

Die für den Befehl verwendeten Bereiche sind:

- voreingestellter Wertebereich: **SV** Preset (Set)
- tatsächlich abgelaufener Wertebereich: **EV** Count (Elapsed)

Beim Wechseln in den RUN-Modus wird die eingestellte Sollzeit in **SV** übertragen. Wenn der Trigger **start** des Zeitgebers auf TRUE gesetzt ist, wird die voreingestellte Zeit von **SV** nach **EV** übertragen.

Während der Zeitberechnung wird diese Zeit von **EV** abgezogen.

Die Zykluszeit wird ebenfalls im nächsten Zyklus von **EV** abgezogen.

Der Zeitgeberkontakt **T** schaltet ein, wenn **EV** 0 wird.

Berechnung der Schaltzeit:

Schaltzeit = Sollzeit - 0 bis 1/4 Zeitbasis (25ms) + Zykluszeit

Beispiel:

1500ms Sollzeit und 8ms SPS-Zykluszeit

Obere Grenze = 1500 - 0 + 8 = 1508ms
Untere Grenze = 1500 - 25 + 8 = 1483ms

Dies ergibt einen Zeitbereich von 1483ms bis 1508ms.

SPS-Typen

Verfügbarkeit von TM_100ms (s. S. 1197)

Datentypen

Variable	Datentyp	Funktion
start	BOOL	Startet Zeitgeber
Num*	ANY16	Zeitgeberadresse im Systemregister 5 und 6 muss eine Konstante sein
SV		Zielwert
T	BOOL	Zeitgeberkontakt

Operanden

Für	Merker				T/C		Register			Konstante
start	X	Y	R	L	T	C	-	-	-	-
T	-	Y	R	L	-	-	-	-	-	-
Num*	-	- muss eine Konstante sein	-	-	-	-	-	-	-	dez., hex.
SV	-	-	-	-	SV	-	-	-	-	dez., hex.

☞

- **Diese Funktion kann in einem Funktionsblock nicht verwendet werden.**
- **Um die Richtigkeit der Ergebnisse zu gewährleisten, muss sichergestellt sein, dass jede Zeitgeber-Funktion und jeder Zeitgeber-Funktionsbaustein in jedem Zyklus nur genau einmal ausgeführt wird. Aus diesem Grund ist es nicht zulässig, Zeitgeber-Funktionen oder Funktionsbausteine in Interrupt-Programmen oder in Schleifen zu verwenden.**
- **Jeder der verwendeten Zeitgeber muss eine eigene Konstante Num* haben.**
  Die verfügbaren Num*-Adressen sind vom Systemregister 5 und 6 abhängig.
  Zeitgeber vom Typ TM_1s, TM_100ms, TM_10ms und TM_1ms verwenden denselben Num*-Adressbereich (Num*-Eingang).
- **Die System-Zeitgeber-Funktionen (TM_1s, TM_100ms, TM_10ms und TM_1ms) verwenden denselben Num*-Adressbereich (Num*-Eingang) wie die System-Zeitgeber-Funktionsbausteine (TM_1s_FB, TM_100ms_FB, TM_10ms_FB und TM_1ms_FB). Für die Zeitgeber-Funktionsbausteine vergibt der Compiler an jede FB-Instanz automatisch eine NUM*-Adresse. Die Vergabe erfolgt abwärts zählend beginnend mit der höchstmöglichen Adresse. Um Fehlern (Adresskonflikten) vorzubeugen, sollten Sie diese Zeitgeber-Funktionen und -Funktionsbausteine nicht gleichzeitig in einem Projekt verwenden.**
- **Diese Funktion erfordert am Ausgang "T" keine Variable.**

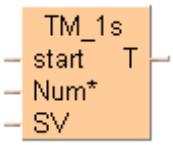
Beispiel Bitte lesen Sie hierzu die Beschreibung zu TM_100ms_FB (s. S. 958).

TM_1s

Zeitgeber zur Zeitbasis 1s einschaltverzögert (0 bis 32767s)

Erklärung

Dieser Zeitgeber mit der Zeitbasis von 1s arbeitet als Einschaltverzögerung (0 bis 32767s).



Die für den Befehl verwendeten Bereiche sind:

- voreingestellter Wertebereich: **SV** Preset (Set)
- tatsächlich abgelaufener Wertebereich: **EV** Count (Elapsed)

Beim Wechseln in den RUN-Modus wird die eingestellte Sollzeit in **SV** übertragen. Wenn der Trigger **start** des Zeitgebers auf TRUE gesetzt ist, wird die voreingestellte Zeit von **SV** nach **EV** übertragen.

Während der Zeitberechnung wird diese Zeit von **EV** abgezogen.

Die Zykluszeit wird ebenfalls im nächsten Zyklus von **EV** abgezogen.

Der Zeitgeberkontakt **T** schaltet ein, wenn **EV** 0 wird.

Berechnung der Schaltzeit:

Schaltzeit = Sollzeit - 0 bis 1/4 Zeitbasis (250ms) + Zykluszeit

Beispiel:

150s Sollzeit und 8ms SPS-Zykluszeit

Obere Grenze = 150000 - 0 + 8 = 150008ms
Untere Grenze = 150000 - 250 + 8 = 149758ms

Dies ergibt einen Zeitbereich von 149758ms bis 150008ms.

SPS-Typen

Verfügbarkeit von TM_1s (s. S. 1197)

Datentypen

Variable	Datentyp	Funktion
start	BOOL	Startet Zeitgeber
Num*	ANY16	Zeitgeberadresse im Systemregister 5 und 6 muss eine Konstante sein
SV		Zielwert
T	BOOL	Zeitgeberkontakt

Operanden

Für	Merker				T/C		Register			Konstante
start	X	Y	R	L	T	C	-	-	-	-
T	-	Y	R	L	-	-	-	-	-	-
Num*	-	-	-	-	-	-	-	-	-	dez., hex.
SV	-	-	-	-	SV	-	-	-	-	dez., hex.

☞

- **Diese Funktion kann in einem Funktionsblock nicht verwendet werden.**
- **Um die Richtigkeit der Ergebnisse zu gewährleisten, muss sichergestellt sein, dass jede Zeitgeber-Funktion und jeder Zeitgeber-Funktionsbaustein in jedem Zyklus nur genau einmal ausgeführt wird. Aus diesem Grund ist es nicht zulässig, Zeitgeber-Funktionen oder Funktionsbausteine in Interrupt-Programmen oder in Schleifen zu verwenden.**
- **Jeder der verwendeten Zeitgeber muss eine eigene Konstante Num* haben.**
  Die verfügbaren Num*-Adressen sind vom Systemregister 5 und 6 abhängig.
  Zeitgeber vom Typ TM_1s, TM_100ms, TM_10ms und TM_1ms verwenden denselben Num*-Adressbereich (Num*-Eingang).
- **Die System-Zeitgeber-Funktionen (TM_1s, TM_100ms, TM_10ms und TM_1ms) verwenden denselben Num*-Adressbereich (Num*-Eingang) wie die System-Zeitgeber-Funktionsbausteine (TM_1s_FB, TM_100ms_FB, TM_10ms_FB und TM_1ms_FB). Für die Zeitgeber-Funktionsbausteine vergibt der Compiler an jede FB-Instanz automatisch eine NUM*-Adresse. Die Vergabe erfolgt abwärts zählend beginnend mit der höchstmöglichen Adresse. Um Fehlern (Adresskonflikten) vorzubeugen, sollten Sie diese Zeitgeber-Funktionen und -Funktionsbausteine nicht gleichzeitig in einem Projekt verwenden.**
- **Diese Funktion erfordert am Ausgang "T" keine Variable.**

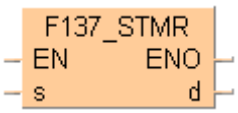
Beispiel Bitte lesen Sie hierzu die Beschreibung zu TM_1s_FB (s. S. 961).

F137_STMR

Zeitgeber 16-Bit (Zeitbasis 0,01s)

Erklärung

Der Hilfszeitgeberbefehl **F137_STMR** ist einschaltverzögert. Sie können Zeiten von 0,01s bis 327,67s programmieren. Dafür setzen Sie den Hilfszeitgeber **s**. Der Sondermerker R900D ist während der aktiven Zeit des Zeitgebers zurückgesetzt.



Der Zeitgeber funktioniert wie folgt:

- Wenn der Trigger **EN** des Hilfszeitgeberbefehls (STMR) im AN-Status ist, wird die Konstante oder der an **s** spezifizierte Wert zum Bereich bei **d** übertragen.
- Während des Zeitgeberprozesses wird die Zeit vom bei **d** spezifizierten Wert abgezogen.
- Der Ausgang ENO schaltet auf AN, wenn der bei **d** spezifizierte Wert 0 wird.

SPS-Typen

Verfügbarkeit von F137_STMR (s. S. 1187)

Datentypen

Variable	Datentyp	Funktion
s	ANY16	16-Bit-Bereich oder äquivalente Konstante für den Zeitgeber-Sollwert
d		16-Bit-Bereich für den Zeitgeber-Istwert

Die Variablen **s** und **d** müssen vom gleichen Datentyp sein.

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.
d	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel

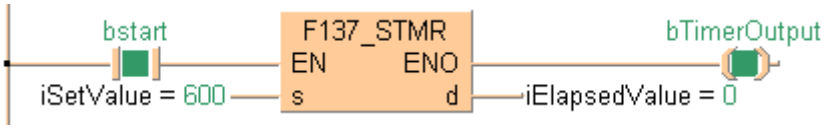
In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	bstart	BOOL	FALSE	activates the timer
1	VAR	iSetValue	INT	600	six seconds (600 * 0,01s)
2	VAR	iElapsedValue	INT	0	
3	VAR	bTimerOutput	BOOL	FALSE	set to TRUE after 6s have elapsed

KOP



F183_DSTM

Zeitgeber 32-Bit

Erklärung Der F183-Befehl aktiviert einen aufwärtszählenden 32-Bit Zeitgeber mit der kleinsten Zeiteinheit von 0,01s. Während der Ausführung von F183 (start = TRUE) wird die ablaufende Zeit zum aktuellen Zählwert **d** addiert. Wenn der Istwert **d** mit dem Zielwert **s** übereinstimmt, wird der Ausgang des Zeitgebers gesetzt. Die Ausführung wird abgebrochen und der aktuelle Zählwert **d** 0 gesetzt, sobald die Startbedingung start auf FALSE gesetzt wird. Der Zielwert **s** kann während der Ausführung von F183 geändert werden.

Die Verzögerungszeit des Zeitgebers kann mit folgender Formel berechnet werden: (Zielwert **s**) * (0,01s) = Einschaltverzögerung

SPS-Typen **Verfügbarkeit von F183_DSTM (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s	ANY32	Zielwert, Bereich von 0 bis 2147483647
d		Istwert, Bereich von 0 bis 2147483647

Operanden

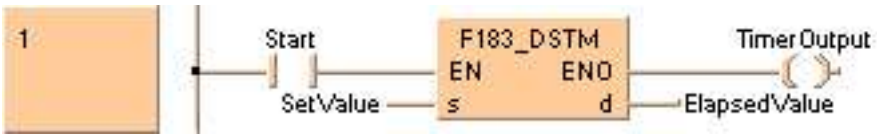
Für	Merker				T/C		Register			Konstante
s	DWX	DWY	DWR	-	DSV	DEV	DDT	-	-	dez., hex.
d	-	DWY	DWR	-	DSV	DEV	DDT	-	-	-

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf:

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	Start	BOOL	FALSE	
1	VAR	SetValue	DINT	0	10 seconds
2	VAR	TimerOutput	BOOL	FALSE	turns on when 10s have elapsed
3	VAR	ElapsedValue	DINT	0	

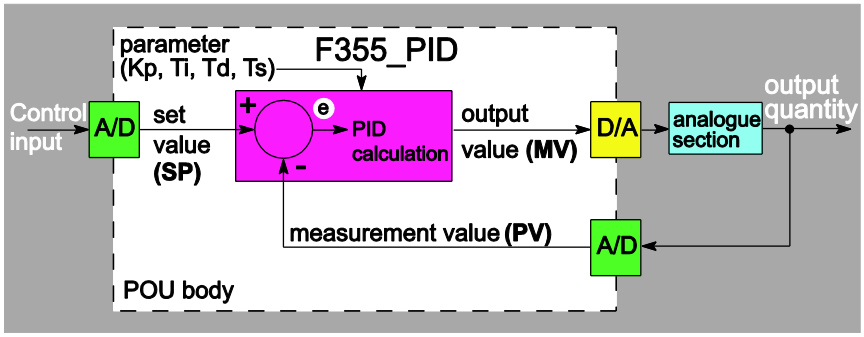
KOP



Kapitel 30

Befehle für die Regelung

30.1 Funktionsweise des PID-Reglers



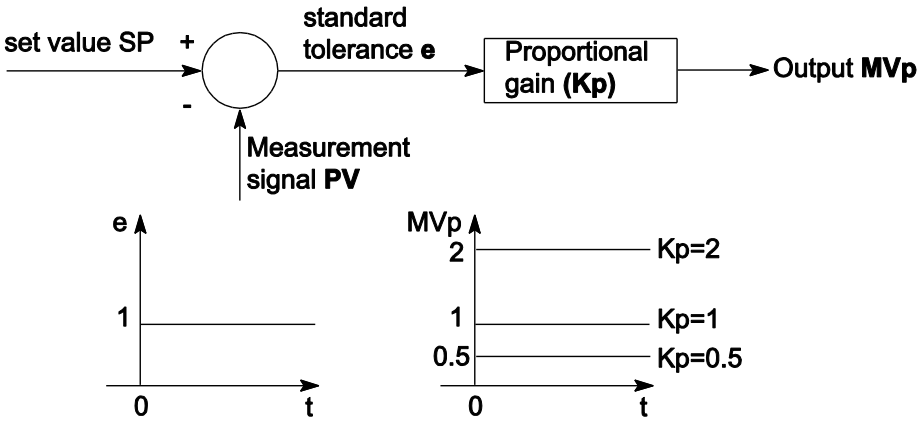
Der oben dargestellte POE-Rumpf stellt den Standardregelkreis dar. Die Führungsgröße wird vom Bediener vorgegeben (z.B. Wunschtemperatur 22°C im Raum). Nach der Analog-Digital-Wandlung wird der Sollwert (SP) dem PID-Regler als Eingangswert zugeführt. Der Istwert (PV) wird gewöhnlich von einem Messglied (Temperaturfühler) ermittelt (z.B. aktuelle Temperatur im Raum) und dem PID-Regler als Eingangswert zugeführt. Aus dem Sollwert und dem Istwert berechnet F355_PID die Regelabweichung e (e = Sollwert - Istwert). Mit den anliegenden Parametern (Proportionalbeiwert Kp, Integrierzeit Ti ...) wird in Abständen der Regelzykluszeit Ts ein neuer Ausgangswert Stellgröße (MV) berechnet. Dieser wird nach der Digital-Analog-Wandlung dem Stellglied (z.B. Ventil zur Wärmeregulierung eines Raumes) zugewiesen. Die analoge Strecke soll z.B. das System Stellglied, Heizung und Wärmeausdehnung im Raum darstellen.

Ein PID-Regler besteht aus drei Komponenten:

1. Proportionalglied (P-Anteil)

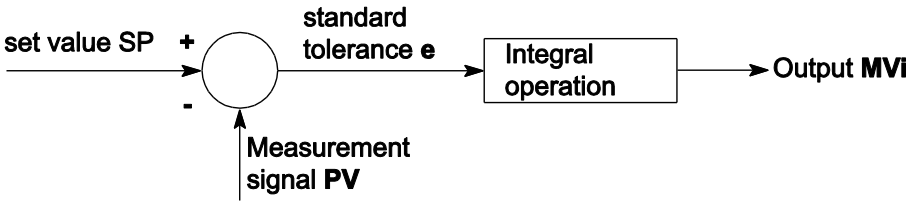
Ein Proportionalglied generiert eine Ausgangsgröße, die proportional zur Eingangsgröße ist. Der Proportionalbeiwert Kp bestimmt die Verstärkung bzw. Abschwächung der Eingangsgröße.
Ein Proportionalglied kann z.B. ein einfacher elektrischer Widerstand oder ein linearer Verstärker sein.

Der P-Anteil weist ein relativ großes maximales Überschwingen, eine große Ausregelzeit sowie eine bleibende Regelabweichung auf.

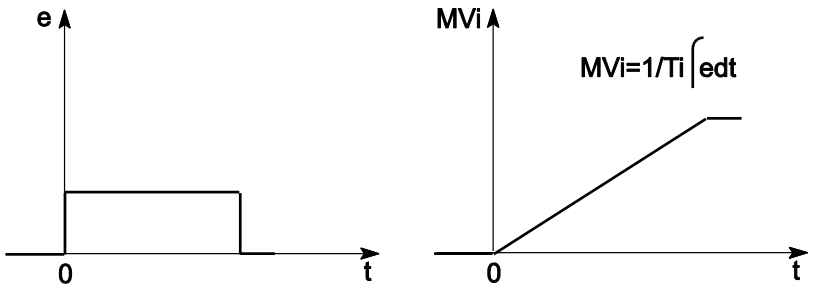


2. Integrierglied (I-Anteil)

Ein Integrierglied generiert eine Ausgangsgröße, die dem Zeitintegral der Eingangsgröße entspricht (Fläche der Eingangsgröße). Die Integrierzeit bewertet dabei die Ausgangsgröße MVi.
Ein Integrierglied kann z.B. die Mengenskala eines Tanks sein, der mit einem Volumenstrom gefüllt wird. Der I-Anteil besitzt aufgrund des langsam einsetzenden I-Verhaltens ein noch größeres maximales Überschwingen als der P-Anteil, dafür aber keine bleibende Regelabweichung.

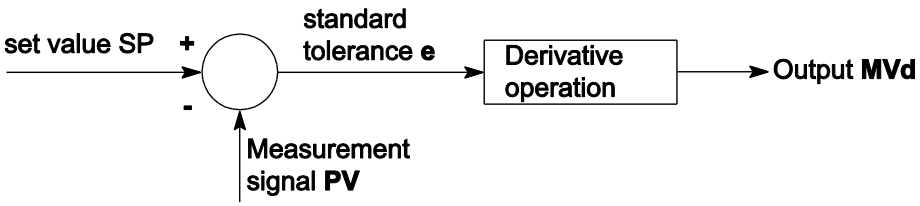


Eingangsgröße e und die generierte Ausgangsgröße MVi.

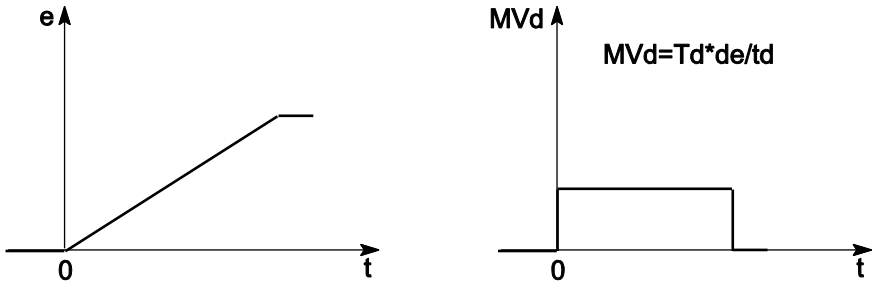


3. Differenzierglied (D-Anteil)

Ein Differenzierglied generiert eine Ausgangsgröße, die der zeitlichen Ableitung der Eingangsgröße entspricht. Die Differenzierzeit Td entspricht der Gewichtung der differenzierten Eingangsgröße.
Ein Differenzierglied kann z.B. ein RC-Spannungsteiler sein (Kondensator in Reihe und Widerstand parallel geschaltet).

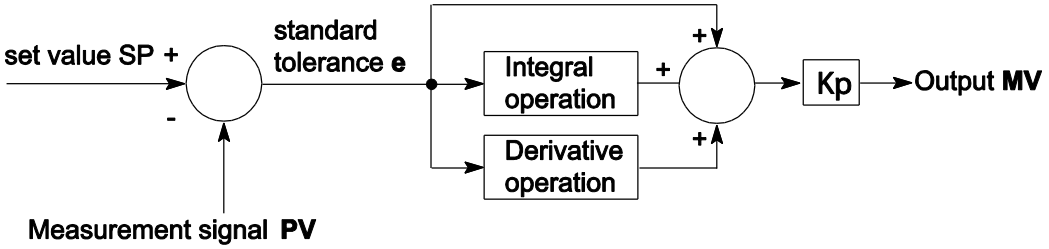


Eingangsgröße e und die generierte Ausgangsgröße MVd.



4. PID-Regler

Ein PID-Regler ist die Kombination von P-Anteil, I-Anteil und D-Anteil. Wenn die Parameter Kp, Ti und Td optimal eingestellt sind, kann ein PID-Regler eine Größe auf einen vorgegebenen Sollwert schnell regeln und halten.



Referenzgleichungen zur Berechnung der Stellgröße MV

Folgende Gleichungen werden unter den genannten Bedingungen zur Berechnung der Stellgröße MV verwendet:

Allgemein gilt:
Der Ausgangswert zum Zeitpunkt n berechnet sich aus dem vorherigen (n-1) Ausgangswert und der Änderung des Ausgangswertes in diesem Zeitintervall.

$$MV_n = MV_{n-1} + \Delta MV$$

Gegensinnige PID-Regelung: Control = 16#X000

$$\Delta MV = Kp \times \left[(e_n - e_{n-1}) + e_n \times \frac{Ts}{Ti} + \Delta D_n \right]$$

$$e_n = SP_n - PV_n$$

$$\Delta Dn = (\eta\beta - 1)D_{n-1} + \beta(PV_{n-1} - PV_n)$$

$$\eta = \frac{1}{8}(\text{constant})$$

$$\beta = \frac{Td}{(Ts + \eta Td)}$$

Gleichsinnige PID-Regelung: Control = 16#X001

$$\Delta MV = Kp \times \left[(e_n - e_{n-1}) + e_n \times \frac{Ts}{Ti} + \Delta D_n \right]$$

$$e_n = PV_n - SP_n$$

$$\Delta Dn = (\eta\beta - 1)D_{n-1} + \beta(PV_n - PV_{n-1})$$

$$\eta = \frac{1}{8}(\text{constant})$$

$$\beta = \frac{Td}{(Ts + \eta Td)}$$

Gegensinnige I-PD-Regelung: Control = 16#X002

$$\Delta MV = Kp \times \left[(PV_{n-1} - PV_n) + e_n \times \frac{Ts}{Ti} + \Delta D_n \right]$$

$$e_n = SP_n - PV_n$$

$$\Delta Dn = (\eta\beta - 1)D_{n-1} + \beta(PV_{n-1} - PV_n)$$

$$\eta = \frac{1}{8}(\text{constant})$$

$$\beta = \frac{Td}{(Ts + \eta Td)}$$

Gleichsinnige I-PD-Regelung: Control = 16#X003

$$\Delta MV = Kp \times \left[(PV_n - PV_{n-1}) + e_n \times \frac{Ts}{Ti} + \Delta D_n \right]$$

$$e_n = PV_n - SP_n$$

$$\Delta Dn = (\eta\beta - 1)D_{n-1} + \beta(PV_n - PV_{n-1})$$

$$\eta = \frac{1}{8}(\text{constant})$$

$$\beta = \frac{Td}{(Ts + \eta Td)}$$

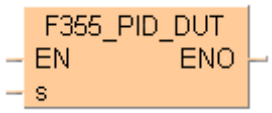
PID-Regler:

- PID_FB_DUT (s. S. 989)
- PID_FB (s. S. 987)
- F355_PID_DUT (s. S. 979)

F355_PID_DUT

PID-Regler mit Auto-Tuning

Erklärung Die PID-Regelung wird verwendet, um mit Hilfe eines gemessenen Istwerts (z.B. Temperatur) und eines vorgegebenen Sollwertes (z.B. 20°C) eine Stelleinrichtung (z.B. Heizung) zu regeln.



Die Funktion berechnet einen PID-Algorithmus, dessen Parameter als Datentabelle (d.h. in Form eines ARRAY) mit 30 Elementen am Eingang **s** eingegeben werden.

Die erforderliche Datentabelle PID_DUT_31 enthält die folgenden Parameter (detaillierte Informationen finden Sie im SDT PID_DUT_31):

Parameter	Datentyp	Funktion		
Control	WORD	Steuermodus 16#X000 Gegensinnige PI-D-Regelung 16#X001 Gleichsinnige PI-D-Regelung 16#X002 Gegensinnige I-PD-Regelung 16#X003 Gleichsinnige I-PD-Regelung		
			Bereich	**Modul**
SP	INT	Sollwert	0-10000	
PV		Istwert	0-10000	
MV		Stellgröße	0-10000	
LowerLimit		Untere Grenze der Stellgröße	0-10000	
UpperLimit		Obere Grenze der Stellgröße	1-10000	
Kp		Proportionalbeiwert	1-9999	0.1
Ti		Integralzeit	1-30000	0,1s
Td		Differenzialzeit	1-10000	0,1s
Ts		Regelzykluszeit (Sampling time)	1-6000	0,01s
AT_Progress		Auto-Tuning-Fortschritt	0-5	
Dummies	ARRAY [11..30] OF WORD	Werden intern vom PID-Regler verwendet		

SPS-Typen **Verfügbarkeit von F355_PID_DUT (s. S. 1191)**

Datentypen

Variable	Datentyp	Funktion
s	PID_DUT_31	Beschreibung der Parameter

Operanden

Für	Relais				T/C		Register			Konstante
s	-	-	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ die Parameterwerte außerhalb des zulässigen Bereichs liegen.
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

Liste der globalen Variablen

In der globalen Variablenliste werden alle globalen Ein- und Ausgangswerte deklariert, die für das Programmieren der Funktion verwendet werden.

Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial	Kommentar
0	VAR_GLOBAL	EnableAutoTuning	X1	%IX0.1	BOOL	FALSE	Switch Auto Tuning on
1	VAR_GLOBAL	Set_Value_SP	WX4	%IW4	INT	0	A/D CH0
2	VAR_GLOBAL	Process_Value_PV	WX5	%IW5	INT	0	A/D CH1
3	VAR_GLOBAL	Output_Value_MV	WY4	%QW4	INT	0	D/A

POE-Kopf

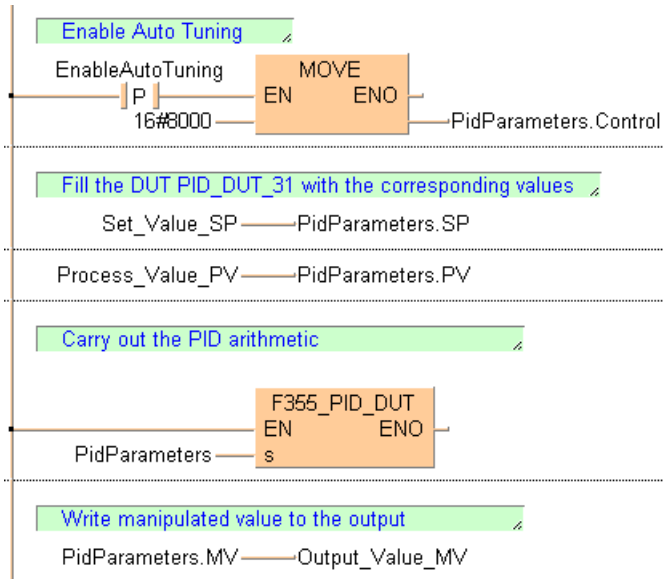
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR_EXTERNAL	EnableAutoTuning	BOOL	FALSE	Switch Auto Tuning on
1	VAR_EXTERNAL	Set_Value_SP	INT	0	A/D CH0
2	VAR_EXTERNAL	Process_Value_PV	INT	0	A/D CH1
3	VAR_EXTERNAL	Output_Value_MV	INT	0	D/A
4	VAR	PidParameters	PID_DUT_31		PID Parameters

Bei der Initialisierung der Variablen PidParameters vom Datentyp PID_DUT_31, wird die Obere Grenze der Stellgröße auf 4000 gesetzt. Der Proportionalbeiwert Kp ist ursprünglich auf 80 (8) gesetzt, Ti und Td auf 200 (20s) und die Regelzykluszeit Ts auf 100 (1s).

Rumpf Die Standardfunktion MOVE kopiert den Wert 16#8000 in das Element Control des SDT PidParameters, wenn die Variable EnableAutoTuning von FALSE auf TRUE schaltet (d.h. aktiviert die Betriebsart Auto-Tuning der Funktion F355_PID_DUT).
Die Variablen Set_Value_SP und Process_Value_PV werden den Elementen SP und PV des SDT PidParameters zugewiesen. Sie erhalten ihre Werte durch die A/D-Wandler Kanäle 0 und 1.
Da der EN-Ausgang des Funktionsbausteins F355_PID_DUT direkt mit der Stromlinie verbunden ist, wird der PID-Regler immer ausgeführt, wenn sich die Steuerung im RUN-Modus befindet. Die berechnete Stellgröße wird im Element MV des SDTs PidParameters gespeichert und der Variablen Output_Value_MV zugeordnet. Ihr Wert wird durch einen D/A-Wandler am Ausgang der Steuerung zurückgegeben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

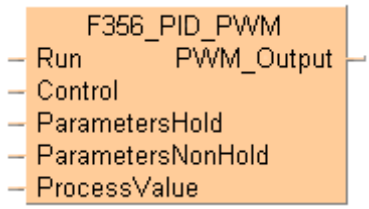
```
(* Auto Tuning: *)
if DF(EnableAutoTuning) then
      PidParameters.Control:=16#8000;
end_if;

(* Fill the DUT PidParameters with the corresponding input values: *)
PidParameters.SP:=Set_Value_SP;
PidParameters.PV:=Process_Value_PV;

(* Carry out the PID arithmetic: *)
F355_PID_DUT(PidParameters);

(* Write the manipulated value to the output: *)
Output_Value_MV:=PidParameters.MV;
```

Allgemeine Programmierinformation Die PID-Regelung dient dazu, einen Istwert (PV = process value) dem festgelegten Sollwert (SP = set point) anzunähern und zu halten. Im Gegensatz zu F355_PID_DUT (s. S. 979) aktiviert dieser Befehl einen PWM-Ausgang. Hierbei handelt es sich um einen pulsweitenmodulierten schaltenden Ausgang. Die PID-Regelungsparameter Kp, Ti und Td können automatisch mit der Auto-Tuning-Funktion ermittelt werden.

F356_PID_PWM

PID-Regler

Erklärung Diese Implementierung erlaubt eine direkte Parametrierung des F355_PID über Argumente:



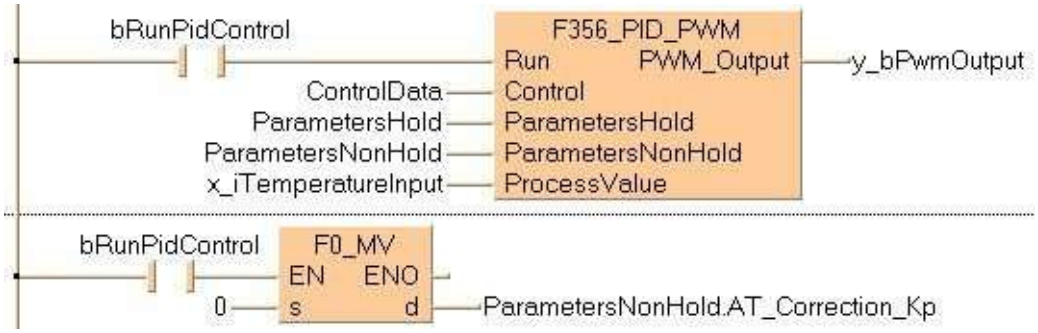
Zur Beschreibung des PID-Regelungsprozesses verwendete Abkürzungen

Abkürzung	Bedeutung	Deutsche Bezeichnungen
PV	Istwert	Istwert, Messwert
SP	Sollwert	Sollwert
MV	Stellgröße	Ausgangswert, Stellgröße MV
Ts	Regelzykluszeit (Sampling time)	Zykluszeit
Ti	Integralzeit	-
Td	Differenzialzeit	-
Kp	Proportionalbeiwert	-
AT	Auto-Tuning	-

Allgemeine Programmierinformation

1. Wurde die Ausführungsbedingung am Eingang **Run** gesetzt, werden die Daten im Argument **ParametersNonHold** initialisiert.
Wenn ein Wert im SDT kein Standardwert sein soll, schreiben Sie die Werte (z.B. mit dem MOVE-Befehl) in den SDT, der kontinuierlich durch die Bedingung TRUE ausgelöst wird.
2. F356_PID_PWM darf nur einmal pro SPS-Zyklus ausgeführt werden. Verwenden Sie den Befehl F356_PID_PWM daher nicht in Interrupt-Programmen oder Schleifen.
3. Die Ausführungsbedingung am Eingang Run muss immer gesetzt sein, da sonst die PID-Regelung abgebrochen wird.
4. Wenn Sie z.B. zur Regelung mehrerer Regelkreise ein paralleles Takten des PWM-Ausgangs verhindern möchten, setzen Sie z.B. Zeitgeber ein, um den Regelungsbeginn entsprechend zu verzögern.

Beispiel:



SPS-Typen **Verfügbarkeit von F356_PID_PWM (s. S. 1191)**

☞ **Die Frequenzperiode des PWM-Ausgangs entspricht der Regelzykluszeit Ts (Frequenz des PWM-Ausgangs = 1/Ts). Das Puls-Pausenverhältnis entspricht der Stellgröße MV x 0,01%, z.B.: MV = 10000 entspricht einem Puls-Pausenverhältnis von 100%.**

Datentypen

Variable	Datentyp	Funktion
Run	BOOL	Startbedingung
Control	F356_Control_DUT	Betriebsart
Parameters Hold	F356_Parameters_Hold_DUT	PID-Parameter
Parameters NonHold	F356_Parameters_NonHold_DUT	Stellgröße (MV), zusätzliche Regelungsparameter, Auto-Tuning- und Arbeitsbereich
ProcessValue	INT	Istwert (-30000–30000)
PWM_Output (see note)	BOOL	Pulsweitenmoduliertes Ausgangssignal (optional anstelle von Stellgröße MV)

Operanden

Für	Merker				T/C		Register			Konstante
Control	-	WY	WR	WL	SV	EV	DT	LD	FL	-
Parameters	WX	WY	WR	WL	SV	EV	DT	LD	FL	-
Process Values	-	WY	WR	WL	SV	EV	DT	LD	FL	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ ein Parameter von F356_Parameters_NonHold_DUT ungültig ist
R9008	%MX0.900.8	kurzzeitig	
R900B	%MX0.900.11	permanent	▪ der mit **UpperLimit** oder **LowerLimit** definierte Bereich ungültig ist

Weitere Informationen:

- Steuerungseinstellungen: F356_Parameters_Hold_DUT
- Sollwert SP und Regelungsparameter: F356_Parameters_NonHold_DUT

■ **Hinweise zum Auto-Tuning:**

- Die Elemente **AT_Progress** in F356_Parameters_NonHold_DUT und **b1_AT_Complete** in F356_Control_DUT werden an der steigenden Flanke des Auto-Tuning-Signals gelöscht.
- Wenn das Auto-Tuning erfolgreich beendet ist, wird das Element **b1_AT_Complete** von F356_Control_DUT gesetzt und der Status im Element **AT_Progress** von F356_Parameters_NonHold_DUT gespeichert.
- Wird das Auto-Tuning abgebrochen, bleiben die Parameter Kp, Ti und Td unverändert.

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste werden alle globalen Ein- und Ausgangswerte deklariert, die für das Programmieren der Funktion verwendet werden.

Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ
0	VAR_GLOBAL	x_iTemperatureInput	WX2	%IW2	INT
1	VAR_GLOBAL	y_bPwmOutput	WY2	%QW2	WORD

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR_EXTERNAL	x_iTemperatureInput	INT	0
1	VAR_EXTERNAL	y_bPwmOutput	BOOL	FALSE
2	VAR	bStartAutoTuning	BOOL	FALSE
3	VAR	bRunPidControl	BOOL	FALSE
4	VAR	ControlData	F356_Control_DUT	
5	VAR	ParametersHold	F356_Parameters_Hold_DUT	
6	VAR	ParametersNonHold	F356_Parameters_NonHold_DUT	

Rumpf Legen Sie zu Beginn das Element **SP** (Sollwert) in F356_Parameters_Hold_DUT fest.

Wenn **bRunPidControl** aktiviert wird, wird der Arbeitsbereich initialisiert, der in F356_Parameters_NonHold_DUT festgelegt wurde. Wurde **b2_HoldMV** in F356_Control_DUT gesetzt, wird die Stellgröße **MV** gehalten.

Standardeinstellungen der Regelungsparameter:

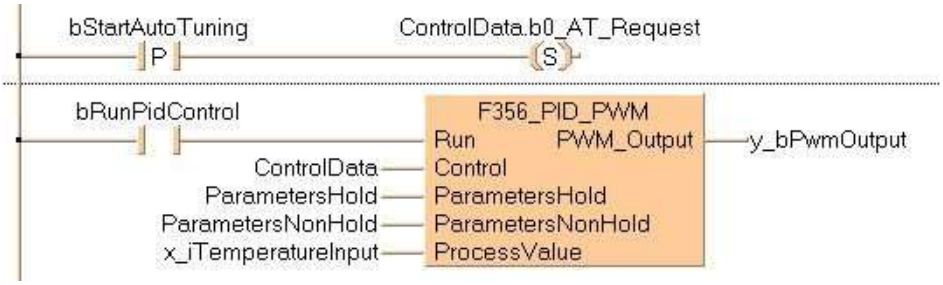
- Frequenzperiode = 1 s
- PD-Regelung, umgekehrte Wirkrichtung (Heizen)
- PWM-Auflösung = 1000

Die PID-Regelung beginnt mit dem nächsten Zyklus und das pulsweitenmodulierte Signal wird an **PWM_Output** ausgegeben.

Das Auto-Tuning beginnt, wenn **b0_AT_Request** von F356_Control_DUT, ein strukturierter Datentyp mit überlappenden Elementen, gesetzt ist. Wenn das Auto-Tuning erfolgreich abgeschlossen ist, wird das Element **b1_AT_Complete** von **F356_Control_DUT** gesetzt und Kp, Ti und Td werden für die PID-Regelung eingestellt. Wenn **bRunPidControl** noch gesetzt ist, beginnt automatisch die PID-Regelung und die PWM-Pulsausgabe erfolgt.

☞ **Wurde die Ausführungsbedingung bRunPidControl während der PID-Regelung zurückgesetzt, endet die Pulsausgabe an PWM_Output. Die Stellgröße MV kann nur gehalten werden, wenn Bit 2 F356_Control_DUT mit b2_HoldMV gesetzt wurde.**

LD



ST

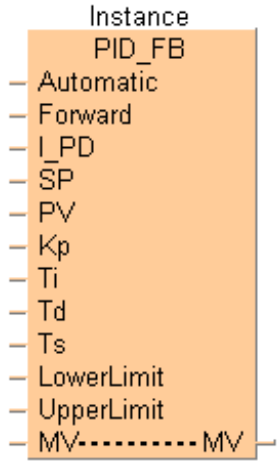
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
(* Auto Tuning: *)
if DF(bStartAutoTuning) then
        ControlData.b0_AT_Request:=TRUE;
end_if;
y_bPwmOutput:=F356_PID_PWM(     Run  := bRunPidControl,
        Control  := ControlData,
        ParametersHold  := ParametersHold,
        ParametersNonHold  := ParametersNonHold,
        ProcessValue  := x_iTemperatureInput);
```

PID_FB

PID-Regler mit Auto-Tuning

Erklärung Diese Implementierung erlaubt eine direkte Parametrierung des F355_PID über Argumente:



Datentypen

Eingangsvariablen (VAR_INPUT):		
Variable	**Datentyp**	**Funktion**
Automatic	BOOL	FALSE: Ermöglicht manuelle Einstellung der Stellgröße TRUE: Stellgröße wird automatisch vorgegeben
Forward		FALSE: Gegensinnige Regelung (Heizen) TRUE: Gleichsinnige Regelung (Kühlen)
I_PD		FALSE: PI-D-Regelung TRUE: I-PD-Regelung
SP	INT	Sollwert, Bereich 0-10000
PV		Istwert, Bereich 0-10000
Kp		Proportionalbeiwert, Bereich: 1-9999, Einheit: 0.1
Ti		Integralzeit, Bereich: 1-30000, Einheit: 0,1s
Td		Differenzialzeit, Bereich: 1-10000, Einheit: 0,1s
Ts		Regelzykluszeit (Sampling time), Bereich: 1-6000, Einheit: 0,01s
LowerLimit		Untere Grenze der Stellgröße, Bereich: 0-10000
UpperLimit		Obere Grenze der Stellgröße, Bereich: 1-10000
Ein-/Ausgangsvariablen (VAR_IN_OUT)		
MV	Stellgröße	

☞

- **Das Auto-Tuning ist mit PID_FB nicht möglich. Verwenden Sie in diesem Fall PID_FB_DUT (s. S. 989).**
- **Der Wert von MV kann von extern entweder gleich beim Programmstart bei der Initialisierung vorgegeben werden oder wenn der Wert von Automatic FALSE ist.**
- **Um maximale Auflösung und minimale Totzeiten jenseits von LowerLimit und UpperLimit zu erzielen, sollten diese Werte möglichst den gesamten Bereich von 0–10000 abdecken.**

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

GVL In der globalen Variablenliste werden alle globalen Ein- und Ausgangswerte deklariert, die für das Programmieren der Funktion verwendet werden. Die Adressen sind vom SPS-Typ abhängig.

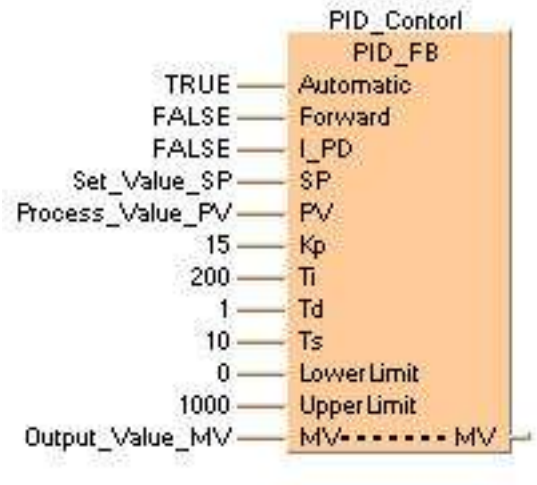
Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Kommentar
0	VAR_GLOBAL	Set_Value_SP	WX4	%IW4	INT	A/D CH0
1	VAR_GLOBAL	Process_Value_PV	WX5	%IW5	INT	A/D CH1
2	VAR_GLOBAL	Output_Value_MV	WY4	%QW4	INT	D/A

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR_EXTERNAL	Set_Value_SP	INT	0
1	VAR_EXTERNAL	Process_Value_PV	INT	0
2	VAR_EXTERNAL	Output_Value_MV	INT	0
3	VAR	PID_Control	PID_FB	

KOP



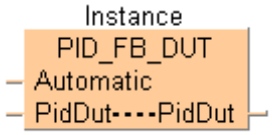
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
PID_Control( Automatic:= TRUE,
        Forward:= FALSE,
        I_PD:= FALSE,
        SP:= Set_Value_SP,
        PV:= Process_Value_PV,
        Kp:= 15,
        Ti:= 200,
        Td:= 1,
        Ts:= 10,
        LowerLimit:= 0,
        UpperLimit:= 1000,
        MV:= Output_Value_MV);
```

PID_FB_DUT

PID-Regler mit Auto-Tuning

Erklärung Diese Implementierung erlaubt den Zugriff auf den Befehl F355_PID über die Struktur PID_DUT.



Diese Struktur, definiert in den Systembibliotheken / "FP Library" / SDTs enthält die folgenden Parameter (nähere Informationen finden Sie im SDT PID_DUT):

Parameter	Datentyp	Funktion		
Ansteuerung	WORD	Steuermodus 16#X000 Gegensinnige PI-D-Regelung 16#X001 Gleichsinnige I-PD-Regelung 16#X002 Gegensinnige I-PD-Regelung 16#X003 Gleichsinnige PI-D-Regelung		
			Bereich	**Modul**
SP	INT	Sollwert	0-10000	
PV		Istwert	0-10000	
MV		Stellgröße	0-10000	
LowerLimit		Untere Grenze der Stellgröße	0-10000	
UpperLimit		Obere Grenze der Stellgröße	1-10000	
Kp		Proportionalbeiwert	1-9999	0.1
Ti		Integralzeit	1-30000	0,1s
Td		Differenzialzeit	1-10000	0,1s
Ts		Regelzykluszeit (Sampling time)	1-6000	0,01s
AT_Progress		Auto-Tuning-Fortschritt	0-5	

Datentypen

Eingangsvariablen (VAR_INPUT):		
Variable	**Datentyp**	**Funktion**
Automatische	BOOL	FALSE: Ermöglicht manuelle Einstellung der Stellgröße TRUE: Stellgröße wird automatisch vorgegeben
Ein-/Ausgangsvariable (VAR_IN_OUT)		
PidDut	PID_DUT	

☞

- **Der SDT PID_DUT darf nicht noch mal unter "SDTs" des aktuellen Projekts angelegt werden.**
- **Der Wert von MV kann von extern entweder gleich beim Programmstart bei der Initialisierung vorgegeben werden oder wenn der Wert von Automatic FALSE ist.**
- **Um maximale Auflösung und minimale Totzeiten jenseits von LowerLimit und UpperLimit zu erzielen, sollten diese Werte möglichst den gesamten Bereich von 0–10000 abdecken.**

Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

GVL In der globalen Variablenliste werden alle globalen Ein- und Ausgangswerte deklariert, die für das Programmieren der Funktion verwendet werden. Die Adressen sind vom SPS-Typ abhängig.

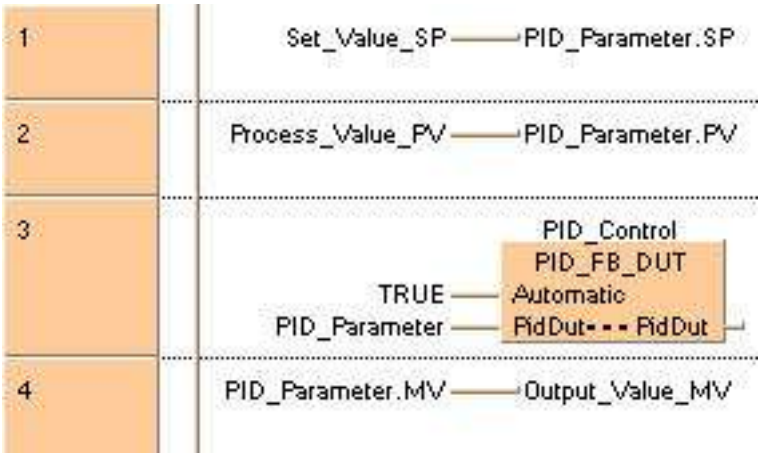
Glob. Variablen

	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Kommentar
0	VAR_GLOBAL	Set_Value_SP	WX4	%IW4	INT	A/D CH0
1	VAR_GLOBAL	Process_Value_PV	WX5	%IW5	INT	A/D CH1
2	VAR_GLOBAL	Output_Value_MV	WY4	%QW4	INT	D/A

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR_EXTERNAL	Set_Value_SP	INT	0
1	VAR_EXTERNAL	Process_Value_PV	INT	0
2	VAR_EXTERNAL	Output_Value_MV	INT	0
3	VAR	PID_Parameter	PID_DUT	UpperLimit := 1000,Kp := 15,Ti := 200,Td := 1,Ts := 10
4	VAR	PID_Control	PID_FB_DUT	

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
PID_Parameter.SP  := Set_Value_SP;
PID_Parameter.PV  := Process_Value_PV;
PID_Control( Automatic:= TRUE,
        PidDut:= PID_Parameter);
Output_Value_MV  := PID_Parameter.MV;
```

Kapitel 31

FP-e, Befehle zur Konfiguration der Anzeige

F180_SCR

Bildschirmanzeigebefehl

Erklärung Mit diesem Befehl wird die Bildschirmanzeige der FP-e für den N-Modus (Normalmodus) und den S-Modus (S = switch, Tastermodus) konfiguriert.

SPS-Typen **Verfügbarkeit von F180_SCR (s. S. 1207)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Gültige Werte für Anzeigemodus s1
s2	ARRAY [0..2] OF ANY16	Steuercode für Bildschirmmaske (3 Worte).
s3	ANY16	Daten im oberen Bereich
s4		Daten im unteren Bereich

Operanden

Für	Merker				T/C		Register			Konstante
s1	WX	WY	WR	-	SV	EV	DT	IX	IY	dez., hex.
s2	WX	WY	WR	-	SV	EV	DT	IX	IY	
s3	WX	WY	WR	-	SV	EV	DT	-	-	
s4	-	WY	WR	-	SV	EV	DT	IX	IY	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ mit Indexmodifizierern definierter Bereich ist größer als zulässiger Bereich
R9008	%MX0.900.8	kurzzeitig	▪ für s1 oder s2 ein ungültiger Wert eingegeben wurde

☞
- **Sonderdatenregister "DT9***" kann für den unteren Anzeigebereich nicht angegeben werden.**
- **Dieser Befehl kann nicht in einem Interrupt-Programm verwendet werden.**

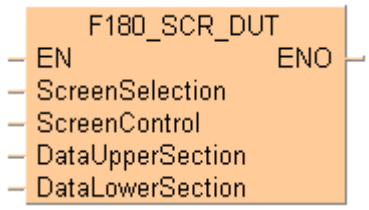
Weitere Informationen finden Sie in der Online-Hilfe:

- Beispiel für den Steuercode
- Anzeige von ASCII-Code
- Anzeige von 7-Segment-Daten

F180_SCR_DUT

Konfiguration der Anzeige der FP-e

Erklärung Dieser Befehl ermöglicht auf einfache Weise die Bildschirmanzeige der FP-e für den N-Modus (Normalmodus) und den S-Modus (S = switch, Tastermodus) zu konfigurieren.



Hierfür wird in einem übersichtlichen Dialog der Steuercode für die Bildschirmanzeige konfiguriert.

Vorgehensweise

1. **Anlegen eines SDTs**
2. **Wählen Sie F180_DUT im Kopf der Deklaration unter "Typ"**
3. **Klicken Sie auf**  **im Feld "Initial"**

Der Konfigurationsdialog öffnet sich.

4. **Nehmen Sie die gewünschten Einstellungen vor**
5. **[OK]**

SPS-Typen **Verfügbarkeit von F180_SCR_DUT (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
ScreenSelection	ANY16	Anzeigemodus
ScreenControl	F180_DUT	Strukturierter Datentyp für den Steuercode der Bildschirmmaske
DataUpperSection	ANY16	Wert im oberen Anzeigebereich
DataLowerSection		Wert im unteren Anzeigebereich

Operanden

Für	Relais				T/C		Register			Konstante
Screen Selection	WX	WY	WR	-	SV	EV	DT	IX	IY	dez., hex.
Screen Control	-	-	-	-	-	-	-	-	-	-
DataUpper Section	WX	WY	WR	-	SV	EV	DT	IX	IY	-
DataLower Section	-	WY	WR	-	SV	EV	DT	IX	IY	-

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ - der mit Indexmodifikatoren definierte Bereich größer ist als der zulässige Bereich ▪ - für **s1** oder **s2** ein ungültiger Wert eingegeben wurde
R9008	%MX0.900.8	kurzzeitig	

☞
- **Sonderdatenregister "DT9***" kann für den unteren Anzeigebereich nicht angegeben werden.**
- **Dieser Befehl kann nicht in einem Interrupt-Programm verwendet werden.**

Beispiel In diesem Beispiel wird die Funktion F180_SCR im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projektes verwendet werden können.

Glob. Variablen

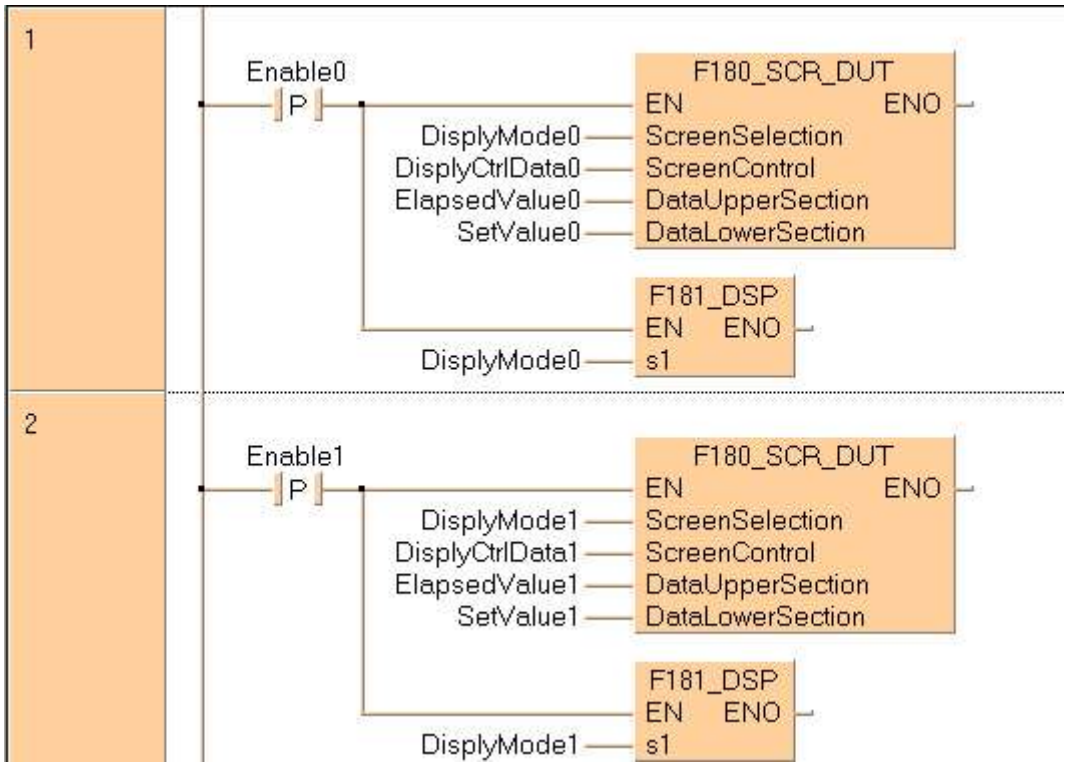
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	ElapsedValue0	EV0	%MW4.0	INT	88
1	VAR_GLOBAL	ElapsedValue1	EV1	%MW4.1	INT	88
2	VAR_GLOBAL	SetValue0	SV0	%MW3.0	INT	100
3	VAR_GLOBAL	SetValue1	SV1	%MW3.1	INT	200

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR_EXTERNAL	ElapsedValue0	INT	88
1	VAR_EXTERNAL	ElapsedValue1	INT	88
2	VAR_EXTERNAL	SetValue0	INT	100
3	VAR_EXTERNAL	SetValue1	INT	200
4	VAR	DisplayCtrlData0	F180_DUT	ScreenControl := 16#83
5	VAR	DisplayCtrlData1	F180_DUT	ScreenControl := 16#83
6	VAR	DisplayMode0	INT	0
7	VAR	DisplayMode1	INT	1
8	VAR	Enable0	BOOL	FALSE
9	VAR	Enable1	BOOL	FALSE

Rumpf Wenn die Variable **Enable0** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt und die FP-e in N-Modus 1 geschaltet. **ProcessValue0** und **SetValue0** werden in der oberen (rot) und unteren Anzeige (orange) dargestellt. Wenn die Variable **Enable1** von FALSE auf TRUE gesetzt wird, wird die Funktion ausgeführt und die FP-e wechselt in N-Modus 2. **ProcessValue1** und **SetValue1** werden in der oberen (rot) und unteren Anzeige (grün) dargestellt. Das Symbol "Werte monitoren" wurde für beide KOP-Rümpfe aktiviert. Mit dem Befehl F181_DSP (s. S. 998) ändern Sie die Anzeige der FP-e.

KOP



DisplayMode0	**DisplayMode1**
DisplayControlData0 ScreenControl := 16#83, UpperDisplayControl := 16#4000, LowerDisplayControl := 16#2000	DisplayControlData1 ScreenControl := 16#83, UpperDisplayControl := 16#4000, LowerDisplayControl := 16#6000
	

Siehe auch in der Online-Hilfe: Kurzbeschreibung der Bedienelemente

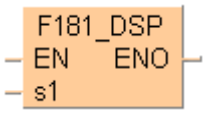
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF (Enable0) THEN
    F180_SCR_DUT(ScreenSelection:=DisplayMode0,
                 ScreenControl:=DisplayCtrlData0,
                 DataUpperSection:=ElapsedValue0,
                 DataLowerSection:=SetValue0);
  F181_DSP (DisplayMode0);
END_IF;

IF DF (Enable1) THEN
    F180_SCR_DUT(ScreenSelection:=DisplayMode1,
                 ScreenControl:=DisplayCtrlData1,
                 DataUpperSection:=ElapsedValue1,
                 DataLowerSection:=SetValue1);
  F181_DSP (DisplayMode1);
END_IF;
```

F181_DSP

Änderung Bildschirmanzeigemodus

Erklärung Mit **s1** wird der Bildschirmanzeigemodus der FP-e eingestellt.



SPS-Typen **Verfügbarkeit von F181_DSP (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Anzeigemodus (0 bis 7)

Operanden

Für	Merker				T/C		Register			Konstante
s1	WX	WY	WR	-	SV	EV	DT	IX	IY	dez., hex.

Gültige Werte für Anzeigemodus s1

Werte für s1	Anzeigemodus
0	N-Modus 1
1	N-Modus 2
2	S-Modus 1
3	S-Modus 2
4	R-Modus 1
5	R-Modus 2
6	I-Modus 1
7	I-Modus 2

(N = Normalmodus, S = Tastermodus [S = switch], R = Registermodus, I = E/A-Monitormodus [I = I/O])

Fehlermerker:

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ mit Indexmodifizierern definierter Bereich ist größer als zulässiger Bereich
R9008	%MX0.900.8	kurzzeitig	▪ für **s1** ein ungültiger Wert eingegeben wurde

☞

- **Wenn für s1 ein anderer Wert als 0-7 angegeben wird, tritt ein Operationsfehler auf.**
- **Dieser Befehl kann nicht in einem Interrupt-Programm verwendet werden.**

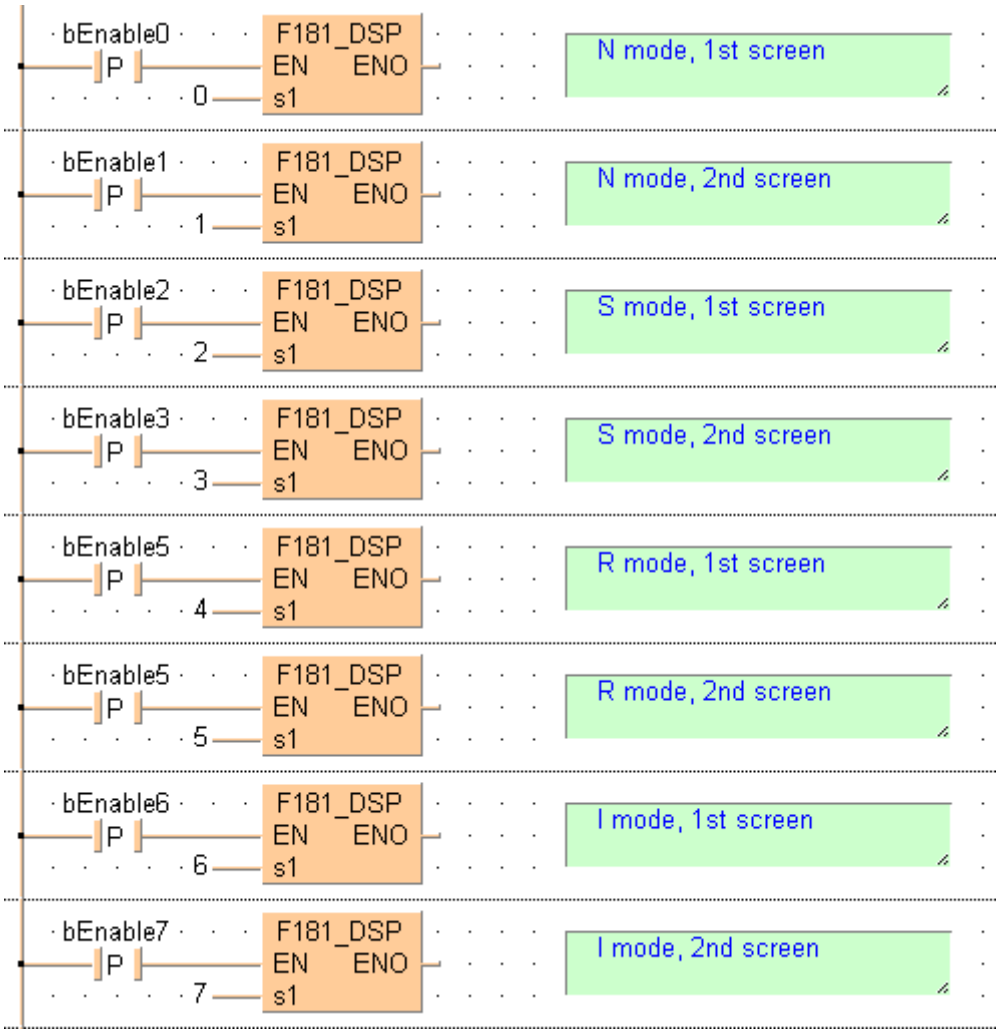
Beispiel In diesem Beispiel wird die Funktion F181_DSP im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bEnable0	BOOL	FALSE
1	VAR	bEnable1	BOOL	FALSE
2	VAR	bEnable2	BOOL	FALSE
3	VAR	bEnable3	BOOL	FALSE
4	VAR	bEnable4	BOOL	FALSE
5	VAR	bEnable5	BOOL	FALSE
6	VAR	bEnable6	BOOL	FALSE
7	VAR	bEnable7	BOOL	FALSE

Rumpf Wenn die Variablen **Enable0** bis **Enable7** auf TRUE gesetzt werden, wird die Funktion ausgeführt und die FP-e wird in den entsprechenden Anzeigemodus geschaltet. (N = Normalmodus, S = Tastermodus [S = switch], R = Registermodus, I = E/A-Monitormodus [I = I/O])

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF DF(bEnable0) THEN
    (* N mode, 1st screen *)
    F181_DSP(s1:=0);
END_IF;
IF DF(bEnable1) THEN
    (* N mode, 2nd screen *)
    F181_DSP(s1:=1);
END_IF;
IF DF(bEnable2) THEN
    (* S mode, 1st screen *)
    F181_DSP(s1:=2);
END_IF;
IF DF(bEnable3) THEN
    (* S mode, 2nd screen *)
    F181_DSP(s1:=3);
END_IF;
IF DF(bEnable4) THEN
    (* R mode, 1st screen *)
    F181_DSP(s1:=4);
END_IF;
IF DF(bEnable5) THEN
    (* R mode, 2nd screen *)
    F181_DSP(s1:=5);
END_IF;
IF DF(bEnable6) THEN
    (* I mode, 1st screen *)
    F181_DSP(s1:=6);
END_IF;
IF DF(bEnable7) THEN
    (* I mode, 2nd screen *)
    F181_DSP(s1:=7);
END_IF;
```

Kapitel 32

Systemregisterbefehle

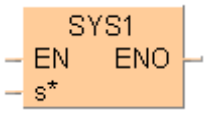
SYS1 SPS-Einstellungen ändern

Erklärung Die Beschreibung für den Befehl SYS1 ist in folgende Abschnitte unterteilt:

- Kommunikationsparameter einstellen (s. S. 1004)
- Passwort einstellen (s. S. 1008)
- Interrupt einstellen (s. S. 1009)
- SPS-Kopplung: Zeiten einstellen (s. S. 1011)
- Schnelle-Zähler-Funktion (s. S. 1013)
- RS485: Ansprechzeit einstellen (s. S. 1014)

SPS-Typen **Verfügbarkeit von SYS1 (s. S. 1197)**

Kommunikationsparameter für die COM-Schnittstellen der CPU einstellen



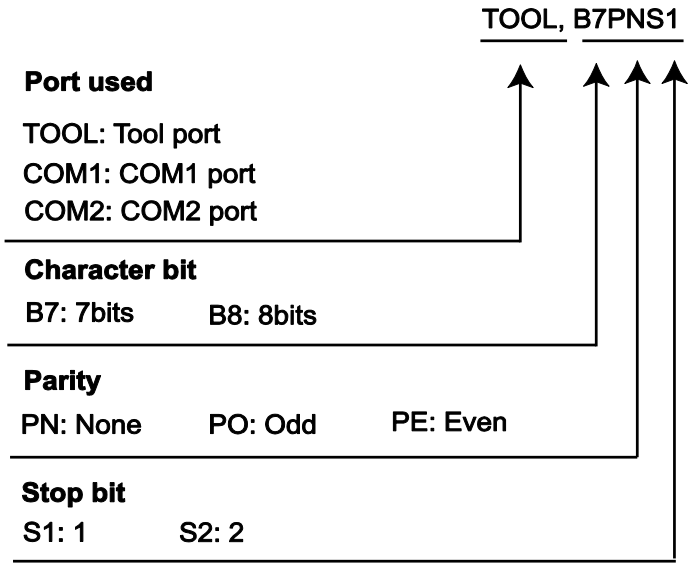
Die Kommunikationseinstellungen für die COM- bzw. TOOL-Schnittstelle werden geändert, indem Sie eine Zeichenkette als Konstante an den Eingang s* anlegen.

Das 1. Schlüsselwort der Zeichenkette legt fest, für welche Schnittstelle die Änderung gilt. Das 2. Schlüsselwort bestimmt die neuen Werte. Trennen Sie die Schlüsselwörter durch ein Komma.

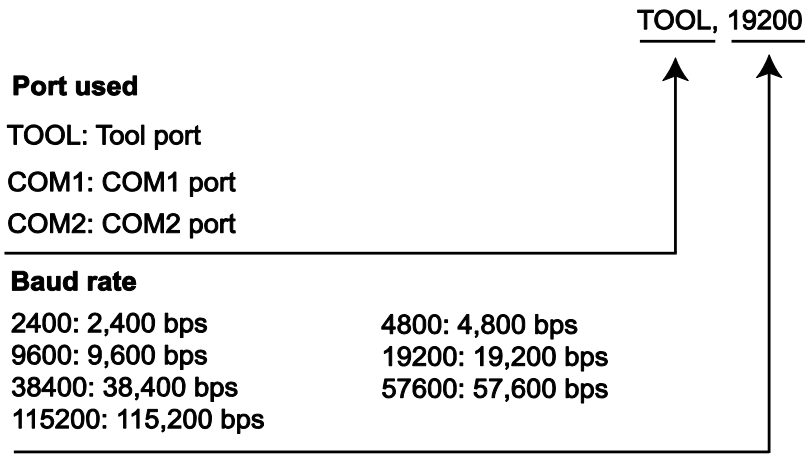
Die folgenden Einstellungen können geändert werden:

1. Kommunikationsformat
2. Baudrate
3. Teilnehmeradresse
4. Start- und Endezeichen
5. RTS-Steuerung (Request to Send = Sendeaufforderung)

Schlüssel-wörter 1. Kommunikationsformat (gilt für TOOL-, COM1- und COM2-Schnittstelle)

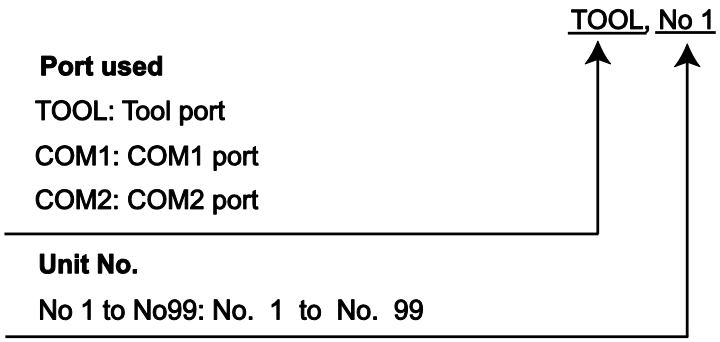


2. Baudrate (gilt für TOOL-, COM1- und COM2-Schnittstelle)



Für die FP-X V2.0 oder höhere Versionen und für die FPΣ V3.1 oder höhere Versionen lassen sich niedrigere Baudraten von 300, 600 und 1200bit/s angeben. Diese Baudraten können nicht in den Systemregistern gesetzt werden.

3. Teilnehmeradresse (gilt für TOOL-, COM1- und COM2-Schnittstelle)



Mit der FP0R verwenden Sie die Schlüsselwörter 'COM1No' und 'TOOLNo', um die Teilnehmeradresse von einem Datenregister (DT0–DT9999) zu lesen, das die Teilnehmeradressen 1–99 enthält. Das Datenregister muss mit genau fünft Zeichen angegeben werden: Zum Beispiel: D0815 gibt DT815 an. Die führenden Nullen müssen eingegeben werden. Das Schlüsselwort unterscheidet zwischen Groß- und Kleinschreibung, d.h. COM1NO, Com1No oder … d0815 wären ungültig.

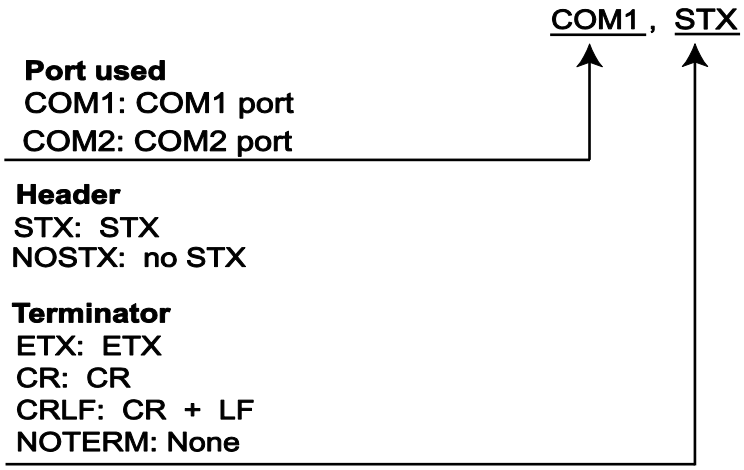
Beispiel:

SYS1 'COM1No,D9999' gibt DT9999 an

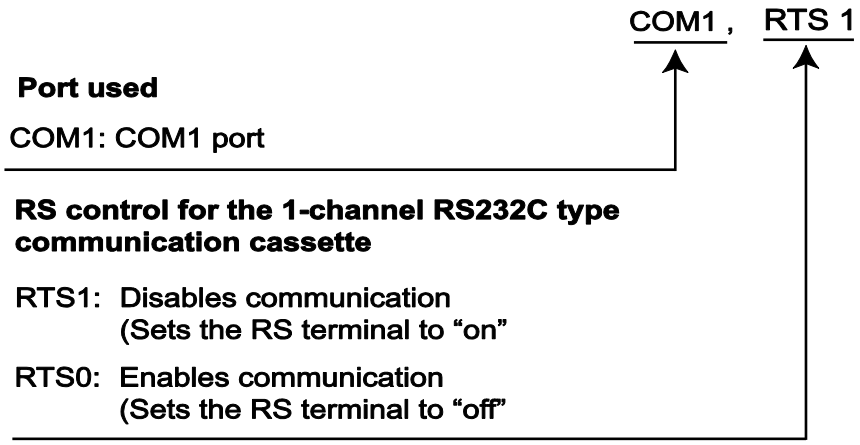
SYS1 'COM1No,D0000' gibt DT0 an

Es tritt ein Berechnungsfehler auf, wenn dem DT-Bereich andere Werte als 1–99 zugewiesen werden.

4. Start- und Endezeichen (gilt für TOOL-, COM1- und COM2-Schnittstelle)



5. RTS-Steuerung (gilt für COM 1-Schnittstelle)



Vorsichts-maßnahmen bei der Program-mierung

- Wenn dieser Befehl ausgeführt wird, wird der Inhalt des System-ROMs in der CPU nicht überschrieben. Das heißt, der Inhalt der Systemregister richtet sich beim Aus- und Wiedereinschalten nach den Software-Einstellungen.
- Es wird empfohlen, diesen Befehl nur bei steigender Flanke auszuführen.

- Da Systemregister-Einstellungen geändert werden, kann es sein, dass ein Fehler ausgegeben wird, wenn die Systemregister mit FPWIN Pro verglichen werden.
- Trennen Sie die Schlüsselwörter durch ein Komma und verwenden Sie keine Leerzeichen.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ etwas anderes als ein Schlüsselwort angegeben wird ▪ kein Komma zwischen dem 1. und 2. Schlüsselwort steht ▪ das Schlüsselwort in Kleinbuchstaben geschrieben ist ▪ kein Schnittstellenmodul installiert ist und COM1 oder COM2 gesetzt ist ▪ COM1 oder COM2 gesetzt ist und die Teilnehmeradresse geändert wird, der Teilnehmeradressschalter jedoch nicht auf 0 steht ▪ eine Teilnehmeradresse kleiner 1 oder größer 99 gesetzt wird ▪ SPS-Kopplung für COM1 festgelegt ist und die Baudrate oder das Übertragungsformat für COM1 geändert wurde
R9008	%MX0.900.8	kurzzeitig	▪ die Baudrate oder das Übertragungsformat geändert wird, während die TOOL-Schnittstelle, der COM1-Schnittstelle oder der COM2-Schnittstelle von einem Modem initialisiert wird ▪ "Andere Geräte" nicht spezifiziert wird, wenn der Start- und Endecode gesetzt sind ▪ RTS-Steuerung angegeben ist und ein anderes Schnittschnellenmodul als Typ 1xRS232C installiert ist ▪ für die COM1-Schnittstelle der Modus SPS-Kopplung gewählt wurde und die angegebene Teilnehmeradresse höher ist als die höchste im Systemregister festgelegte Adresse

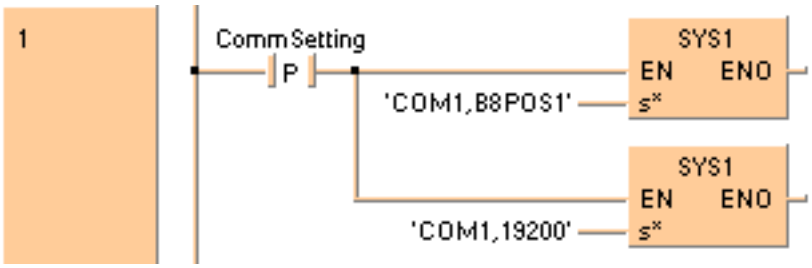
Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	CommSettings	BOOL	FALSE	sets transmission format to:
1	VAR				Character bit 8, Parity: Odd, Stop bit: 1, Baud rate: 19,200bps.

Rumpf Wenn **CommSettings** auf TRUE gesetzt wird, werden Übertragungsformat und Baudrate für COM1 wie folgt gesetzt: Datenlänge: 8, Parität: Ungerade; Stoppbit: 1; Baudrate: 19.200 bit/s

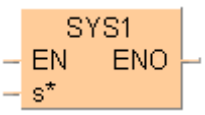
KOP



☞ **Die Zeichenkette, die an s* anliegt, darf am Ende keine Leerzeichen enthalten. In FPWIN Pro ab Version 4.0 werden diese Leerzeichen automatisch vom Compiler gelöscht.**

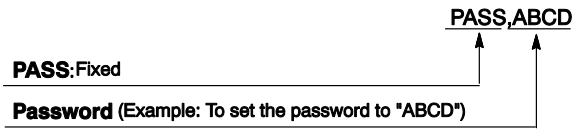
Passwort einstellen Dieser Befehl ändert das Passwort, das von der SPS vorgegeben wurde.

Symbol:



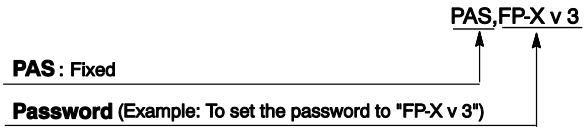
Das 1. Schlüsselwort ist fest, das 2. Schlüsselwort ändert das Passwort. Trennen Sie die Schlüsselwörter durch ein Komma.

Schlüsselworteinstellung (4 Zeichen hexadezimal)



Schlüsselworteinstellung (8 Zeichen alphanumerisch)

Geben Sie z.B. 'PAS,FP-X␣v␣3' ein. Leerzeichen am Ende des Passworts werden nicht berücksichtigt.



Vorsichtsmaßnahmen bei der Prog.

- Wenn dieser Befehl ausgeführt wird, wird der interne FROM-Speicher neu beschrieben. Dies dauert etwa 100ms.
- Ist das neue Passwort identisch mit dem alten Passwort, wird der FROM-Speicher nicht neu beschrieben.
- Es wird empfohlen, diesen Befehl nur bei steigender Flanke auszuführen.
- Trennen Sie die Schlüsselwörter durch ein Komma und fügen Sie keine Leerzeichen hinzu.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ etwas anderes als ein Schlüsselwort angegeben wird ▪ kein Komma zwischen dem 1. und 2. Schlüsselwort steht ▪ das Schlüsselwort in Kleinbuchstaben geschrieben ist
R9008	%MX0.900.8	kurzzeitig	▪ das Passwort nicht aus genau 4 Zeichen oder aus anderen als den Zeichen 0 bis 9 und A bis F besteht.

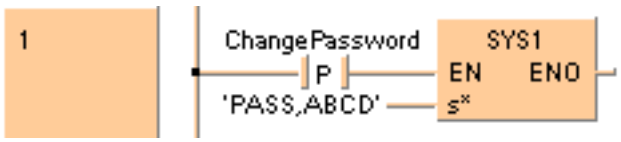
Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	ChangePassword	BOOL	FALSE

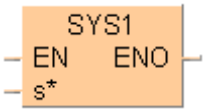
Rumpf Wenn **ChangePassword** freigegeben wird, wird das Passwort der SPS zu "ABCD" geändert.

KOP



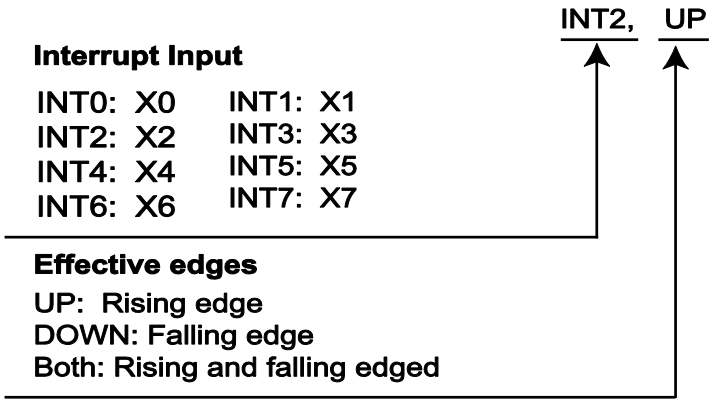
☞ **Die Zeichenkette, die an s* anliegt, darf am Ende keine Leerzeichen enthalten. In FPWIN Pro ab Version 4.0 werden diese Leerzeichen automatisch vom Compiler gelöscht.**

Interrupt einstellen Dieser Befehl stellt den Interrupt-Eingang ein.



Das 1. Schlüsselwort der Zeichenkette bestimmt den Interrupt-Eingang, das 2. Schlüsselwort die Bedingungen. Trennen Sie die Schlüsselwörter durch ein Komma.

Schlüsselwörter



Bei der FP-X können Sie die Eingänge INT0 bis INT13 festlegen.

Vorsichtsmaßnahmen bei der Programmierung

- Wenn dieser Befehl ausgeführt wird, wird der Inhalt des System-ROMs in der CPU nicht überschrieben. Das heißt, der Inhalt der Systemregister richtet sich beim Aus- und Wiedereinschalten nach den Software-Einstellungen.
- Es wird empfohlen, diesen Befehl nur bei steigender Flanke auszuführen.
- Wenn UP bzw. DOWN spezifiziert wird, werden die Systemregister-Einstellungen geändert. Bei einem Vergleich der Systemregister mit FPWIN Pro kann also ein Fehler ausgegeben werden. Wenn BOTH spezifiziert wird, bleiben die Systemregister unverändert.
- Trennen Sie die Schlüsselwörter durch ein Komma und fügen Sie keine Leerzeichen hinzu.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ etwas anderes als ein Schlüsselwort angegeben wird ▪ kein Komma zwischen dem 1. und 2. Schlüsselwort steht ▪ das Schlüsselwort in Kleinbuchstaben geschrieben ist
R9008	%MX0.900.8	kurzzeitig	

Beispiel

In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

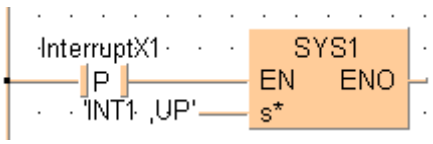
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	InterruptX1	BOOL	FALSE

Rumpf

Wenn **InterruptX1** freigegeben wird, dann ändert sich die Bedingung am Interrupt-Eingang X1 auf "steigende Flanke".

KOP



ST

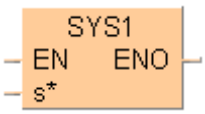
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if (DF(InterruptX1)) then
      SYS1('INT1, UP');
end_if;
```

☞ **Die Zeichenkette, die an s* anliegt, darf am Ende keine Leerzeichen enthalten. In FPWIN Pro ab Version 4.0 werden diese Leerzeichen automatisch vom Compiler gelöscht.**

SPS-Kopplung: Zeiten einstellen

Dieser Befehl legt die Zeiteinstellungen für SPS-Kopplung fest.



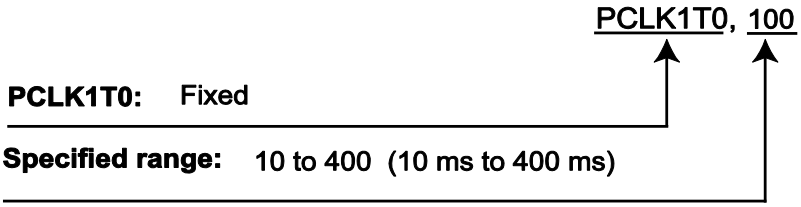
Mit dem 1. Schlüsselwort wird der Parameter angegeben, für den mit dem 2. Schlüsselwort die Zeit eingestellt wird. Trennen Sie die Schlüsselwörter durch ein Komma.

Mit dem SYS1-Befehl können Sie das Abfrageintervall für die Zuschaltprüfung (PCLK1T0) und damit die Übertragungszykluszeit verkürzen. Die Zuschaltprüfung dauert um so länger, je mehr Teilnehmer nicht zugeschaltet sind. (Nicht zugeschaltete Teilnehmer sind Teilnehmer, die nicht zwischen den ersten Teilnehmer und den Teilnehmer mit der höchsten Adresse geschaltet wurden, oder die nicht eingeschaltet wurden.)

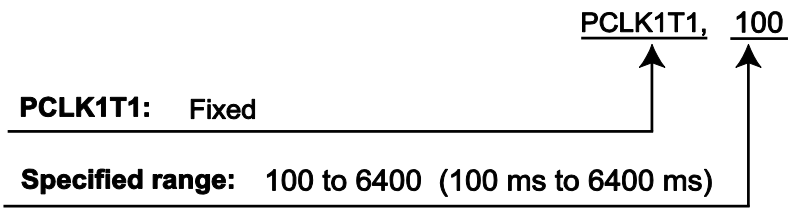
Mit dem SYS1-Befehl können Sie auch die Fehlererkennungszeit bei einem Übertragungsfehler (PCLK1T1) einstellen. Hiermit können Sie die Zeit zwischen dem Abschalten eines Teilnehmers und dem Zurücksetzen des Sondermerkers R9060 bis R906F (Übertragungsfehler) bei einem anderen Teilnehmer verkürzen.

Schlüsselwörter

1. Abfrageintervall für Zuschaltprüfung



2. Fehlererkennungszeit bei Übertragungsfehler



Vorsichtsmaßnahmen bei der Programmierung

- Das Programm sollte zu Beginn auf allen miteinander verbundenen Steuerungen mit den gleichen Einstellungen ausgeführt werden.
- Der Sondermerker R9014 (Initialisierimpuls AUS) sollte den Befehl bei steigender Flanke freigeben (siehe Programmierbeispiel).
- Die Systemregister-Einstellungen werden nicht geändert, wenn dieser Befehl ausgeführt wird.
- Trennen Sie die Schlüsselwörter durch ein Komma und fügen Sie keine Leerzeichen hinzu.

Vorsichtsmaßnahmen beim Einstellen des Abgrageintervalls für die Zuschaltprüfung

- Der Wert sollte mindestens doppelt so groß sein wie die längste Zykluszeit der miteinander verbundenen Steuerungen.
- Wenn ein zu kleiner Wert festgelegt wird, kann es passieren, dass Steuerungen nicht zugeschaltet werden, obwohl sie eingeschaltet sind.
- Wenn es Teilnehmer gibt, die noch nicht zugeschaltet sind, sollten Sie trotzdem die Einstellung nicht ändern, auch wenn die Übertragungszykluszeit dadurch länger wird. (Die Voreinstellung beträgt 400 ms.)

Vorsichtsmaßnahmen beim Einstellen der Fehlererkennungszeit bei einem Übertragungsfehler

- Der Wert sollte mindestens doppelt so groß sein wie die längste Übertragungszykluszeit der miteinander verbundenen Steuerungen.
- Wenn ein zu kleiner Wert festgelegt wird, kann es passieren, dass die Sondermerker R9060 bis R906F nicht richtig funktionieren.
- Sie sollten die Einstellung nicht ändern, auch wenn die Fehlererkennungszeit länger wird. (Die Voreinstellung beträgt 6400 ms.)

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007 R9008	%MX0.900.7 %MX0.900.8	permanent kurzzeitig	▪ etwas anderes als ein Schlüsselwort angegeben wird ▪ kein Komma zwischen dem 1. und 2. Schlüsselwort steht ▪ das Schlüsselwort in Kleinbuchstaben geschrieben ist ▪ der angegebene Wert außerhalb des gültigen Bereichs liegt

Beispiel

Im folgenden Beispiel wird der Befehl im Kontaktplan (KOP) programmiert. Da FP-Adressen und Zeichenketten direkt an die Kontakte anhängt werden, ist kein POE-Kopf nötig.

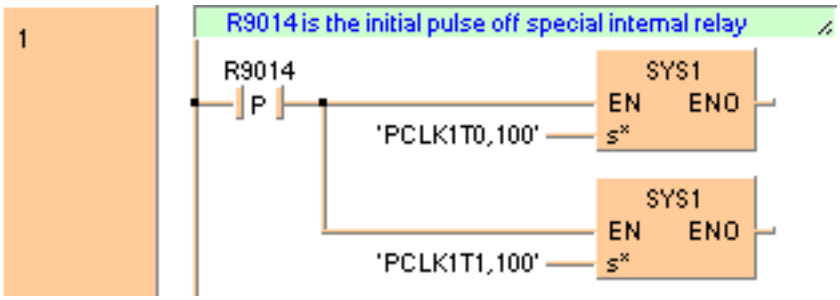
Rumpf

Wenn R9014 bei einer SPS-Kopplung auf TRUE gesetzt wird, werden das Abfrageintervall für die Zuschaltprüfung und die Fehlererkennungszeit bei einem Übertragungsfehler wie folgt gesetzt:

Abfrageintervall: 100 ms

Fehlererkennungszeit bei Übertragungsfehler 100 ms

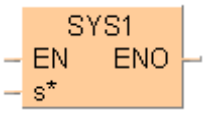
KOP



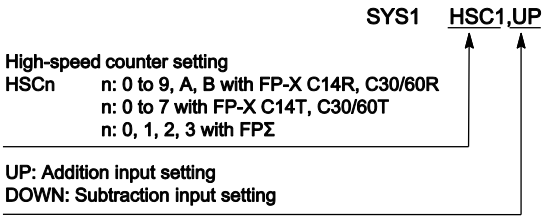
☞ **Die Zeichenkette, die an s* anliegt, darf am Ende keine Leerzeichen enthalten. In FPWIN Pro ab Version 4.0 werden diese Leerzeichen automatisch vom Compiler gelöscht.**

Zählrichtung des schnellen Zählers ändern

Dieser Befehl ändert die Zählrichtung des schnellen Zählers, indem Sie eine Zeichenkette als Konstante an den Eingang s* anlegen.



Schlüsselwörter



Vorsichtsmaßnahmen bei der Programmierung

- Wenn das entsprechende HSC-Systemregister auf "Unbenutzt" gesetzt ist, tritt ein Operationsfehler auf. Setzen Sie das Systemregister vorab auf "Vorwärtszähleingang" oder "Rückwärtszähleingang".
- Wenn dieser Befehl ausgeführt wird, wird der Inhalt des System-ROMs in der CPU nicht überschrieben. Das heißt, der Inhalt der Systemregister richtet sich beim Aus- und Wiedereinschalten nach den Software-Einstellungen.
- Es empfiehlt sich, diesen Befehl nur einmal auszuführen, zum Beispiel bei der steigenden oder fallenden Flanke einer Ausführungsbedingung.
- Wenn UP bzw. DOWN angegeben ist, werden die Systemregister-Einstellungen geändert. Bei einer Überprüfung oder Kompilierung des Programms kann also ein Fehler ausgegeben werden. Wenn BOTH angegeben wird, bleiben die Systemregister unverändert. Trennen Sie die Schlüsselwörter durch ein Komma "," z.B. **HSCB,UP**; fügen Sie keine Leerzeichen hinzu. Ansonsten kann ein Operationsfehler auftreten.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	Permanent	▪ etwas anderes als ein Schlüsselwort angegeben wird ▪ kein Komma zwischen dem ersten und zweiten Schlüsselwort steht
R9008	%MX0.900.8	kurzzeitig	▪ die Buchstaben des Schlüsselworts nicht groß geschrieben sind ▪ im HSC-Systemregister nicht der Vorwärts- oder Rückwärtszähleingang eingestellt ist

Beispiel

In diesem Beispiel wird die Funktion HSC im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert.

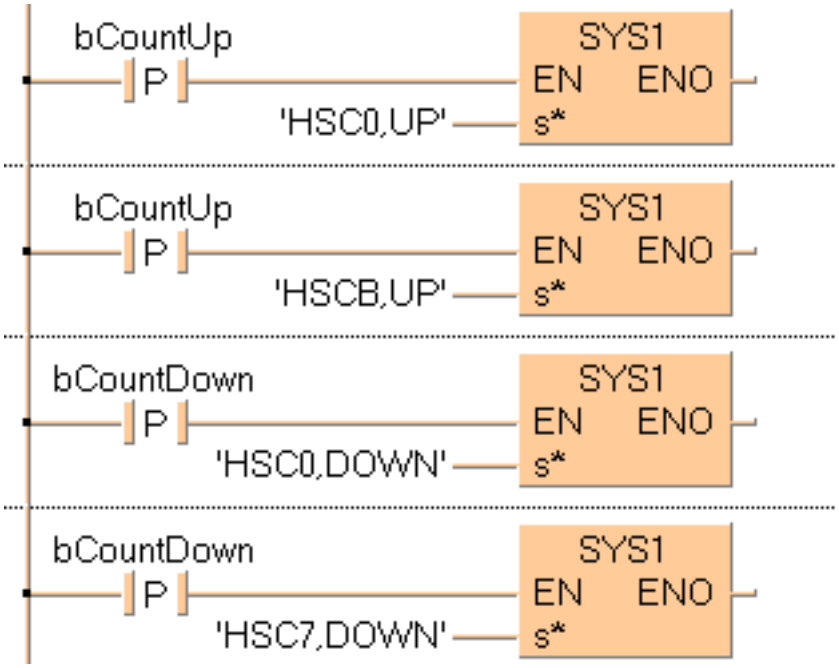
POE Kopf

	Klasse	Bezeichner	Typ	Initial
0	VAR	bCountUp	BOOL	FALSE
1	VAR	bCountDown	BOOL	FALSE

Rumpf

Wenn die Variable **bCountUp** auf TRUE gesetzt wird, wird die Funktion ausgeführt. Das Systemregister für den betreffenden Kanal wird auf Aufwärtszählung gesetzt. Wenn **bCountDown** auf TRUE gesetzt ist, wird der angegebene Kanal auf Abwärtszählung gesetzt.

LD

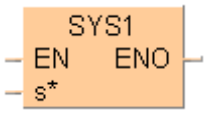


ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
INT_value:=TIME_TO_INT(time_value);
```

RS485: Ansprechzeit ändern

Dieser Befehl ändert in der RS485-Kommunikation die Ansprechzeit der COM- bzw. TOOL-Schnittstelle.

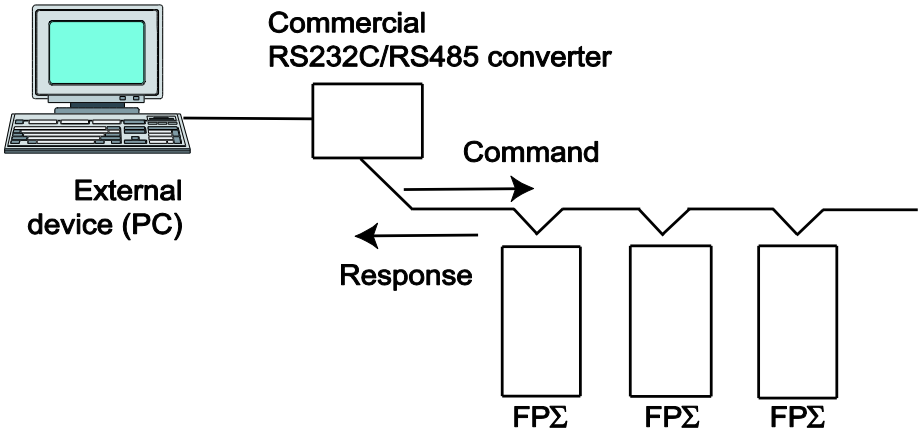


Das 1. Schlüsselwort spezifiziert die Schnittstelle, deren Ansprechzeit geändert werden soll; das 2. Schlüsselwort bestimmt die Ansprechzeit. Dieser Befehl wird verwendet, um die Ansprechzeit steuerungsseitig so weit zu verzögern, bis Befehle von einem externen Gerät gesendet und Antworten von der SPS empfangen werden können.

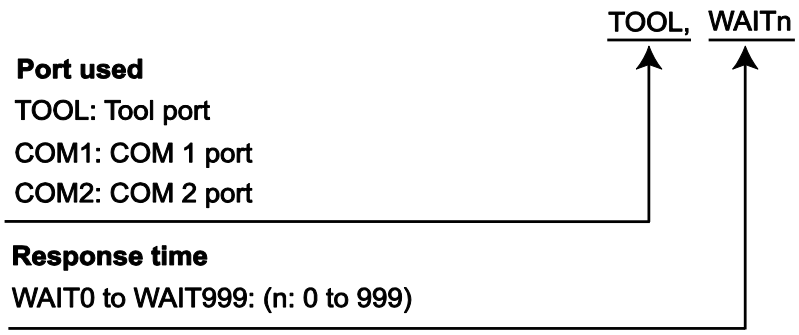
Trennen Sie die Schlüsselwörter durch ein Komma.

Anwendungsbeispiel

Wenn ein handelsüblicher RS232C/RS485-Umwandler verwendet wird, um die Kommunikation zwischen einem PC und einer FPΣ zu ermöglichen, gewährleistet dieser Befehl, dass die SPS erst antwortet, nachdem der Umwandler das Freigabesignal gesendet hat.



Schlüssel-wörter



Wenn die Kommunikationsschnittstelle auf PC-Kopplung gesetzt ist, entspricht die Ansprechzeit der Zykluszeit x n (n: 0 bis 999).

Wenn die Kommunikations-Schnittstelle auf SPS-Kopplung gesetzt ist, ist die Ansprechzeit n μs (n: 0 bis 999).

Bei n = 0, erfolgt keine Verzögerung.

Vorsichtsmaßnahmen bei der Programmierung

- Dieser Befehl gilt nur, wenn auf der SPS-Seite PC-Kopplung oder SPS-Kopplung eingestellt ist. Bei der Einstellung "Programmgesteuerte Kommunikation" kann er nicht verwendet werden.
- Die Systemregister-Einstellungen werden von diesem Befehl nicht beeinflusst.
- Es wird empfohlen, diesen Befehl nur bei steigender Flanke auszuführen.
- Wenn die SPS abgeschaltet wird, werden die Einstellungen gelöscht, die dieser Befehl gesetzt hat. (Der gesetzte Wert wird 0.) Wenn die SPS in den PROG-Modus geschaltet wird, nachdem der Befehl ausgeführt wurde, werden die Einstellung gehalten.
- Wenn ein handelsüblicher RS232C/RS485-Umwandler bei der SPS-Kopplung benutzt wird, sollten Sie diesen Befehl für alle zugeschalteten Teilnehmer (Steuerungen) programmieren.
- Trennen Sie die Schlüsselwörter durch ein Komma und fügen Sie keine Leerzeichen hinzu.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ etwas anderes als ein Schlüsselwort angegeben wird ▪ kein Komma zwischen dem 1. und 2. Schlüsselwort steht
R9008	%MX0.900.8	kurzzeitig	▪ das Schlüsselwort in Kleinbuchstaben geschrieben ist ▪ COM1 oder COM2 gesetzt ist und kein Schnittstellenmodul installiert wurde

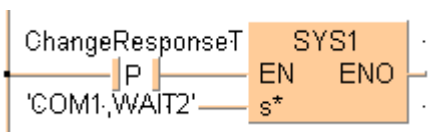
Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	ChangeResponseT	BOOL	FALSE	changes response time of RS485

Rumpf Wenn **ChangeResponseT** auf TRUE gesetzt wird, wird die Ansprechzeit für die COM1-Schnittstelle um 2 μs verzögert.

KOP



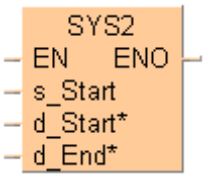
Die Zeichenkette, die an s* anliegt, darf am Ende keine Leerzeichen enthalten. In FPWIN Pro ab Version 4.0 werden diese Leerzeichen automatisch vom Compiler gelöscht.

SYS2 Systemregister für Koppelbereich einstellen

Erklärung

Mit SYS2 können die Systemregsiterwerte im Koppelbereich im RUN-Modus geändert werden. **s_Start** beinhaltet die neuen Werte für die Systemregister, die mit **d_Start*** und **d_End*** begrenzt werden.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Der Koppelbereich wird festgelegt durch die Systemregister 40-47 (bei der FP0R, FP-Σ 32k, FP-X außerdem 50-57).

Vorsichtsmaßnahmen bei der Programmierung

- Wenn dieser Befehl ausgeführt wird, wird der Inhalt des System-ROMs in der CPU nicht überschrieben. Das heißt, der Inhalt der Systemregister richtet sich beim Aus- und Wiedereinschalten nach den Software-Einstellungen.
- Sie sollten einen Wert zwischen 40 und 47 für **d_Start*** oder **d_End*** festlegen. Darüber hinaus sollte d_Start* ≤ d_End*.
- Da Systemregister-Einstellungen geändert werden, kann es sein, dass ein Fehler ausgegeben wird, wenn die Systemregister mit FPWIN Pro verglichen werden.

SPS-Typen **Verfügbarkeit von SYS2 (s. S. 1197)**

Datentypen

Variable	Datentyp	Funktion
s_Start	ANY16	Beinhaltet die neuen Werte für die Systemregister, die von den anderen beiden Variablen begrenzt werden.
d_Start*		Erstes Systemregister (von 40-47), dem ein neuer Wert zugewiesen wird. muss eine Konstante sein
d_End*		Letztes Systemregister (von 40-47), dem ein neuer Wert zugewiesen wird. muss eine Konstante sein

Operanden

Für	Merker				T/C		Register			Konstante
s_Start	-	-	-	-	-	-	DT	-	-	-
d_Start*	-	-	-	-	-	-	-	-	-	dez., hex.
d_End*	-	-	-	-	-	-	-	-	-	dez., hex.

Fehlermerker

Nr.	IEC-Adresse	Gesetzt	Wenn
R9007	%MX0.900.7	permanent	▪ d1 > d2 ▪ die festgelegten Werte außerhalb der Bereiche liegen, die für die einzelnen Systemregister definiert sind
R9008	%MX0.900.8	kurzzeitig	

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP) programmiert.

SDT Ein strukturierter Datentyp (SDT) kann aus mehreren Datentypen zusammengesetzt werden. Ein SDT wird zunächst im SDT-Pool definiert und dann wie die Standardtypen (BOOL, INT usw.) in der globalen Variablenliste oder im POE-Kopf verarbeitet.

SYS2_DUT [SDT]

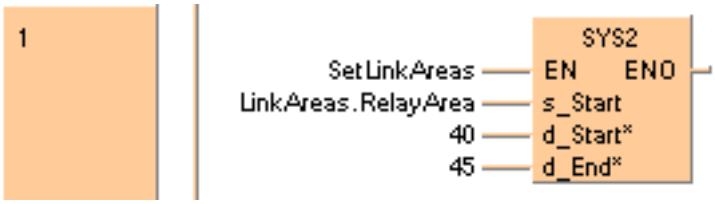
	Bezeichner	Typ	Initial
0	RelayArea	INT	0
1	RegisterArea	INT	0
2	RelaySendStart	INT	0
3	RelaySendSize	INT	0
4	RegisterSendStart	INT	0
5	RegisterSendSize	INT	0

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	LinkAreas	LINK_AREAS	RelayArea := 64, RegisterArea := 128
1	VAR	SetLinkAreas	BOOL	FALSE

Rumpf Die Werte für den Koppelbereich in den Systemregistern 40 bis 45 werden entsprechend den Einstellungen im SDT geändert, wenn **SetLinkAreas** auf TRUE gesetzt wird.

KOP

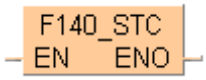


Kapitel 33

Spezielle Befehle

F140_STC — Carry-Flag setzen

Erklärung Der Sondermerker R9009 dient als Carry-Flag. Der Merker wird gesetzt, wenn die Ausführungsbedingung **EN** auf TRUE gesetzt wird. Dieser Befehl kann zusammen mit Befehlen verwendet werden, die das Carry-Flag R9009 verwenden (z.B. F122_RCR (s. S. 570) und F123_RCL (s. S. 572)).



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Beispiel In diesem Beispiel wird die Funktion F140_STC im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function; result after a leading edge from start: carry-flag (R9009) will be set ON
1	VAR				

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

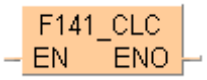
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F140_STC();
END_IF;
```

F141_CLC — Carry-Flag zurücksetzen

Erklärung

Der Sondermerker R9009 dient als Carry-Flag. Der Merker wird zurückgesetzt, wenn die Ausführungsbedingung **EN** auf TRUE gesetzt wird. Dieser Befehl kann zusammen mit Befehlen verwendet werden, die das Carry-Flag R9009 verwenden (z.B. F122_RCR (s. S. 570) und F123_RCL (s. S. 572)).



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function; result after a leading edge from start: carry-flag (R9009) will be set OFF
1	VAR				

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

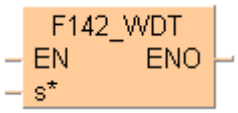
```
IF start THEN
    F141_CLC();
END_IF;
```

F142_WDT

Zykluszeitüberwachung einstellen (Watchdog timer)

Erklärung

Mit der dezimalen Konstanten **s*** wird eine Überwachungszeit für die Programmabarbeitung eingestellt. Der Wertebereich für **s*** ist 1 bis 255. Damit ergibt sich eine einstellbare Überwachungszeit von 2,5 ms * **s*** (637,5 ms).



Die Zykluszeitüberwachung spricht an, wenn die Zeit zur Programmabarbeitung eines Zyklus länger als die mit dem Systemregister 30 eingestellte Zeit dauert. Der voreingestellte Wert des Systemregisters 30 ist bei jedem neuen Programmbearbeitungszyklus maßgebend. Wollen Sie jedoch innerhalb des Zyklus bestimmte Programmteile zusätzlich mit einer kürzeren Zeit überwachen, können Sie mit dem Befehl F/P142_WDT eine entsprechende Überwachungszeit aktivieren. Bei Beginn des nächsten Zyklus ist dann wieder die im Systemregister 30 voreingestellte Zeit wirksam.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F142_WDT (s. S. 1187)

Datentypen

Variable	Datentyp	Funktion
s	INT	Spezifiziert Zykluszeitüberwachungswert

Beispiel

In diesem Beispiel wird die Funktion F142_WDT im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

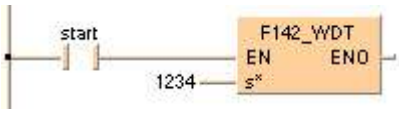
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    (* Watchdog timer value is changed to 123.4ms *)
    F142_WDT(1234);
END_IF;
```

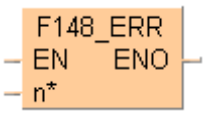
F148_ERR

Selbstdiagnose-Fehlercode

Schritte 3

Erklärung

Die durch **n*** angegebene Fehlernummer wird in die Systemvariable sys_iSelfDiagnosticErrorCode kopiert, die die zugehörigen Sonderdatenregister liest. Beim Setzen von **n***=0 werden alle Fehlernummern größer 43 gelöscht, und die ERROR-LED erlischt.



Gleichzeitig wird der Selbstdiagnose-Fehlermerker R9000 gesetzt und die ERROR-LED an der CPU leuchtet auf.

Der Inhalt des Fehlermerkers R9000 und die Fehlernummer lassen sich mit Control FPWIN Pro (**Monitor → Sondermerker und -datenregister → Basisfehler**) oder den zugehörigen Systemvariablen lesen und prüfen.

Fehlernummernbereiche:

Wenn **n*** = 100 bis 199, wird die Operation angehalten.

Wenn **n*** = 200 bis 299, wird die Operation fortgeführt.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen

Verfügbarkeit von F148_ERR (s. S. 1187)

Datentypen

Variable	Datentyp	Beschreibung
n*	ANY16	▪ muss eine Konstante sein ▪ Selbstdiagnose-Fehlercodenummer, Bereich: 0 und 100 bis 299 ▪ Siehe auch: SPS-Status in der Online-Hilfe

Operanden

Für	Merker				T/C		Register			Konstante
n*	-	-	-	-	-	-	-	-	-	dez. oder hex.

Fehlermerker

Nr.	IEC-Adresse	Setzen	Wenn
R9007	%MX0.900.7	dauerhaft	▪ **n** übersteigt Grenzwert.
R9008	%MX0.900.8	dauerhaft	

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

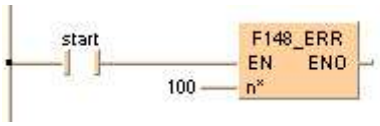
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

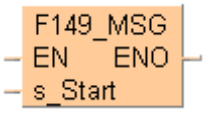
```
IF start THEN
    (* Sets the self-diagnostic error 100 *)
    (* The ERROR/ALARM LED of the PLC is on,
    and operation stops. *)
    F148_ERR(100);
END_IF;
```

F149_MSG

Anzeige Zeichenkonstante

Erklärung

Dieser Befehl wird verwendet, um die mit **s** festgelegte Meldung auf dem Display des Handprogrammiergerätes anzuzeigen.



Mit Aktivierung des Befehls **F149_MSG** wird der Sondermerker R9026 gesetzt und die Meldung in die Sonderdatenregister DT9030 bis DT9035 (DT90030 bis DT90035 bei FP0 T32CP, FP2/2SH, FP10/10S/10SH) geschrieben. Der Befehl **F149_MSG** wird nicht ausgeführt, wenn bereits eine Meldung angezeigt wird. Die Meldung kann mit dem Handprogrammiergerät gelöscht werden.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F149_MSG (s. S. 1187)**

Datentypen

Variable	Datentyp	Funktion
s	STRING(12)	Anzuzeigende Nachricht

Operanden

Für	Merker				T/C		Register			Konstante
s	-	-	-	-	-	-	-	-	-	Zeichen

Beispiel

In diesem Beispiel wird die Funktion F149_MSG im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

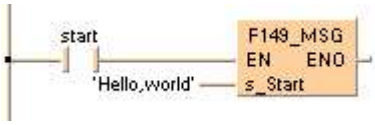
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

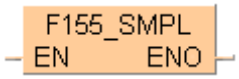
```
IF start THEN
    F149_MSG('Hello, world');
END_IF;
```

F155_SMPL

Abtastdaten übertragen

Erklärung Dieser Befehl überträgt die vom Abtasteditor angegebenen Abtastdaten in den Abtastspeicher.

Der Befehl F155_SMPL lässt sich nur im Abtastmodus "pro Zyklus" verwenden.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Abtastarten mit Control FPWIN Pro angeben:

Merker: max. 16 Ein-/Ausgänge
Verfügbar für (FP-Format): X, Y, R, L, T, C

Daten: 3 Worte
Verfügbar für (FP-Format): WX, WY, WR, WL, SV, EV, DT, LD, FL

SPS Typen **Verfügbarkeit von F155_SMPL (s. S. 1187)**

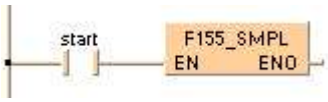
Beispiel In diesem Beispiel wird die Funktion F155_SMPL im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function

Rumpf Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F155_SMPL();
END_IF;
```

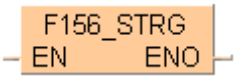
F156_STRG

Abtast-Trigger setzen

Erklärung

Dieser Befehl setzt den Abtast-Trigger, der das Abtasten nach der Verzögerung stoppt, die durch die Parameter für das Abtasten im Trace angegeben wurde.

Der Befehl F156_STRG lässt sich mit beiden Abtastmodi, "pro Zyklus" und "pro Zeitintervall", verwenden.



Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

Abtastarten mit Control FPWIN Pro angeben:

Merker: max. 16 Ein-/Ausgänge
Verfügbar für (FP-Format): X, Y, R, L, T, C

Daten: 3 Worte
Verfügbar für (FP-Format): WX, WY, WR, WL, SV, EV, DT, LD, FL

SPS-Typen	Verfügbarkeit von F156_STRG (s. S. 1187)

Beispiel

In diesem Beispiel wird die Funktion F156_STRG im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP

ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    F156_STRG();
END_IF;
```

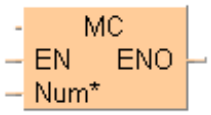
Kapitel 34

Steuerbefehle

MC

Haupt-Kontrollrelais

Erklärung Die Befehle **MC** und **MCE** schließen einen Programmbereich ein, der in Abhängigkeit vom Signalzustand der Eingangsbedingung des MC-Befehls abgearbeitet oder übersprungen wird.



Ist der Signalzustand der Eingangsbedingung "0", wird der eingeschlossene Programmbereich nicht abgearbeitet.

Ein Haupt-Kontrollbefehlssatz (MC und MCE) kann im Programm auch innerhalb eines anderen Haupt-Kontrollbefehlssatzes stehen. Diese Form heißt "verschachtelt".

Ein Haupt-Kontrollbefehlssatz wird begrenzt durch die Marken MC und MCE mit der gleichen Konstante am Eingang **Num***.

☞

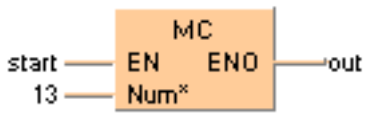
- **Diese Funktion kann nicht in einem Funktionsbaustein verwendet werden.**
- **Der maximale mögliche Wert für Num* hängt vom SPS-Typ ab.**

SPS-Typen **Verfügbarkeit von MC (s. S. 1194)**

Datentypen

Variable	Datentyp	Funktion
Num*	Konstante	Konstante zur Identifizierung der Marke MCE, die das Programm begrenzen soll

Beispiel

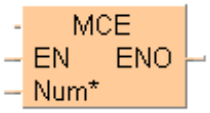


MCE

Ende Haupt-Kontrollrelais

Erklärung

Die Befehle MC (s. S. 1029) und **MCE** schließen einen Programmbereich ein, der in Abhängigkeit vom Signalzustand der Eingangsbedingung des MC-Befehls abgearbeitet oder übersprungen wird.



Ist der Signalzustand der Eingangsbedingung "0", wird der eingeschlossene Programmbereich nicht abgearbeitet.

Ein Haupt-Kontrollbefehlssatz (MC und MCE) kann im Programm auch innerhalb eines anderen Haupt-Kontrollbefehlssatzes stehen. Diese Form heißt "verschachtelt".

Ein Haupt-Kontrollbefehlssatz wird begrenzt durch die Marken MC und MCE mit der gleichen Konstante am Eingang **Num***.

☞

- **Diese Funktion kann nicht in einem Funktionsbaustein verwendet werden.**
- **Der maximale mögliche Wert für Num* hängt vom SPS-Typ ab.**

SPS-Typen **Verfügbarkeit von MCE (s. S. 1194)**

Datentypen

Variable	Datentyp	Funktion
Num*	Konstante	Konstante zur Identifizierung der Marke MC, die das Programm begrenzen soll

Beispiel

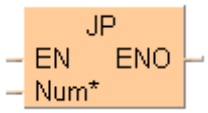
In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet.

```
AWL   KOP     start   (* EN = start; Starting signal for the MC/MCE function. *)
      MC      1       (* 1 = Num* *)
                      (* ... *)
                      (* Execute or execute not this program part. *)
                      (* ... *)
      MCE     1       (* 1 = Num* *)
```

JP — Sprung zu einer Marke

Erklärung

Die Befehle **JP** (JUMP) und **LBL** (LABEL) schließen einen Programmbereich ein, der übersprungen wird, wenn die Eingangsbedingung **EN** des Sprungbefehls **JP** TRUE ist.



Am Eingang **Num*** der Funktionen **JP** und **LBL** muss die gleiche Konstante anliegen. Wenn ein**JP**-Befehl ausgeführt wird, verkürzt sich die Zykluszeit um die Ausführungszeit der übersprungenen Befehle. Innerhalb eines Programms können mehr als zwei **JP**-Befehle mit der gleichen Konstante **Num*** verwendet werden. Es dürfen jedoch nicht mehrere **LBL**-Befehle die gleiche Nummer besitzen. **LBL**-Befehle werden als Sprungmarken für die Befehle **JP**, LOOP (s. S. 1033) und F19_SJP (s. S. 1033) verwendet.

Ein Befehlspaar (**JP** und **LBL**) kann im Programm auch innerhalb eines anderen Befehlspaares stehen. Diese Form heißt "verschachtelt".

☞

- **Diese Funktion kann nicht in einem Funktionsbaustein verwendet werden.**
- **Der maximale mögliche Wert für Num* hängt vom SPS-Typ ab.**

SPS-Typen **Verfügbarkeit von JP (s. S. 1194)**

Datentypen

Variable	Datentyp	Funktion
Num*	Konstante	Konstante zur Identifizierung der Marke LBL, die das Programm begrenzen soll

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet.

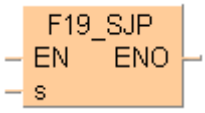
```
AWL   LD    start   (* EN = start; Starting signal for the JP function. *)
      JP    1       (* Num* = 1 (Address of Label) *)
```

F19_SJP

Indirekter Sprung zu einer Marke

Erklärung

Wenn die Eingangsbedingung zur Ausführung des Sprungbefehls **F19_SJP** erfüllt ist, wird die Programmabarbeitung bei der Marke **LBL** fortgesetzt, deren Nummer am Eingang **s** angegeben ist.



Die Nummer **s** der Marke LBL kann zwischen 0 und 255 liegen.

Dieser Befehl existiert auch als P-Befehl (für die SPS-Typen FP2/2SH, FP3/5, FP10/10SH), der nur bei steigender Flanke am EN-Eingang ausgeführt wird. Wählen Sie im Dialog "Befehle" die Schaltfläche **[P-Befehl]**, wenn Sie einen P-Befehl benötigen. Eine Wiederverwendung des Befehls ist einfach, denn er erscheint dann in der Liste "Zuletzt verwendet". Drücken Sie im Programmierbereich **<Strg>+<Umsch>+<v>**, um die Liste der kürzlich verwendeten Elemente anzuzeigen.

SPS-Typen **Verfügbarkeit von F19_SJP (s. S. 1188)**

Datentypen

Variable	Datentyp	Funktion
s	ANY16	Speichert Nummer der Marke (0 bis 255)

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	KOP	FL	-

Beispiel

In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf

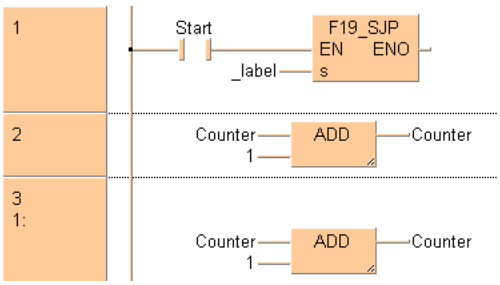
Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

	Klasse	Bezeich...	Typ	Initial
0	VAR	Start	BOOL	FALSE
1	VAR	_label	INT	1
2	VAR	Counter	INT	0

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
(* if Start is true Counter will be incremented by 1, else by 2 *)
IF Start THEN
     F19_SJP(_label);
END_IF;
Counter:=Counter+1;
LBL(1);
Counter:=Counter+1;
```

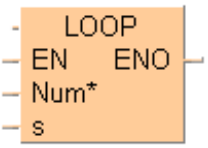
LOOP

Mehrfacher Sprung zu einer Marke

Erklärung

Die Befehle LOOP und LBL (s. S. 1034) schließen einen Programmbereich ein, der abhängig von der Anzahl der Sprünge **s** des Schleifenbefehls **LOOP** mehrmals abgearbeitet wird.

Symbol:



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

Mit der Variablen **Num*** des Befehls **LBL** wird die Nummer der Marke, zu der gesprungen werden soll angegeben. Innerhalb eines Programms können nicht mehr als zwei **LBL**-Befehle mit der gleichen Konstante **Num*** verwendet werden. Wenn Sie am Eingang **s** den Wert 0 festlegen, wird der Befehl **LOOP** nicht ausgeführt.

SPS-Typen

Verfügbarkeit von LOOP (s. S. 1194)

- **Diese Funktion kann nicht in einem Funktionsbaustein verwendet werden.**
- **Der maximale mögliche Wert für Num* hängt vom SPS-Typ ab.**

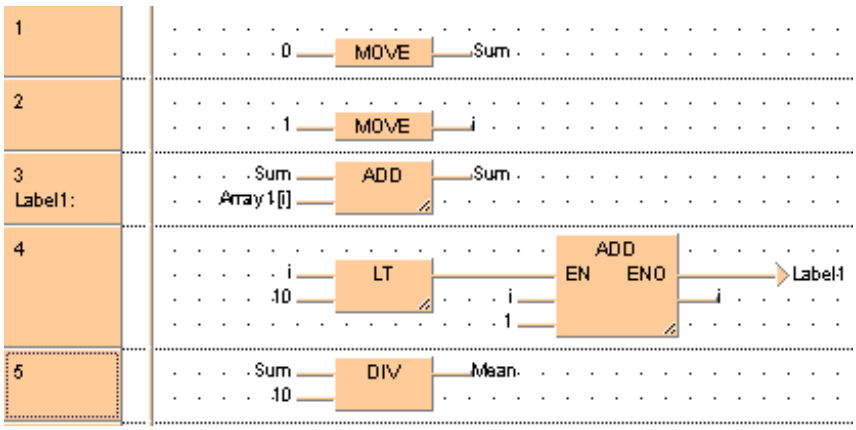
Datentypen

Variable	Datentyp	Funktion
s	INT, WORD	Einstellung
Num*	Konstante	Konstante zur Identifizierung der Marke LBL, die das Programm begrenzen soll; die Schleife wird so oft ausgeführt, bis die Variable **s** den Wert 0 erreicht.

Operanden

Für	Merker				T/C		Register			Konstante
s	WX	WY	WR	WL	SV	EV	DT	LD	FL	-

Beispiel

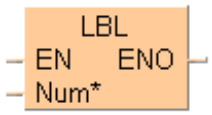


LBL

Marke; Sprungziel für die JP- und LOOP-Befehle

Erklärung

Die Befehle **JP** und **LBL** (Label) schließen einen Programmbereich ein, der in Abhängigkeit vom Signalzustand der Eingangsbedingung des Sprungbefehls JP abgearbeitet oder übersprungen wird. Mit der Variablen Num* wird die Nummer der Marke, zu der gesprungen werden soll angegeben.



Die Befehle LOOP (s. S. 1033) und **LBL** schließen einen Programmbereich ein, der abhängig von der Anzahl der Sprünge **s** des Schleifenbefehls **LOOP** mehrmals abgearbeitet wird. Mit der Variablen Num* wird die Nummer der Marke, zu der gesprungen werden soll angegeben.

- **Diese Funktion kann nicht in einem Funktionsbaustein verwendet werden.**
- **Der maximale mögliche Wert für Num* hängt vom SPS-Typ ab.**

SPS-Typen

Verfügbarkeit von LBL (s. S. 1194)

Datentypen

Variable	Datentyp	Funktion
Num*	Konstante	Konstante, die übereinstimmen muss mit der Sprungmarke im entsprechenden JP-, LOOP- oder F19-Befehl

Beispiel

In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.

POE-Kopf

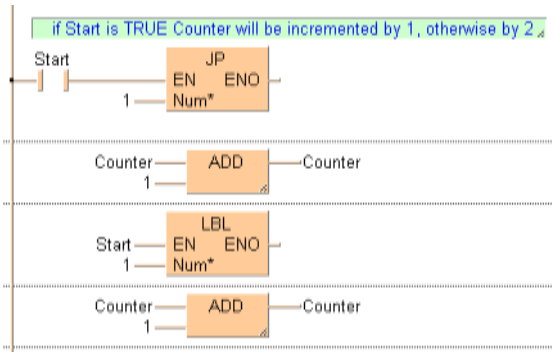
Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Start	BOOL	FALSE
1	VAR	Counter	INT	0

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
(* if Start is true Counter will be incremented by 1, else by 2 *)
IF Start THEN
    JP (1);
END_IF;

Counter:=Counter+1;
LBL (1);
Counter:=Counter+1;
```

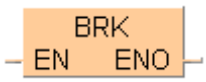
BRK

Break

Erklärung

Der Befehl **BRK** (Breakpoint = Haltepunkt) stoppt die Ausführung an der Adresse des BRK-Befehls im Testlaufmodus, wenn der Trigger **EN** TRUE ist.

Sobald der Befehl ausgeführt ist, wird das Programm angehalten. Um die Programmausführung wieder fortzusetzen, sollte der Modus im Testlauf (fortlaufend oder schrittweise) gewählt werden. Im schrittweisen Modus wird das Programm Befehl für Befehl abgearbeitet, unabhängig von der Art der Befehle. Im fortlaufenden Modus wird das Programm so lange ausgeführt, bis es vom nächsten BRK-Befehl angehalten wird, oder bis die Befehlsausführung zu Ende ist (Endebefehl ED).



☞ **Der Modus Testlauf wird ausgeführt, wenn der Betriebsarten-Wahlschalter der SPS auf RUN und der Initialisierungs-/Testmodus-Wahlschalter auf TEST steht.**

Beispiel

In diesem Beispiel wird die Funktion im Kontaktplan (KOP) und im strukturierten Text (ST) programmiert. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf

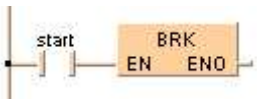
Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	activates the function

Rumpf

Wenn die Variable **start** auf TRUE gesetzt wird, wird die Funktion ausgeführt.

KOP



ST

Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
IF start THEN
    BRK();
END_IF;
```

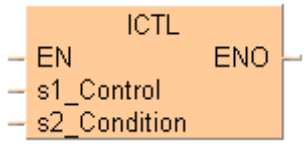
ICTL

Interrupt Control (Interruptsteuerung)

Erklärung

Der Befehl **ICTL** wird zur Einstellung der Interruptsteuerung verwendet. Mit jeder Ausführung des Befehls ICTL können die Interruptparameter wie Interruptart und Status (aktiviert/deaktiviert) neu festgelegt werden. Die Parameter werden mit **s1** und **s2** festgelegt.

- **s1** 16-Bit-Konstante oder 16-Bit-Speicherregister zur Spezifizierung von Interruptart und Funktion von s2.
- **s2** 16-Bit-Konstante oder 16-Bit-Speicherregister zur Auswahl der zu steuernden Interruptprogramme.



Verfügbare Interrupt-Programme:

- **16** über Interrupteingänge gesteuerte Interrupt-Programme (INT 0 bis INT 15)
- **8** über spezielle Module (z.B. Positioniermodule) gesteuerte Interrupt-Programme (INT 16 bis INT 23)
- **1** Zeitgesteuertes Interrupt-Programm (INT 24) (Zeitabstand 0,5ms einstellbar für FP2/2SH, FP10SH)

Durch Voranstellen eines DF-Befehls (oder durch die Festlegung einer steigenden Flanke im KOP) vor den Befehl ICTL wird sichergestellt, dass der Befehl nur bei steigender Flanke ausgeführt wird.

Ein und dieselbe Ausführungsbedingung kann mehrere ICTL-Befehle steuern.

Bit	**15 .. 8**	**7 .. 0**
s1 16#	**Funktion des Registers s2** **00**: Interruptmaske **01**: Interrupt löschen	**Art des Interrupts** **00**: Interrupteingang (INT 0-15) **01**: Spezielle Module (INT 16-23) **02**: Zeitinterrupt (INT 24)
s2 2#	Bit 0: **0** Interrupt-Programm 0 nicht ausgewählt Bit 0: **1** Interrupt-Programm 0 ausgewählt Bit 1: **0** Interrupt-Programm 1 nicht ausgewählt ... Bit 15: **1** Interrupt-Programm 15 ausgewählt Beispiel: s2 = 2#0000000000001010	

☞	• **Der aktuelle Status (maskiert/nicht maskiert) des Programms, das in die Interruptroutine eingetragen ist, kann im Sonderdatenregister DT90025 überwacht werden.** • **Der aktuelle Status (maskiert/nicht maskiert) des Programms, das nicht in die Interruptroutine eingetragen ist, kann im Sonderdatenregister DT90026 überwacht werden.** • **Das aktuelle Intervall des Zeit-Interrupts kann im Sonderdatenregister DT90027 überwacht werden.** • **Ein in eine Interruptroutine eingetragenes Interrupt-Programm wird automatisch durch die Initialisierung beim Programmstart aktiviert.** • **Mit dem ICTL-Befehl kann eine Interruptroutine durch das Programm aktiviert oder deaktiviert werden.**
SPS-Typen	**Verfügbarkeit von ICTL (s. S. 1193)**

Datentypen

Variable	Datentyp	Funktion
s1	ANY16	Funktion des Registers **s2** und Art des Interrupts
s2		Auswahl des Interrupt-Programms

Operanden

Für	Merker			T/C		Register			Konstante
s1, s2	WY	WR	WL	SV	EV	DT	LD	FL	dez., hex.

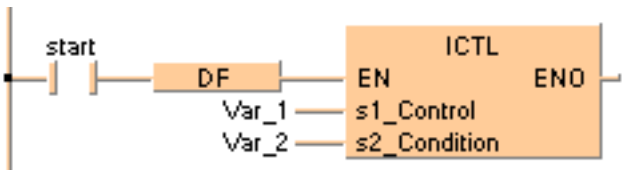
Beispiel In diesem Beispiel wird für alle Programmiersprachen der gleiche POE-Kopf verwendet. Ein Beispiel für AWL (Anweisungsliste) finden Sie in der Online-Hilfe.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial	Kommentar
0	VAR	start	BOOL	FALSE	enable signal
1	VAR	input_value_1	WORD	16#0002	first input parameter
2	VAR	input_value_2	WORD	10	second input parameter

Rumpf In diesem Beispiel wird bei Aktivierung der Startbedingung start für INT24 der Zeitinterrupt auf den Abstand 100ms (die Zeitbasis 10ms ist gewählt) gesetzt.

KOP



Kapitel 35

Analogmodulbefehle

Unit_AnalogInOut_FP0_A21

Auf Modul FP0-A21 schreiben und davon lesen

Erklärung

Diese Funktion schreibt digitale Werte in den Ausgangskanal des Analogmoduls FP0-A21 und liest digitale Umwandlungswerte aus dessen Eingangskanälen. Der digitale Wert, der umgewandelt und als analoger Wert ausgegeben werden soll, wird bei **iOutChannel** eingegeben. Die Daten aus dem Analogmodul werden je nach Kanal in den Ausgangsvariablen **iInChannel0** und **iInChannel1** gespeichert.

REFERENZ

In der Online-Hilfe finden Sie nur eine kurze Übersicht zu den Einstellungen und der Verdrahtung des DIP-Schalters. Weitere technische Informationen entnehmen Sie bitte dem Handbuch Analogmodule für FP0 und FP-Sigma, ACGM0158, auf der Installations-CD von Control FPWIN Pro.

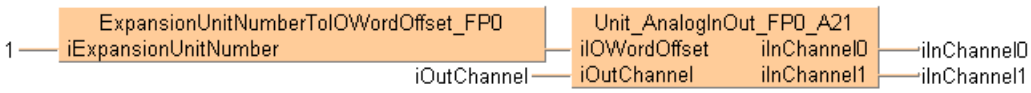
SPS Typen s. S. 1198

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iOutChannel	INT	0
1	VAR	iModuleOffSetWX	INT	0
2	VAR	iInChannel0	INT	0
3	VAR	iInChannel1	INT	0

KOP Verwenden Sie ExpansionUnitNumberToIOWordOffset_FP0 (s. S. 1053) um den Wort-Offset des an eine FP0, FP0R oder FPΣ angeschlossenen Analogmoduls zu berechnen.



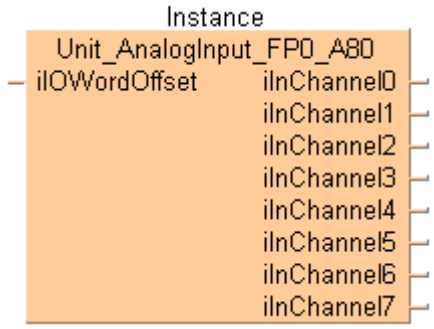
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Unit_AnalogInOut_FP0_A21(iIOWordOffset := iIOWordOffset,
                         iOutChannel := iOutChannel,
                         iInChannel0 => iInChannel0,
                         iInChannel1 => iInChannel1);
```

Unit_AnalogInput_FP0_A80

Daten von Modul FP0-A80 lesen

Erklärung Dieser Funktionsbaustein liest die konvertierten digitalen Daten von den analogen Eingangskanälen des FP0-A80-Moduls. Die digitalen Daten werden je nach Kanal in den Ausgangsvariablen **iInChannel0–iInChannel7** gespeichert.



REFERENZ

In der Online-Hilfe finden Sie nur eine kurze Übersicht zu den Einstellungen und der Verdrahtung des DIP-Schalters. Weitere technische Informationen entnehmen Sie bitte dem Handbuch Analogmodule für FP0 und FP-Sigma, ACGM0158, auf der Installations-CD von Control FPWIN Pro.

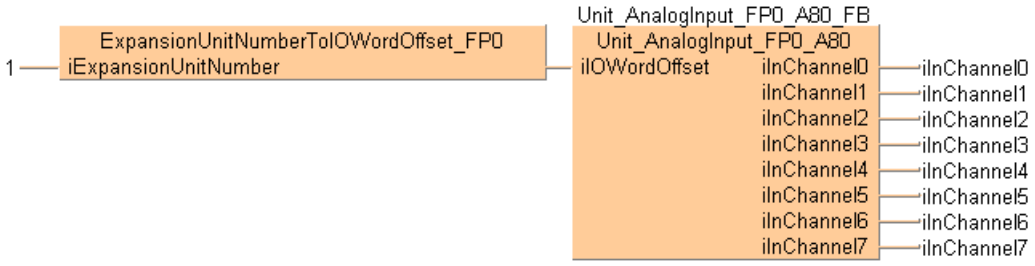
SPS Typen s. S. 1198

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	Uint_AnalogInput	Unit_AnalogInput_FP0_A80	
1	VAR	iModuleOffsetWX	INT	0
2	VAR	iInChannel0	INT	0
3	VAR	iInChannel1	INT	0
4	VAR	iInChannel2	INT	0
5	VAR	iInChannel3	INT	0
6	VAR	iInChannel4	INT	0
7	VAR	iInChannel5	INT	0
8	VAR	iInChannel6	INT	0
9	VAR	iInChannel7	INT	0

KOP Verwenden Sie ExpansionUnitNumberToIOWordOffset_FP0 (s. S. 1053) um den Wort-Offset des an eine FP0, FP0R oder FPΣ angeschlossenen Analogmoduls zu berechnen.



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

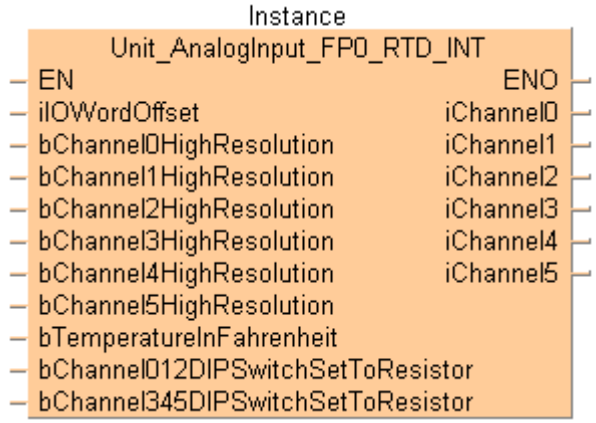
```
Unit_AnalogInput_FP0_A80_FB(iIOWordOffset := iIOWordOffset,
                            iInChannel0 => iInChannel0,
                            iInChannel1 => iInChannel1,
                            iInChannel2 => iInChannel2,
                            iInChannel3 => iInChannel3,
                            iInChannel4 => iInChannel4,
                            iInChannel5 => iInChannel5,
                            iInChannel6 => iInChannel6,
                            iInChannel7 => iInChannel7);
```

Unit_AnalogInput_ FP0_RTD_INT

Daten von Modul FP0-RTD6 lesen

Erklärung Dieser Funktionsbaustein liest die konvertierten digitalen Daten von den analogen Eingangskanälen des FP0-RTD6-Moduls. Die digitalen Daten werden je nach Kanal in den Ausgangsvariablen **iChannel0–iChannel5** gespeichert. Die gespeicherten digitalen Werte sind vom Datentyp INTEGER.

Zum Messen der RTD-Eingangsdaten können Sie folgende Geräte nutzen: Pt100 (gemäß IEC751), Pt1000 (gemäß IEC751), Ni1000 (gemäß DIN43760) oder einen Widerstand.



- **Zwischen dem Einschalten und der ersten gültigen Datenumwandlung ändert sich der Digitalwert zu 8191 bzw. 16383. Achten Sie darauf, dass im Programm keine Daten aus diesem Zeitraum verarbeitet werden.**
- **Bei einem Drahtbruch ändert sich der Digitalwert zu 8191 bzw. 16383. Achten Sie auf eine drahtbruchsichere Programmierung! Ein defektes Widerstandsthermometer muss ausgetauscht werden.**

REFERENZ

In der Online-Hilfe finden Sie nur eine kurze Übersicht zu den Einstellungen und der Verdrahtung des DIP-Schalters. Weitere technische Informationen entnehmen Sie dem Handbuch Analogmodule für FP0 und FP-Sigma, ACGM0158, auf der Installations-CD von Control FPWIN Pro.

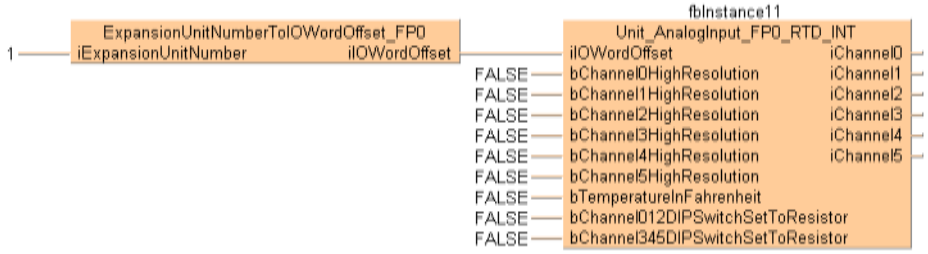
SPS Typen siehe 1198

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Class	Identifier	Type
0	VAR	fbInstance11	Unit_AnalogInput_FP0_RTD_INT

KOP Verwenden Sie ExpansionUnitNumberToIOWordOffset_FP0 (s. S. 1053) um den Wort-Offset des an eine FP0, FP0R oder FPΣ angeschlossenen Analogmoduls zu berechnen.



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

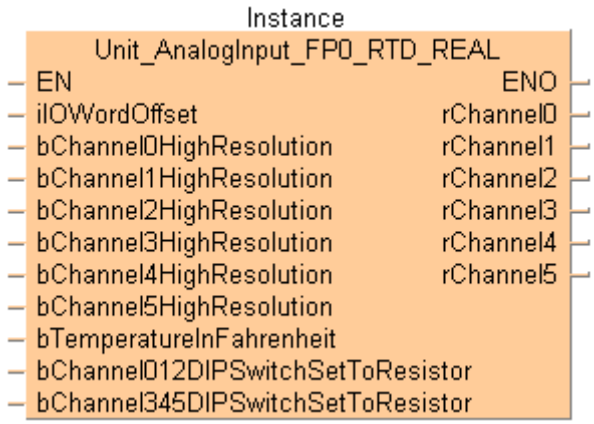
```
fbInstance11(iIOWordOffset := iIOOffsetFP0,
             bChannel0HighResolution := bHighResolutionChannel0,
             bChannel1HighResolution := bHighResolutionChannel1,
             bChannel2HighResolution := bHighResolutionChannel2,
             bChannel3HighResolution := bHighResolutionChannel3,
             bChannel4HighResolution := bHighResolutionChannel4,
             bChannel5HighResolution := bHighResolutionChannel5,
             bTemperatureInFahrenheit := bHighResolutionChannel6,
             bChannel012DIPSwitchSetToResistor := bSetToResistor012,
             bChannel345DIPSwitchSetToResistor := bSetToResistor345,
             iChannel0 => iIn1,
             iChannel1 => iIn2,
             iChannel2 => iIn3,
             iChannel3 => iIn4,
             iChannel4 => iIn5,
             iChannel5 => iIn6);
```

Unit_AnalogInput_ FP0_RTD_REAL

Daten von Modul FP0-RTD6 lesen

Erklärung Dieser Funktionsbaustein liest die konvertierten digitalen Daten von den analogen Eingangskanälen des FP0-RTD6-Moduls. Die digitalen Daten werden je nach Kanal in den Ausgangsvariablen **iChannel0–iChannel5** gespeichert. Die gespeicherten digitalen Werte sind vom Datentyp REAL.

Zum Messen der RTD-Eingangsdaten können Sie folgende Geräte nutzen: Pt100 (gemäß IEC751), Pt1000 (gemäß IEC751), Ni1000 (gemäß DIN43760) oder einen Widerstand.



- **Zwischen dem Einschalten und der ersten gültigen Datenumwandlung ändert sich der Digitalwert zu 8191 bzw. 16383. Achten Sie darauf, dass im Programm keine Daten aus diesem Zeitraum verarbeitet werden.**
- **Bei einem Drahtbruch ändert sich der Digitalwert zu 8191 bzw. 16383. Achten Sie auf eine drahtbruchsichere Programmierung! Ein defektes Widerstandsthermometer muss ausgetauscht werden.**

REFERENZ

In der Online-Hilfe finden Sie nur eine kurze Übersicht zu den Einstellungen und der Verdrahtung des DIP-Schalters. Weitere technische Informationen entnehmen Sie dem Handbuch Analogmodule für FP0 und FP-Sigma, ACGM0158, auf der Installations-CD von Control FPWIN Pro.

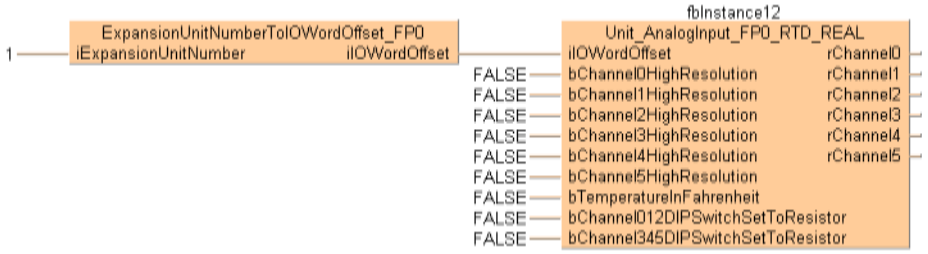
SPS Typen siehe 1198

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ
0	VAR	fbInstance11	Unit_AnalogInput_FP0_RTD_INT

KOP Verwenden Sie ExpansionUnitNumberToIOWordOffset_FP0 (s. S. 1053) um den Wort-Offset des an eine FP0, FP0R oder FPΣ angeschlossenen Analogmoduls zu berechnen.



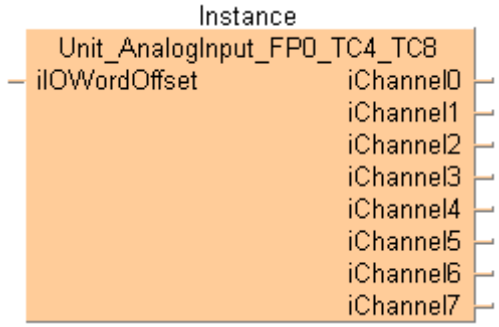
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
fbInstance12(iIOWordOffset := iIOOffsetFP0,
             bChannel0HighResolution := bHighResolutionChannel0,
             bChannel1HighResolution := bHighResolutionChannel1,
             bChannel2HighResolution := bHighResolutionChannel2,
             bChannel3HighResolution := bHighResolutionChannel3,
             bChannel4HighResolution := bHighResolutionChannel4,
             bChannel5HighResolution := bHighResolutionChannel5,
             bTemperatureInFahrenheit := bHighResolutionChannel6,
             bChannel012DIPSwitchSetToResistor := bSetToResistor012,
             bChannel345DIPSwitchSetToResistor := bSetToResistor345);
             rReal0 := fbInstance7.rChannel0;
             rReal1 := fbInstance7.rChannel1;
             rReal2 := fbInstance7.rChannel2;
             rReal3 := fbInstance7.rChannel3;
             rReal4 := fbInstance7.rChannel4;
```

Unit_AnalogInOut_ FP0_TC4_TC8

Von Modul FP0-TC4/FP0-TC8 lesen

Erklärung Dieser Funktionsbaustein liest die konvertierten digitalen Daten von den analogen Eingangskanälen des Moduls FP0-TC4 (vier analoge Eingangskanäle) oder FP0-TC8 (acht analoge Eingangskanäle). Die digitalen Daten werden je nach Kanal in den Ausgangsvariablen **iChannel0–iChannel3** für FP0-TC4 und **iChannel0–iChannel7** für FP0-TC8 gespeichert.



REFERENZ

Weitere technische Informationen entnehmen Sie bitte dem Handbuch Analogmodule für FP0 und FP-Sigma, ACGM0158, auf der Installations-CD von Control FPWIN Pro.

SPS Typen s. S. 1198

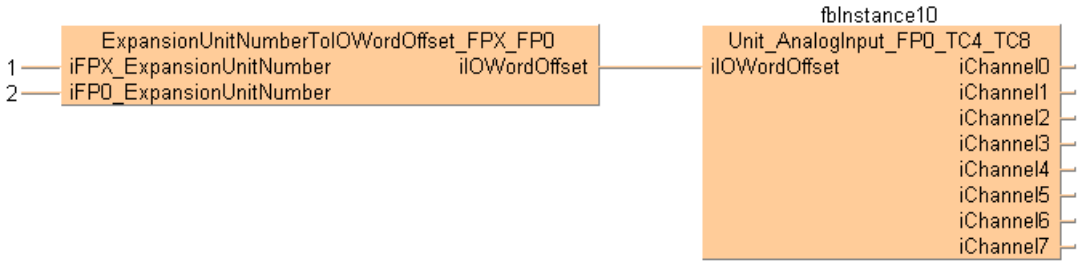
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ
0	VAR	fbInstance10	Unit_AnalogInput_FP0_TC4_TC8

Rumpf Verwenden Sie ExpansionUnitNumberToIOWordOffset_FP0 (s. S. 1053) um den Wort-Offset des an eine FP0, FP0R oder FPΣ angeschlossenen Analogmoduls zu berechnen.

KOP

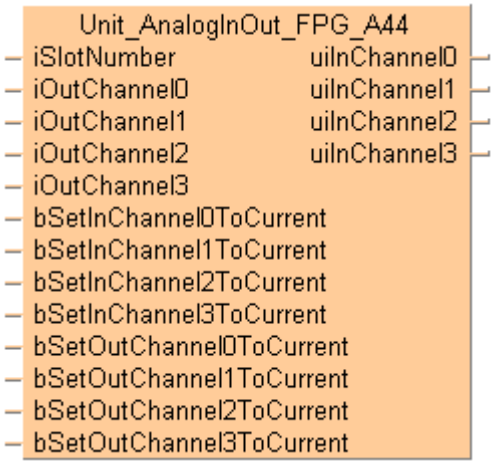


```
ST fbInstance10(iIOWordOffset := iIOOffsetFP0);
                iIn1 := fbInstance10.iChannel0;
                iIn2 := fbInstance10.iChannel1;
                iIn3 := fbInstance10.iChannel2;
                iIn4 := fbInstance10.iChannel3;
                iIn5 := fbInstance10.iChannel4;
                iIn6 := fbInstance10.iChannel5;
                iIn7 := fbInstance10.iChannel6;
                iIn8 := fbInstance10.iChannel7;
```

Unit_AnalogInOut_FPG_A44

Auf Modul FPG-A44 schreiben und davon lesen

Erklärung Diese Funktion schreibt digitale Werte in den Ausgangskanal des Analogmoduls FPG-A44 und liest digitale Umwandlungswerte aus dessen Eingangskanälen. Die digitalen Werte, die umgewandelt und als analoge Werte ausgegeben werden sollen, werden bei **iOutChannel0–iOutChannel3** eingegeben. Die Daten aus dem Analogmodul werden je nach Kanal in den Ausgangsvariablen **iOutChannel0**-**iOutChannel3** gespeichert.

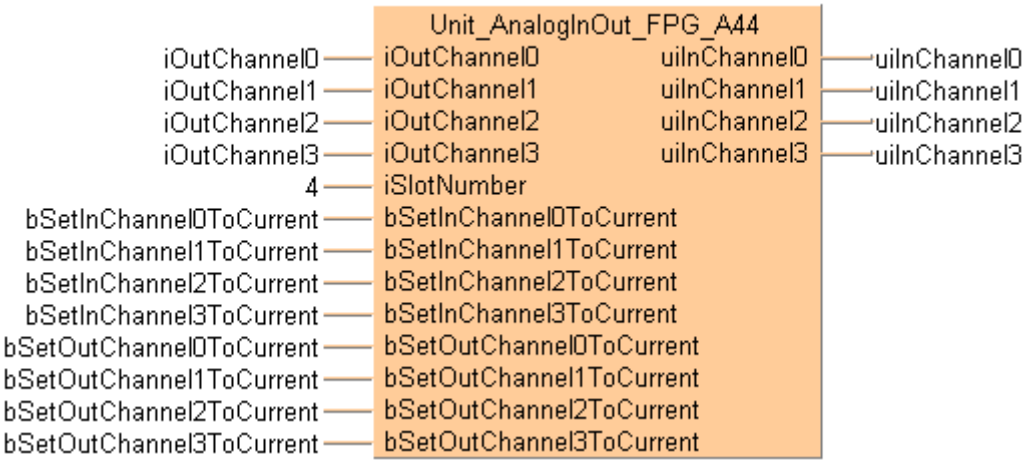


Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iOutChannel0	INT	0
1	VAR	iOutChannel1	INT	0
2	VAR	iOutChannel2	INT	0
3	VAR	iOutChannel3	INT	0
4	VAR	bSetInChannel0ToCurrent	BOOL	FALSE
5	VAR	bSetOutChannel0ToCurrent	BOOL	FALSE
6	VAR	bSetInChannel1ToCurrent	BOOL	FALSE
7	VAR	bSetInChannel2ToCurrent	BOOL	FALSE
8	VAR	bSetInChannel3ToCurrent	BOOL	FALSE
9	VAR	bSetOutChannel1ToCurrent	BOOL	FALSE
10	VAR	bSetOutChannel2ToCurrent	BOOL	FALSE
11	VAR	bSetOutChannel3ToCurrent	BOOL	FALSE
12	VAR	uiInChannel0	UINT	0
13	VAR	uiInChannel1	UINT	0
14	VAR	uiInChannel2	UINT	0
15	VAR	uiInChannel3	UINT	0

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

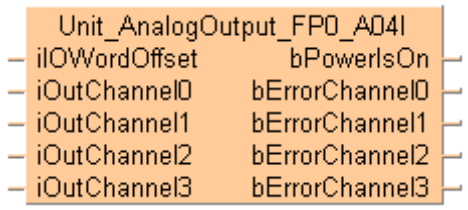
```
Unit_AnalogInOut_FPG_A44(iOutChannel0 := iOutChannel0,
                         iOutChannel1 := iOutChannel1,
                         iOutChannel2 := iOutChannel2,
                         iOutChannel3 := iOutChannel3,
                         iSlotNumber := 4,
                         bSetInChannel0ToCurrent :=
bSetInChannel0ToCurrent,
                         bSetInChannel1ToCurrent :=
bSetInChannel1ToCurrent,
                         bSetInChannel2ToCurrent :=
bSetInChannel2ToCurrent,
                         bSetInChannel3ToCurrent :=
bSetInChannel3ToCurrent,
                         bSetOutChannel0ToCurrent :=
bSetOutChannel0ToCurrent,
                         bSetOutChannel1ToCurrent :=
bSetOutChannel1ToCurrent,
                         bSetOutChannel2ToCurrent :=
bSetOutChannel2ToCurrent,
                         bSetOutChannel3ToCurrent :=
bSetOutChannel3ToCurrent,
                         uiInChannel0 => uiInChannel0,
                         uiInChannel1 => uiInChannel1,
                         uiInChannel2 => uiInChannel2,
                         uiInChannel3 => uiInChannel3);
```

Unit_AnalogOutput _FP0_A04I

In Modul FP0-A04V schreiben

Erklärung

Diese Funktion schreibt digitale Werte in die analogen Ausgangskanäle des Stromausgangstyps FP0-A04I. Die digitalen Werte, die umgewandelt und als analoge Werte ausgegeben werden sollen, werden bei **iOutChannel0–iOutChannel3** eingegeben.



REFERENZ

Weitere technische Informationen entnehmen Sie bitte dem Handbuch Analogmodule für FP0 und FP-Sigma, ACGM0158 auf der Installations-CD von Control FPWIN Pro.

SPS Typen s. S. 1198

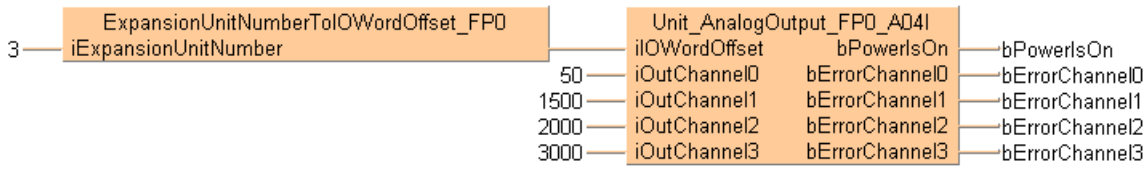
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bEnable	BOOL	FALSE
1	VAR	fbInstance2	E_Unit_AnalogOutput_FP0_A04I	
2	VAR			

Rumpf Die bei **iOutChannel0–iOutChannel3** eingegebenen digitalen Werte werden in das Analogmodul geschrieben und in analoge Werte umgewandelt. Die analogen Daten werden an den entsprechenden Ausgangskanälen ausgegeben.

KOP Verwenden Sie ExpansionUnitNumberToIOWordOffset_FP0 (s. S. 1053) um den Wort-Offset des an eine FP0, FP0R oder FPΣ angeschlossenen Analogmoduls zu berechnen.

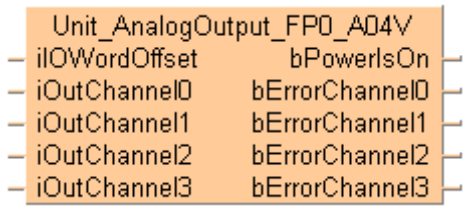


```
ST Unit_AnalogOutput_FP0_A04I(iIOWordOffset := iIOWordOffset,
                              iOutChannel0 := 50,
                              iOutChannel1 := 1500,
                              iOutChannel2 := 2000,
                              iOutChannel3 := 3000,
                              bPowerIsOn => bPowerIsOn,
                              bErrorChannel0 => bErrorChannel0,
                              bErrorChannel1 => bErrorChannel1,
                              bErrorChannel2 => bErrorChannel2,
                              bErrorChannel3 => bErrorChannel3);
```

Unit_AnalogOutput_FP0_A04V

In Modul FP0-A04V schreiben

Erklärung Diese Funktion schreibt digitale Werte in die analogen Ausgangskanäle des Spannungsausgangstyp FP0-A04. Die digitalen Werte, die umgewandelt und als analoge Werte ausgegeben werden sollen, werden bei **iOutChannel0–iOutChannel3** eingegeben.



REFERENZ

Weitere technische Informationen entnehmen Sie bitte dem Handbuch Analogmodule für FP0 und FP-Sigma, ACGM0158 auf der Installations-CD von Control FPWIN Pro.

SPS Typen s. S. 1198

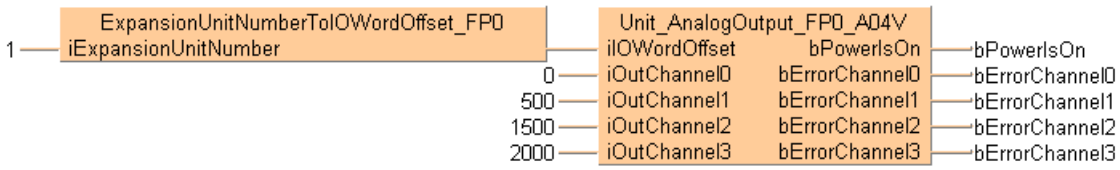
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bEnable	BOOL	FALSE
1	VAR	fbInstance2	E_Unit_AnalogOutput_FP0_A04I	
2	VAR			

Rumpf Die bei **iOutChannel0–iOutChannel3** eingegebenen digitalen Werte werden in das Analogmodul geschrieben und in analoge Werte umgewandelt. Die analogen Daten werden an den entsprechenden Ausgangskanälen ausgegeben.

KOP Verwenden Sie ExpansionUnitNumberToIOWordOffset_FP0 (s. S. 1053) um den Wort-Offset des an eine FP0, FP0R oder FPΣ angeschlossenen Analogmoduls zu berechnen.



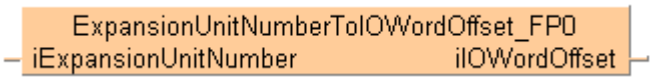
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Unit_AnalogOutput_FP0_A04V(iIOWordOffset := iIOWordOffset,
                           iOutChannel0  := 0,
                           iOutChannel1  := 500,
                           iOutChannel2  := 1500,
                           iOutChannel3  := 2000,
                           bPowerIsOn    => bPowerIsOn,
                           bErrorChannel0  => bErrorChannel0,
                           bErrorChannel1  => bErrorChannel1,
                           bErrorChannel2  => bErrorChannel2,
                           bErrorChannel3  => bErrorChannel3);
```

ExpansionUnitNumberToIOWordOffset_FP0

E/A-Offset der Analogmodule für die FP0R/FP-Sigma berechnen

Erklärung Dieser Befehl berechnet den Wort-Offset beim Anschluss eines FP0/FP0R-Analogmoduls an eine FP0, FP0R oder FPΣ.



REFERENZ

Weitere technische Informationen entnehmen Sie bitte dem Handbuch Analogmodule für FP0 und FP-Sigma, ACGM0158, auf der Installations-CD von Control FPWIN Pro.

SPS Typen s. S. 1186

Datentypen

Eingangsvariable	Datentyp	Beschreibung
iExpansionUnitNumber	INT	Positionsnummer des FP0/FP0R-Erweiterungsmoduls 1–3

Ausgangsvariable	Datentyp	Beschreibung
iIOWordOffset	INT	Offset des E/A-Worts (WX/WY 2/4/6)

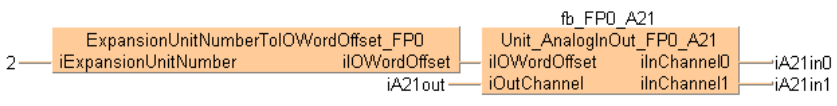
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. (ST).Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	fb_FP0_A21	Unit_AnalogInOut_FP0_A21	
1	VAR	iA21out	INT	0
2	VAR	iA21in0	INT	0
3	VAR	iA21in1	INT	0

Rumpf Der Funktionsbaustein wandelt die Positionsnummer des FP0-Erweiterungsmoduls 1–3, wobei 0 der CPU der FP0, FP0R oder FPΣ entspricht, in den entsprechenden E/A-Wort-Offset 2, 4 oder 6 um.

KOP



FP0 CPU	Exp. 1	A04I 2	Exp. 3
WX	2	4	6
WY	2	4	6

iExpansionUnitNumber: 2
--> iIOWordOffset = 2 (WX/WY 4)

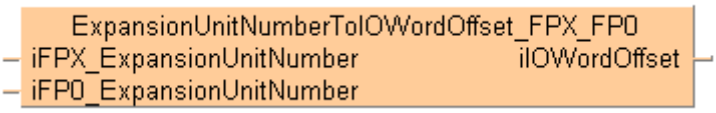
ST

```
Unit_AnalogInOut_FP0_A21(iIOWordOffset :=
ExpansionUnitNumberToIOWordOffset_FP0(2),
                         iOutChannel  := iA21out,
                         iInChannel0  => iA21in0,
                         iInChannel1  => iA21in1);
```

ExpansionUnitNumberToIOWordOffset_FPX_FP0

E/A-Offset der Analogmodule für die FP-X/FP-X0 berechnen

Erklärung Dieser Befehl berechnet den Wort-Offset beim Anschluss eines FP0/FP0R-Analogmoduls an eine FP-X oder FP-X0.



REFERENZ

Weitere technische Informationen entnehmen Sie bitte dem Handbuch Analogmodule für FP0 und FP-Sigma, ACGM0158, auf der Installations-CD von Control FPWIN Pro.

SPS Typen s. S. 1186

Datentypen

Eingangsvariable	Datentyp	Beschreibung
iFPX_ExpansionUnitNumber	INT	Positionsnummer des FP0/FP0R-Erweiterungsmoduls: 1–8
iFP0_ExpansionUnitNumber	INT	Positionsnummer des FP0/FP0R-Erweiterungsmoduls 1–3

Ausgangsvariable	Datentyp	Beschreibung
ilOWordOffset	INT	Offset des E/A-Worts (WX/WY)

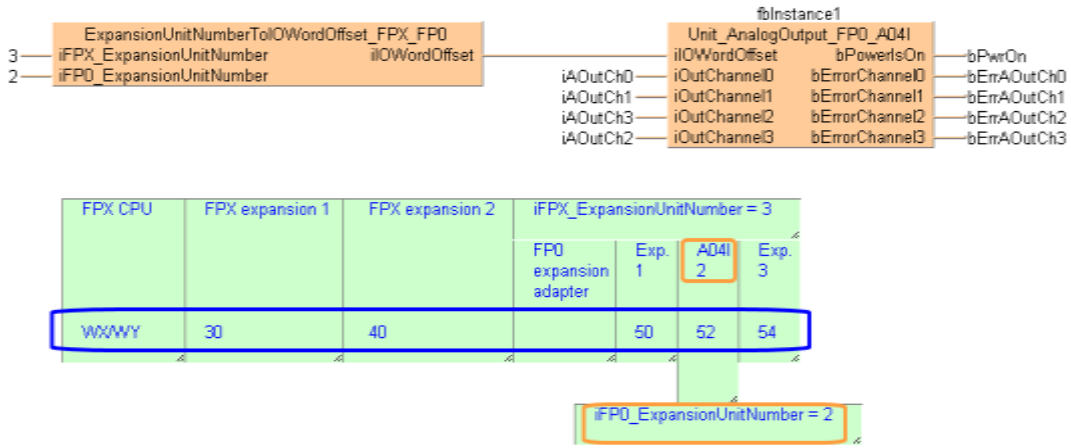
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	fbInstance1	Unit_AnalogOutput_FP0_A04I	
1	VAR	iAOutCh0	INT	0
2	VAR	iAOutCh1	INT	0
3	VAR	iAOutCh2	INT	0
4	VAR	iAOutCh3	INT	0
5	VAR	bPwrOn	BOOL	FALSE
6	VAR	bErrAOutCh0	BOOL	FALSE
7	VAR	bErrAOutCh1	BOOL	FALSE
8	VAR	bErrAOutCh2	BOOL	FALSE
9	VAR	bErrAOutCh3	BOOL	FALSE

Rumpf Dieser Befehl wertet den E/A-Wort-Offset eines FP0/FP0R-Analogmoduls aus, das an eine FP-X-CPU angeschlossen ist. **iFPX_ExpansionUnitNumber** ist der Installationsort des FP0/FP0R-Erweiterungsadapters (FP-X-Erweiterungsmodulnummer 1–8).
iFP0_ExpansionUnitNumber ist der Installationsort des FP0/FP0R-Analogmoduls in Bezug auf das FP0/FP0R-Erweiterungsmodul (Modulnummer 1–3).

KOP



ST

```
Unit_AnalogOutput_FP0_A04I(iIOWordOffset :=
ExpansionUnitNumberToIOWordOffset_FPX_FP0(3, 2),
                           iOutChannel0 := iAOutCh0,
                           iOutChannel1 := iAOutCh1,
                           iOutChannel2 := iAOutCh2,
                           iOutChannel3 := iAOutCh2,
                           bPowerIsOn => bPwrOn,
                           bErrorChannel0 => bErrAOutCh0,
                           bErrorChannel1 => bErrAOutCh1,
                           bErrorChannel2 => bErrAOutCh2,
                           bErrorChannel3 => bErrAOutCh3);
```

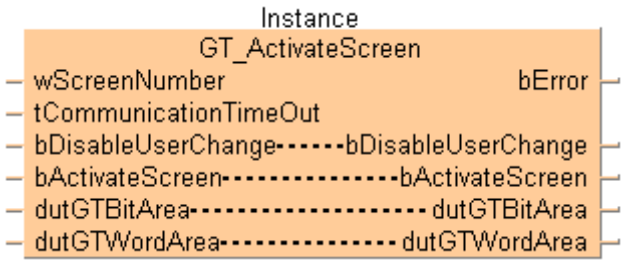
Kapitel 36

GT-Bediengerätebefehle

GT_CtrlActivateScreen

GT-Bildschirm steuern

Erklärung Funktionsbaustein, der von der SPS aus eine bestimmte GT-Seite mit den Variablen aktiviert oder ändert, die in der Tabelle der Datentypen beschrieben sind.



SPS Typen s. S. 1192

Datentypen

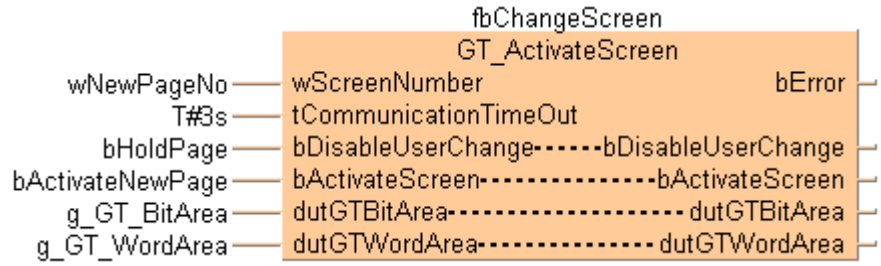
Eingangsvariable	**Datentyp**	**Beschreibung**
wScreenNumber	WORD	Seitennummer
tComTimeOut	TIME	Maximale Kommunikationswartezeit
Ein-/Ausgangsvariable	**Datentyp**	**Beschreibung**
bDisableUserChange	BOOL	Seitenänderung durch GT-Berührung deaktivieren
bActivateScreen		Neue Seite aktivieren
dutGTBitArea	GT_CommunicationBitArea	Bit-Adresse der GT-Kommunikation
dutGTWordArea	GT_CommunicationWordArea	Wort-Adresse der GT-Kommunikation
Ausgangsvariable	**Datentyp**	**Beschreibung**
bError	BOOL	Wird aktiviert, wenn die Seite nicht innerhalb der maximalen Kommunikationswartezeit wechselt

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR_EXTERNAL	g_GT_WordArea	GT_BasicComWordArea	
1	VAR_EXTERNAL	g_GT_BitArea	GT_BasicComBitArea	
2	VAR	bActivateNewPage	BOOL	FALSE
3	VAR	wNewPageNo	WORD	0
4	VAR	g_bStartPage	BOOL	FALSE
5	VAR	fbChangeScreen	GT_CtrlActivateScreen	
6	VAR	bHoldPage	BOOL	FALSE

KOP



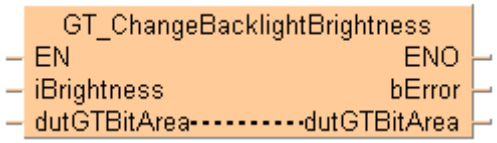
ST

```
fb_GT_ActivateScreen(wScreenNum := wNewPageNo,
                     tComTimeOut := T#3s,
                     bDisableUserChange := bHoldPage,
                     bActivateScreen := bActivateNewPage,
                     dutGTBitArea := g_GT_BitArea,
                     dutGTWordArea := g_GT_WordArea,
                     bErrorActivateScreen =>
    bErrorActivateScreen);
```

GT_ChangeBacklightBrightness

Helligkeit GT-Hintergrundbeleuchtung ändern

Erklärung Dieser Befehl ändert die Helligkeit des GT-Bediengeräts mit Hilfe der Variablen, die in der Datentyptabelle beschrieben sind.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen siehe (s. S. 1192)

Datentypen

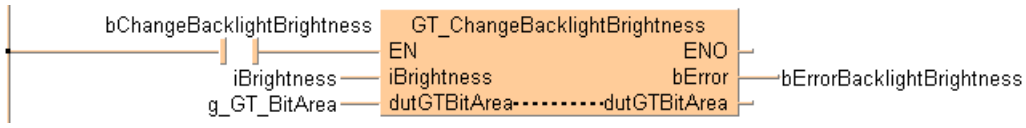
Eingangsvariable	Datentyp	Beschreibung
iBrightness	INT	Helligkeitswert 0–15
Ein-/Ausgangsvariable		
dutGTBitArea	GT_CommunicationBitArea	Bit-Adresse der GT-Kommunikation
Ausgangsvariable		
bError	BOOL	Wird aktiviert, wenn der Helligkeitswert außerhalb des gültigen Bereichs liegt

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet.Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR_EXTERNAL	g_GT_WordArea	GT_BasicComWordArea	
1	VAR_EXTERNAL	g_GT_BitArea	GT_BasicComBitArea	
2	VAR	bActivateNewPage	BOOL	FALSE
3	VAR	wNewPageNo	WORD	0
4	VAR	g_bStartPage	BOOL	FALSE
5	VAR	fbChangeScreen	GT_CtrlActivateScreen	
6	VAR	bHoldPage	BOOL	FALSE
7	VAR	iBrightness	INT	0
8	VAR	bErrorBacklightBrightness	BOOL	FALSE
9	VAR	bChangeBacklightBrightness	BOOL	FALSE

KOP



ST

```
fb_GT_ChangeBacklightBrightness((* EN := TRUE, *)
                                iBrightness := iBrightness,
                                dutGTBitArea := g_GT_BitArea,
                                bError => bErrorBacklightBrightness
                                (* , ENO => ?BOOL? *));
```

Kapitel 37

Schnelle Zählerbefehle

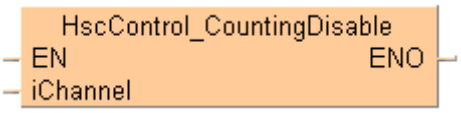
37.1 Steuerbefehle für schnelle Zähler

In diesem Abschnitt:

HscControl_Counting Disable

Schnellen Zähler deaktivieren

Erklärung Dieser Befehl deaktiviert den Zähler am durch **iChannel** festgelegten schnellen Zählerkanal. Bit 1 des Steuercodes für den schnellen Zähler (s. S. 861) wird auf TRUE gesetzt.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscControl_CountingEnable (s. S. 1063)
- HscInfo_IsCountingDisabled (s. S. 1079)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

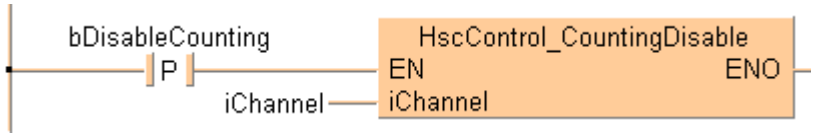
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bDisableCounting	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf Wenn die Variable **bDisableCounting** von FALSE zu TRUE wechselt, wird der Zähler am durch **iChannel** festgelegten Kanal deaktiviert.

KOP



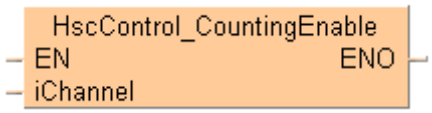
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bDisableCounting) then
    HscControl_CountingDisable(iChannel := iChannel);
end_if;
```

HscControl_CountingEnable

Schnellen Zähler aktivieren

Erklärung Dieser Befehl aktiviert den Zähler an einem durch **iChannel** festgelegten schnellen Zählerkanal, der zuvor mit HscControl_CountingDisable (s. S. 1062) deaktiviert worden ist. Bit 1 des Steuercodes für den schnellen Zähler (s. S. 861) wird auf FALSE gesetzt.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscControl_CountingDisable (s. S. 1062)
- HscInfo_IsCountingDisabled (s. S. 1079)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

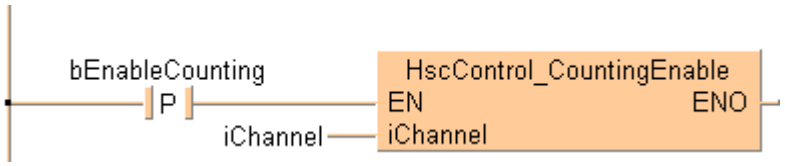
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bEnableCounting	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf Wenn die Variable **bEnableCounting** von FALSE zu TRUE wechselt, wird der Zähler am durch **iChannel** festgelegten Kanal aktiviert.

KOP



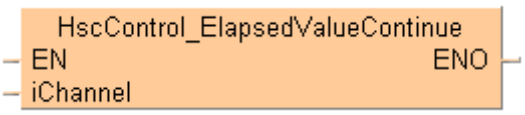
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bEnableCounting) then
    HscControl_CountingEnable(iChannel := iChannel);
end_if;
```

HscControl_ElapsedValueContinue

Zählen nach Reset fortsetzen

Erklärung Dieser Befehl setzt das Zählen auf einem mit **iChannel** festgelegten Kanal fort, dessen Istwert mit HscControl_ElapsedValueReset (s. S. 1066) auf 0 zurückgesetzt wurde. Bit 0 des Steuercodes für den schnellen Zähler (s. S. 861) wird auf FALSE gesetzt.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscControl_WriteElapsedValue (s. S. 1072)
- HscInfo_ReadElapsedValue (s. S. 1082)
- HscInfo_IsElapsedValueReset (s. S. 1080)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

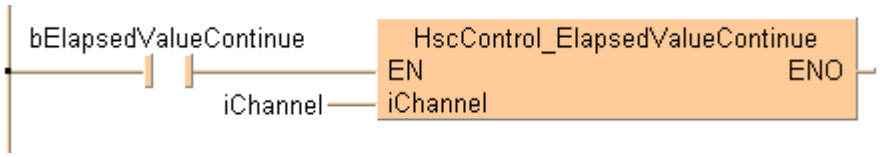
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bElapsedValueContinue	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf Wenn die Variable **bElapsedValueContinue** auf TRUE gesetzt wird, wird das Zählen am durch **iChannel** festgelegten Kanal fortgesetzt.

KOP



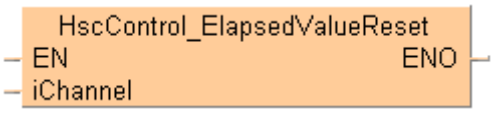
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if (bElapsedValueContinue) then
    HscControl_ElapsedValueContinue(iChannel := iChannel);
end_if;
```

HscControl_ElapsedValueReset

Istwert auf 0 setzen

Erklärung Dieser Befehl setzt den Istwert am durch **iChannel** festgelegten schnellen Zählerkanal auf 0. Bit 0 des Steuercodes für den schnellen Zähler (s. S. 861) wird auf TRUE gesetzt.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscControl_WriteElapsedValue (s. S. 1072)
- HscControl_ElapsedValueContinue (s. S. 1065)
- HscInfo_IsElapsedValueReset (s. S. 1080)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

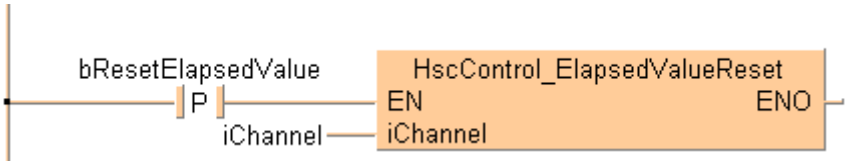
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bResetElapsedValue	BOOL	FALSE

Rumpf Wenn die Variable **bResetElapsedValue** von FALSE zu TRUE wechselt, wird der Istwert am durch **iChannel** festgelegten Kanal auf 0 gesetzt.

KOP



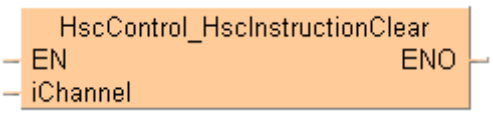
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF (bResetElapsedValue) then
    HscControl_ElapsedValueReset(iChannel := iChannel);
end_if;
```

HscControl_HscInstruction Clear

Schnellen Zählerbefehl löschen

Erklärung Dieser Befehl bricht die Ausführung eines schnellen Zählerbefehls am durch **iChannel** festgelegten Kanal ab. Bit 3 des Steuercodes für den schnellen Zähler (s. S. 861) wird auf TRUE gesetzt und anschließend wieder auf FALSE.



Siehe auch: Tool-Befehle für die schnellen Zähler (s. S. 1061)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

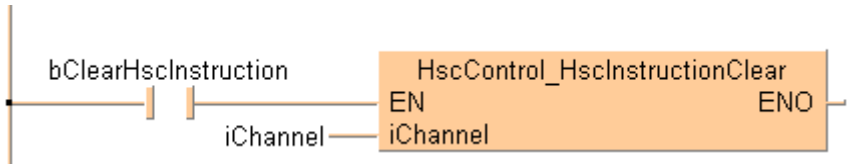
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bClearHscInstruction	BOOL	FALSE

Rumpf Wenn die Variable **bClearHscInstruction** auf TRUE gesetzt wird, wird die Ausführung des schnellen Zählerbefehls am durch **iChannel** festgelegten Kanal abgebrochen.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if (bClearHscInstruction) then
    HscControl_HscInstructionClear(iChannel := iChannel);
end_if;
```

HscControl_ResetInput Disable

Rücksetzeingang deaktivieren

Erklärung Dieser Befehl deaktiviert den Rücksetzeingang am durch **iChannel** festgelegten schnellen Zählerkanal. Bit 2 des Steuercodes für den schnellen Zähler (s. S. 861) wird auf TRUE gesetzt.

Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscInfo_IsResetInputDisabled (s. S. 1081)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

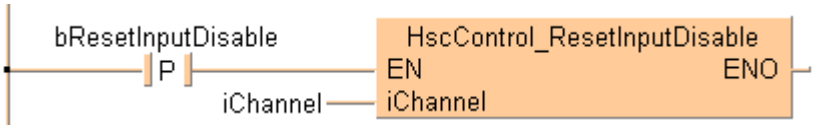
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bResetInputDisable	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf Wenn die Variable **bResetInputDisable** von FALSE zu TRUE wechselt, wird der Rücksetzeingang am durch **iChannel** festgelegten Kanal deaktiviert.

KOP



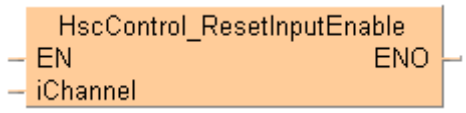
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bResetInputDisable) then
    HscControl_ResetInputDisable(iChannel := iChannel);
end_if;
```

HscControl_ResetInput Enable

Rücksetzeingang aktivieren

Erklärung Dieser Befehl aktiviert den Rücksetzeingang am durch **iChannel** festgelegten Kanal. Bit 2 des Steuercodes für den schnellen Zähler (s. S. 861) wird auf FALSE gesetzt.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscInfo_IsResetInputDisabled (s. S. 1081)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

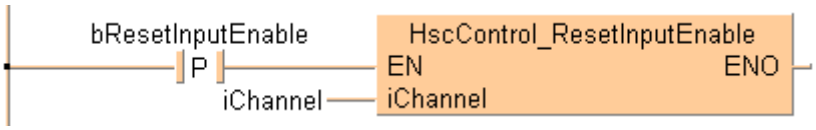
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bResetInputEnable	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf Wenn die Variable **bResetInputEnable** von FALSE zu TRUE wechselt, wird der Rücksetzeingang am durch **iChannel** festgelegten Kanal aktiviert.

KOP



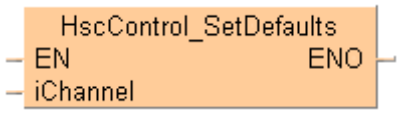
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bResetInputEnable) then
    HscControl_ResetInputEnable(iChannel := iChannel);
end_if;
```

HscControl_SetDefaults

Standardwerte für schnellen Zählerkanal einstellen

Erklärung Dieser Befehl setzt alle Bits des Steuercodes für den schnellen Zähler (s. S. 861) am durch **iChannel** festgelegten Kanal auf 0. 0 ist die Standardeinstellung.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

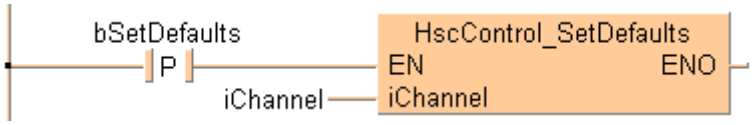
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bSetDefaults	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf Wenn die Variable **bSetDefaults** von FALSE zu TRUE wechselt, werden alle Einstellungen des durch **iChannel** festgelegten Kanals auf ihre Standardwerte gesetzt.

KOP



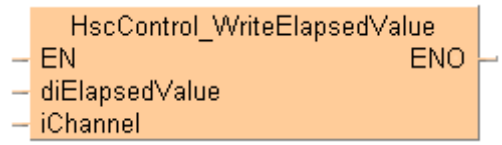
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bSetDefaults) then
    HscControl_SetDefaults(iChannel := iChannel);
end_if;
```

HscControl_WriteElapsedValue

Istwert in schnellen Zählerkanal schreiben

Erklärung Dieser Befehl schreibt einen Istwert in den durch **iChannel** festgelegten schnellen Zählerkanal.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscControl_ElapsedValueReset
- HscInfo_ReadElapsedValue (s. S. 1082)
- HscInfo_IsElapsedValueReset (s. S. 1080)

- FP-Befehle:

- Istwert schreiben und lesen (s. S. 861)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
diElapsedValue	DINT	In Kanal **iChannel** zu schreibender Istwert.
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

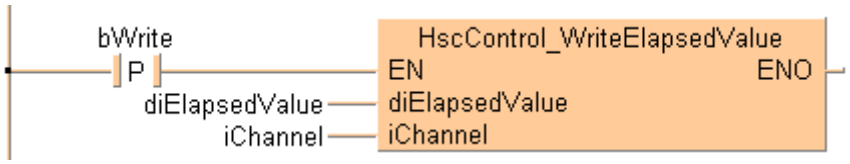
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bWrite	BOOL	FALSE
1	VAR	diElapsedValue	DINT	5000
2	VAR	iChannel	INT	0

Rumpf Wenn die Variable **bWrite** von FALSE zu TRUE wechselt, wird der in **diElapsedValue** angegebene Istwert in den durch **iChannel** festgelegten Kanal geschrieben.

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bWrite) then
    HscControl_WriteElapsedValue(diElapsedValue := diElapsedValue,
    iChannel := iChannel);
end_if;
```

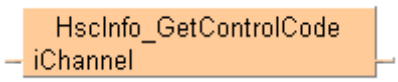
37.2 Informationsbefehle für schnelle Zähler

In diesem Abschnitt:

HscInfo_GetControlCode

Steuercode des schnellen Zählerkanals zurückgeben

Erklärung Dieser Befehl gibt den Steuercode (s. S. 861) des durch **iChannel** festgelegten schnellen Zählerkanals zurück.



Siehe auch: Tool-Befehle für die schnellen Zähler (s. S. 1061)

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	WORD	Speichert den Steuercode

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	wChannelControlCode	WORD	0

KOP

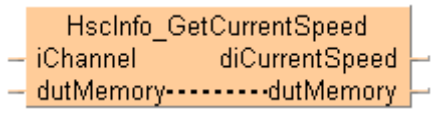
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
wChannelControlCode := HscInfo_GetControlCode(iChannel := iChannel);
```

HscInfo_GetCurrentSpeed

Aktuelle Geschwindigkeit des schnellen Zählerkanals zurückgeben

Erklärung Dieser Befehl gibt die aktuelle Zählgeschwindigkeit in Hz des durch **iChannel** festgelegten schnellen Zählerkanals zurück.



Siehe auch: Tool-Befehle für die schnellen Zähler (s. S. 1061)

SPS Typen s. S. 1193

Datentypen

Eingangsvariable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
dutMemory	SDT (s. S. 47)	HscInfo_GetCurrentSpeed_DUT
Ausgangsvariable		
diCurrentSpeed	DINT	Speichert die aktuelle Geschwindigkeit des durch **iChannel** festgelegten Kanals

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	Memory_DUT	HscInfo_GetCurrentSpeed_DUT	
2	VAR	diCurrentSpeed	DINT	0

KOP

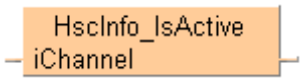
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
HscInfo_GetCurrentSpeed(iChannel :=iChannel,
dutMemory :=Memory_DUT,
diCurrentSpeed => diCurrentSpeed);
```

HscInfo_IsActive

Aktiven Status des schnellen Zählers prüfen

Erklärung Dieser Befehl wertet den Kontrollmerker des schnellen Zählers (s. S. 860) aus und gibt TRUE zurück, wenn der durch **iChannel** festgelegte schnelle Zählerkanal aktiv ist.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscControl_HscInstructionClear (s. S. 1068)

SPS Typen s. S. 1193

Datentypen

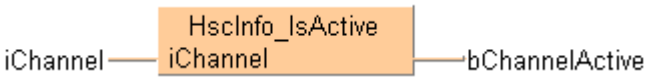
Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn der durch **iChannel** festgelegte schnelle Zählerkanal aktiv ist

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bChannelActive	BOOL	FALSE

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bChannelActive := HscInfo_IsActive (iChannel := iChannel);
```

HscInfo_IsChannelEnabled

Aktiven Status des schnellen Zählerkanals prüfen

Erklärung Dieser Befehl gibt TRUE zurück, wenn der durch **iChannel** festgelegte schnelle Zählerkanal in den Systemregistern aktiviert ist und durch den ausgewählten SPS-Typ unterstützt wird.

Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- Erforderliche Systemregistereinstellungen (s. S. 860)

SPS Typen s. S. 1193

Datentypen

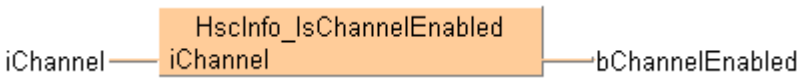
Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn der durch **iChannel** festgelegte Kanal aktiv ist

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel_Kopie	INT	0
1	VAR	bChannelEnabled	BOOL	FALSE

KOP



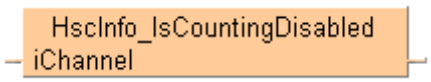
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bChannelEnabled := HscInfo_IsChannelEnabled (iChannel := iChannel);
```

HscInfo_IsCounting Disabled

Inaktiven Status des Zählers prüfen

Erklärung Dieser Befehl gibt TRUE zurück, wenn der Zähler am durch **iChannel** festgelegten Kanal deaktiviert wurde.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscControl_CountingDisable (s. S. 1062)
- HscControl_CountingEnable (s. S. 1063)

- FP-Befehle:

- Zählen aktivieren/deaktivieren (s. S. 861)

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn der Zähler am durch **iChannel** festgelegten Kanal inaktiv ist

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bCountingDisabled	BOOL	FALSE

KOP

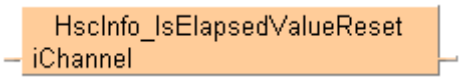
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bCountingDisabled := HscInfo_IsCountingDisabled(iChannel := iChannel);
```

HscInfo_IsElapsedValueReset

Istwert = 0 prüfen

Erklärung Dieser Befehl gibt TRUE zurück, wenn der Istwert des durch **iChannel** festgelegten schnellen Zählerkanals auf 0 zurückgesetzt wurde.



Siehe auch:

- Tool-Befehle:
- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscControl_ElapsedValueReset
- HscControl_ElapsedValueSet (s. S. 1065)
- FP-Befehle:
- Istwert zurücksetzen (Software-Reset) (s. S. 861)

SPS Typen s. S. 1193

Datentypen

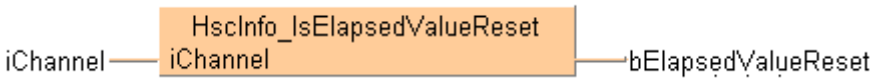
Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn der durch **iChannel** festgelegte Kanal zurückgesetzt wurde

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bElapsedValueReset	BOOL	FALSE

KOP



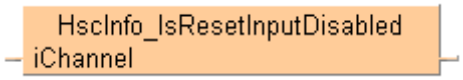
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bElapsedValueReset := HscInfo_IsElapsedValueReset(iChannel := iChannel);
```

HscInfo_IsResetInputDisabled

Inaktiven Status des Rücksetzeingangs prüfen

Erklärung Dieser Befehl gibt TRUE zurück, wenn der Rücksetzeingang am durch **iChannel** festgelegten Kanal deaktiviert ist.



Siehe auch:

- Tool-Befehle:
  - Tool-Befehle für die schnellen Zähler (s. S. 1061)
  - HscControl_ResetInputEnable
  - HscControl_ResetInputDisable (s. S. 1069)
- FP-Befehle:
  - Rücksetzeingang aktivieren/deaktivieren (Hardware-Reset) (s. S. 861)

SPS Typen s. S. 1193

Datentypen

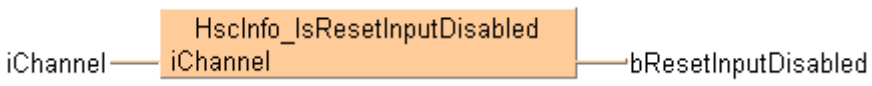
Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn der Rücksetzeingang am durch **iChannel** festgelegten Kanal inaktiv ist

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bResetInputDisabled	BOOL	FALSE

KOP



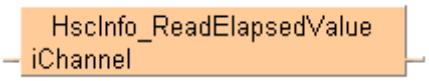
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if (bResetInput_Check) then
    bResetInputDisabled := HscInfo_IsResetInputDisabled(iChannel :=
iChannel);
end_if;
```

HscInfo_ReadElapsed Value

Istwert aus schnellem Zählerkanal lesen

Erklärung Dieser Befehl liest den Istwert aus dem durch **iChannel** festgelegten schnellen Zählerkanal.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)
- HscControl_WriteElapsedValue (s. S. 1072)
- HscControl_ElapsedValueReset (s. S. 1066)
- HscControl_ElapsedValueContinue (s. S. 1065)

- FP-Befehle:

- Istwert schreiben und lesen (s. S. 864)

SPS Typen s. S. 1193

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	DINT	Speichert den Istwert des durch **iChannel** festgelegten Kanals

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	diElapsedValue	DINT	0

KOP

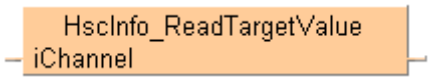
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
diElapsedValue := HscInfo_ReadElapsedValue (iChannel := iChannel);
```

HscInfo_ReadTargetValue

Sollwert aus schnellem Zählerkanal lesen

Erklärung Dieser Befehl liest den Sollwert aus dem durch **iChannel** festgelegten schnellen Zählerkanal.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)

Datentypen

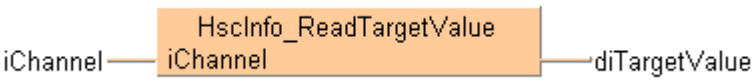
Variable	Datentyp	Beschreibung
iChannel	INT	Kanalnummer des schnellen Zählers: FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	DINT	Speichert den Sollwert des durch **iChannel** festgelegten Kanals

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	diTargetValue	DINT	0

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
diTargetValue := HscInfo_ReadTargetValue (iChannel := iChannel);
```

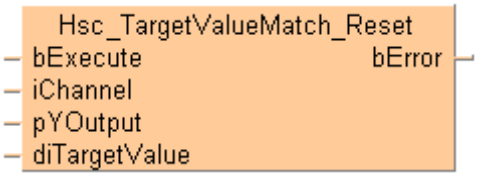
37.3 Zählervergleichsfunktionen des schnellen Zählers

In diesem Abschnitt:

Hsc_TargetValueMatch_Reset

Zählervergleichsausgang zurücksetzen (schneller Zähler)

Erklärung Wenn der Istwert den Sollwert **diTargetValue** des durch **iChannel** festgelegten schnellen Zählerkanals erreicht, wird der durch **pYOutput** festgelegte Ausgang sofort auf FALSE gesetzt.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die schnellen Zähler. Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F167_HighSpeedCounter_Reset (s. S. 884)

Um die Kombination von Kanal und Y-Ausgang zu prüfen, benötigt der Compiler das folgende Benennungsmuster für globale Variablen:
`%s`Hsc_TargetValueMatch_`Channel%d_Y%d_%s`
Verwenden Sie für globale Variablen in Zählervergleichsfunktionen grundsätzlich dieses Muster:

- `Channel%d` muss ein schneller Zählerkanal sein, der in den Systemregistern aktiviert ist.
- `Y%d` muss eine explizite, von der SPS unterstützte Ausgangsadresse sein.

FP-Σ, FP0, FP-e:	Y0–Y7
FP-Σ (V3.1 oder neuer), FP0R:	Y0–Y1F
FP-X:	Y0–Y29F

- `%s` ist ein optionaler Deskriptor am Musteranfang
- `%s` ist ein optionaler Deskriptor am Musterende

Optional	Vorgegebenes Muster	Optional
`g_b`	Hsc_TargetValueMatch_`ChannelA_Y11F`	`_MotorOn`

Diese globale Variable erzeugt den Code für Kanal **A** und Ausgang **Y11F**.

SPS Typen s. S. 1192

Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Eine steigende Flanke aktiviert die Funktion; Auswertung des Kontrollmerkers "Schneller Zähler" (s. S. 860) mit HscInfo_IsActive (s. S. 1077).
iChannel	INT	FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
pYOutput	POINTER	Durch GetPointer erhaltenes Zeigerergebnis von einer globalen Variable, die Kanalnummer und Ausgang liefert.
diTargetValue	DINT	Geben Sie für den Sollwert einen 32-Bit-Datenwert innerhalb des folgenden Bereichs ein: FP0, FP-e: -838808–+8388607 FPΣ, FP-X, FP0R: -2147483467–+2147483648

Ausgangsvariable	Datentyp	Beschreibung
iError	BOOL	TRUE, wenn die Kombination von **Channel%d** und **pYOuput.iOffset** nicht mit einer gültigen Kombination von Kanalnummer und Ausgang übereinstimmt, wie sie von der globalen Variablen vorgegeben wird.

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. (ST).Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Globale Variablen*

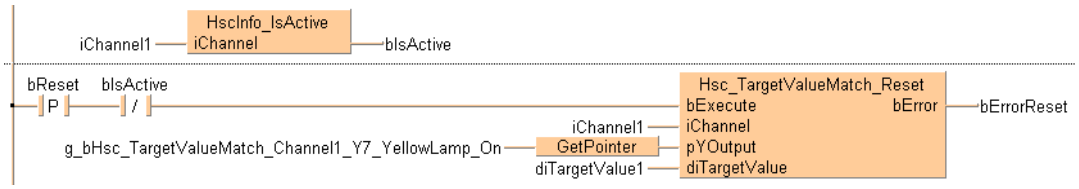
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
1	VAR_GLOBAL	g_bHsc_TargetValueMatch_Channel1_Y7_YellowLamp_On	Y7	%QX0.7	BOOL	FALSE

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR_EXTERNAL	g_bHsc_TargetValueMatch_Channel1_Y7_YellowLamp_On	BOOL	FALSE
1	VAR	bExecute	BOOL	0
2	VAR	iChannel1	INT	1
3	VAR	diTargetValue1	DINT	15000
4	VAR	bIsActive	BOOL	FALSE
5	VAR	diElapsedValue	DINT	0
6	VAR	bReset	BOOL	FALSE
7	VAR	bErrorReset	BOOL	FALSE

Rumpf Verwenden Sie HscInfo_IsActive (s. S. 1077), um den durch **iChannel1** festgelegten Kanal auszuwerten. Wenn eine steigende Flanke bei **bReset** erkannt wird, und wenn **bIsActive nicht** TRUE ist, wird der Befehl ausgeführt. Die Kombination aus Kanalnummer und Ausgangskontakt wird in der globalen Variablen **g_bHsc_TargetValueMatch_Channel1_Y7_YellowLamp_On** überprüft. Wenn der schnelle Zähler an Kanal 1 den Sollwert **diTargetValue1** erreicht, wird der Ausgang **Y7** auf FALSE gesetzt.

KOP



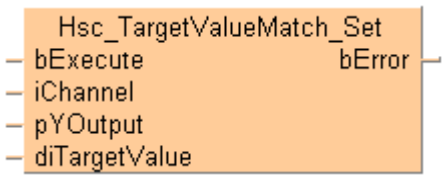
ST

```
bIsActive:=HscInfo_IsActive(iChannel1);

Hsc_TargetValueMatch_Reset(bExecute := DF(bReset) AND NOT bIsActive,
                iChannel := iChannel1,
                pYOutput :=
GetPointer(g_bHsc_TargetValueMatch_Channel1_Y7_YellowLamp_On),
                diTargetValue := diTargetValue1,
                bError => bErrorReset);
```

Hsc_TargetValueMatch_Set

Zählervergleichsausgang setzen (schneller Zähler)

Erklärung Wenn der Istwert den Sollwert **diTargetValue** des durch **iChannel** festgelegten schnellen Zählerkanals erreicht, wird der durch **pYOutput** festgelegte Ausgang sofort auf TRUE gesetzt.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die schnellen Zähler. Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F166_HighSpeedCounter_Set (s. S. 878)

Um die Kombination von Kanal und Y-Ausgang zu prüfen, benötigt der Compiler das folgende Benennungsmuster für globale Variablen:
`%s`Hsc_TargetValueMatch_`Channel%d`_`Y%d`_`%s`
Verwenden Sie für globale Variablen in Zählervergleichsfunktionen grundsätzlich dieses Muster:

- `Channel%d` muss ein schneller Zählerkanal sein, der in den Systemregistern aktiviert ist.
- `Y%d` muss eine explizite, von der SPS unterstützte Ausgangsadresse sein.

FP-Σ, FP0, FP-e:	Y0–Y7
FP-Σ (V3.1 oder neuer), FP0R:	Y0–Y1F
FP-X:	Y0–Y29F

- `%s` ist ein optionaler Deskriptor am Musteranfang
- `%s` ist ein optionaler Deskriptor am Musterende

Optional	Vorgegebenes Muster	Optional
`g_b`	Hsc_TargetValueMatch_`ChannelA`_`Y11F`	`_MotorOn`

Diese globale Variable erzeugt den Code für Kanal **A** und Ausgang **Y11F**.

SPS Typen s. S. 1193

Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Eine steigende Flanke aktiviert die Funktion; Auswertung des Kontrollmerkers "Schneller Zähler" (s. S. 860) mit HscInfo_IsActive (s. S. 1077).
iChannel	INT	FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
pYOutput	POINTER	Durch GetPointer erhaltenes Zeigerergebnis von einer globalen Variable, die Kanalnummer und Ausgang liefert.
diTargetValue	DINT	Geben Sie für den Sollwert einen 32-Bit-Datenwert innerhalb des folgenden Bereichs ein: FP0, FP-e: -838808–+8388607 FPΣ, FP-X, FP0R: -2147483467–+2147483648
Ausgangsvariable	**Datentyp**	**Beschreibung**

bError	BOOL	TRUE, wenn die Kombination von **Channel%d** und **pYOuput.iOffset** nicht mit einer gültigen Kombination von Kanalnummer und Ausgang übereinstimmt, wie sie von der globalen Variablen vorgegeben wird.

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Globale Variablen*

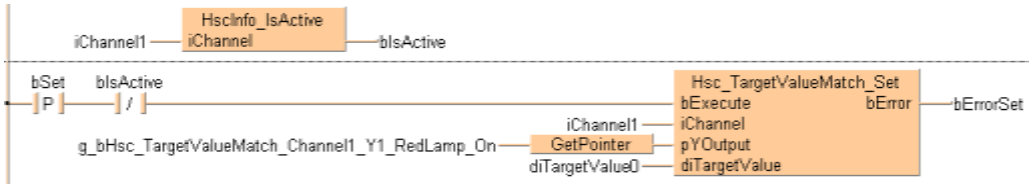
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	g_bHsc_TargetValueMatch_Channel1_Y1_RedLamp_On	Y1	%QX0.1	BOOL	FALSE
1	VAR_GLOBAL	g_bHsc_TargetValueMatch_Channel3_Y9_GreenLamp_On			BOOL	FALSE

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR_EXTERNAL	g_bHsc_TargetValueMatch_Channel1_Y1_RedLamp_On	BOOL	FALSE
1	VAR_EXTERNAL	g_bHsc_TargetValueMatch_Channel3_Y9_GreenLamp_On	BOOL	FALSE
2	VAR	bExecute	BOOL	0
3	VAR	iChannel1	INT	1
4	VAR	diTargetValue0	DINT	15000
5	VAR	bIsActive	BOOL	FALSE
6	VAR	diElapsedValue	DINT	0
7	VAR	bSet	BOOL	FALSE
8	VAR	bErrorSet	BOOL	FALSE

Rumpf Verwenden Sie HscInfo_IsActive (s. S. 1077), um den durch **iChannel1** festgelegten Kanal auszuwerten. Wenn eine steigende Flanke bei **bSet** erkannt wird, und wenn **bIsActive nicht** TRUE ist, wird der Befehl ausgeführt. Die Kombination aus Kanalnummer und Ausgangskontakt wird in der globalen Variablen **g_bHsc_TargetValueMatch_Channel1_Y1_RedLamp_On** überprüft. Wenn der schnelle Zähler an Kanal 1 den Sollwert **diTargetValue0** erreicht, wird der Ausgang **Y1** auf TRUE gesetzt.

KOP



ST

```
bIsActive:=HscInfo_IsActive(iChannel1);

Hsc_TargetValueMatch_Set(bExecute := DF(bSet) AND NOT bIsActive,
                iChannel := iChannel1,
                pYOutput :=
GetPointer(g_bHsc_TargetValueMatch_Channel1_Y1_RedLamp_On),
                diTargetValue := diTargetValue0,
                bError => bErrorSet);
```

Kapitel 38

Tool-Befehle für die Pulsausgabe

38.1 Pulsausgabe-Funktionsbausteine

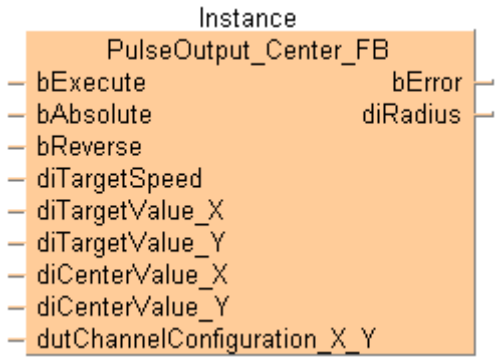
In diesem Abschnitt:

PulseOutput_Center_FB

Kreisinterpolation (Mittelpunktverfahren)

Erklärung

Durch eine zweikanalige Pulsausgabe wird eine bogenförmige Verfahrstrecke erzeugt. Die Parameter für die Pulsausgabe werden in einem Funktionsbaustein und einem SDT festgelegt. Der Radius des Kreisbogens wird aus dem angegebenen Mittelpunkt und der Endposition errechnet. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F176_PulseOutput_Center (s. S. 937)

Prüfen Sie mit PulseInfo_IsActive (s. S. 1134), ob der Kontrollmerker für den gewählten Kanal FALSE ist.

SPS Typen

s. S. 1195

Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Aktivierung des Funktionsbausteins (mit dauerhaftem Trigger)
bAbsolute		Absolutwert-Steuerung = TRUE, Relativwertpositionierung = FALSE
bCounterclockwise		Drehrichtung: Rückwärts = TRUE, Vorwärts = FALSE
diTargetSpeed	DINT	Sollgeschwindigkeit: Resultierende Geschwindigkeit an beiden Achsen = 100–20000 (100Hz–20kHz)
diTargetValue_X		Sollwert [Pulse]: -8388608–8388607
diTargetValue_Y		
diCenterValue_X		
diCenterValue_Y		
dutChannelConfiguration_X_Y	Vordefinierter System-SDT für die Kanalkonfiguration: PulseOutput_Channel_Configuration_DUT Kanal: 0, 2	
Ausgangsvariable	**Datentyp**	**Beschreibung**
bError	BOOL	TRUE, wenn ein zugewiesener Eingangswert falsch ist. Der Funktionsbaustein wird nicht weiter ausgeführt.
diRadius	DINT	Radius [Pulse]

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP)strukturierten Text (ST) programmiert. Ein Beispiel für den strukturierten Text (ST) finden Sie in der Online-Hilfe. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

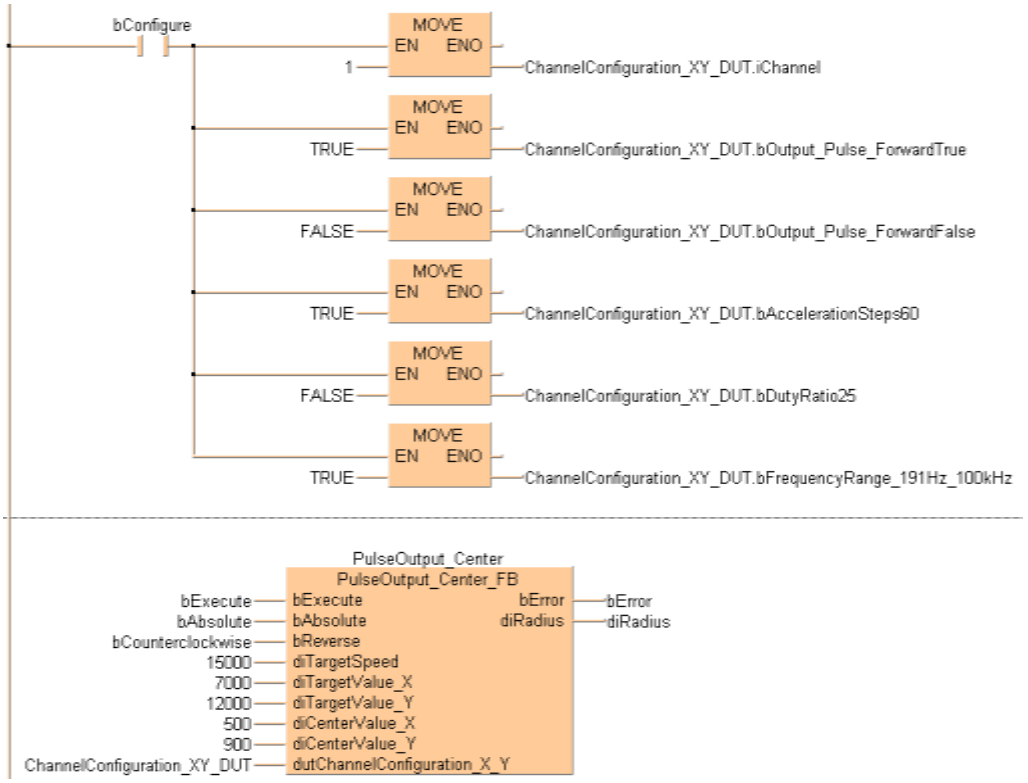
SDT Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: PulseOutput_Channel_Configuration_DUT

	Bezeichner	Typ	Initial	Kommentar
0	iChannel	INT	0	FP-SIGMA: 0, 2
1	bOutput_Pulse_ForwardTrue	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
2	bOutput_Pulse_ForwardFalse	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
3	bAccelerationSteps60	BOOL	TRUE	FP-SIGMA, FP-X: Number of acceleration/deceleration steps: 60 (else 30 steps)
4	bDutyRatio25	BOOL	TRUE	FP-SIGMA, FP-X: Duty ratio (for pulse duration and period): 25% (else 50%)
5	bFrequencyRange_48Hz_100kHz	BOOL	FALSE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 48Hz–100kHz
6	bFrequencyRange_191Hz_100kHz	BOOL	TRUE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 191Hz–100kHz
7	bPulseWidth80µs	BOOL	FALSE	FP0, FP-e Home, Trapezoidal: 80µs (else 50%)
8	iDutyRatioIn10PercentSteps	INT	0	FP0, FP-e Jog: 1-9: duty ratio in increments of 10%, 0: fixed pulse width of 80µs
9	bEnableHomeOnlyAfterNearHomeDeceleration	BOOL	FALSE	FP0R: Type 1 (else type 0), FP-SIGMA, FP-X: Type 2 (else type 1)
10	iHomeInputDeviationCounterClearSignalOutputTime_in_0p5ms	INT	0	FP0R, FP-SIGMA, FP-X: 0 to 200 [x0.5ms]
11	bCalculationOnly	BOOL	FALSE	FP0R: Jog, Trapezoidal: Output operation calculation only (else pulse output)
12	bTrapezoidalMaximumTargetSpeed50kHz	BOOL	FALSE	FP0R: Output operation: Type 1: The target speed can be up to the maximum s...
13	bExecuteInInterrupt	BOOL	FALSE	FP0R Jog positioning, trapezoidal: Execute in or called from interrupt program (el...
14	bJogWithNoCounting	BOOL	FALSE	Only pulse outputs without counting, no target value match. FP-SIGMA, FP-X: b...
15	bContinueAfterDone	BOOL	FALSE	FP-SIGMA circular pulse output: 0=Execution stops when target value has been ...

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	PulseOutput_Center	PulseOutput_Center_FB	
1	VAR	bExecute	BOOL	FALSE
2	VAR	bAbsolute	BOOL	FALSE
3	VAR	bContinueAfterDone	BOOL	FALSE
4	VAR	bCounterclockwise	BOOL	FALSE
5	VAR	ChannelConfiguration_X_DUT	PulseOutput_Channel_Configuration_DUT	
6	VAR	bError	BOOL	FALSE
7	VAR	diRadius	DINT	0
8	VAR	bConfigure	BOOL	FALSE

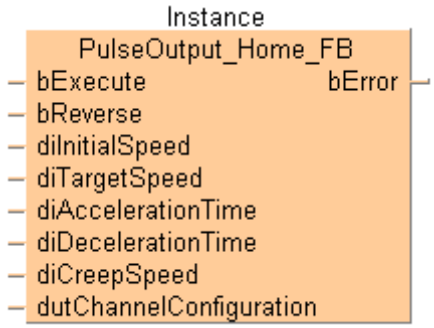
KOP



PulseOutput_Home_FB

Referenzpunktfahrt

Erklärung Anhand der Parameter im Funktionsbaustein und dem angegebenen strukturierten Datentyp wird eine Referenzpunktfahrt durchgeführt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe:

- FPΣ, FP-X: F171_PulseOutput_Home (s. S. 910)
- FP0R: F177_PulseOutput_Home (s. S. 945)
- FP-e, FP0: F168_PulseOutput_Home (s. S. 894)

Prüfen Sie mit PulseInfo_IsActive (s. S. 1134), ob der Kontrollmerker für den gewählten Kanal FALSE ist.

Mit PulseInfo_IsHomeInputTrue (s. S. 1138) prüfen Sie, ob der Referenzpunkteingang TRUE ist.

☞ **Vermeiden Sie Fehlfunktionen und Betriebsfehler wie folgt:**

- Wenn Sie einen Referenzpunkteingang verwenden, stellen Sie sicher, dass das Systemregister auf Pulsausgabemodus gesetzt ist.
- Der Referenzpunkteingang darf nicht von anderen Befehlen, wie Pulsspeicherfunktion, Interrupt-Eingang oder schneller Zähler, belegt sein.

SPS Typen s. S. 1195

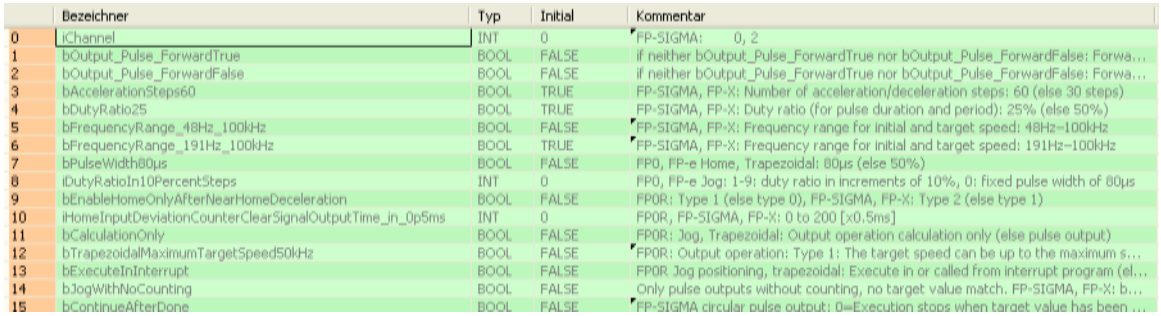
Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Eine steigende Flanke aktiviert den Funktionsbaustein
bReverse		Bewegungsrichtung: Vorwärts = FALSE, Rückwärts = TRUE
diInitialSpeed	DINT	Anfangsgeschwindigkeit/Sollgeschwindigkeit: Setzen Sie diesen Wert gemäß dem in PulseOutput_Channel_Configuration_DUT festgelegten Frequenzbereich: FPΣ, FP-X: 1 bis 9800 (1,5Hz–9,8kHz) 48 bis 100000 (48Hz–100kHz) 191 bis 100000 (191–100kHz) FP0R: 1 bis 50000 (1Hz–50kHz) FP0, FP-e: 40 bis 5000 (40Hz–5kHz)
diTargetSpeed		
diAccelerationTime		Beschleunigungs-/Bremszeit (FPΣ, FP-X): Mit 30 Schritten: 30ms–32760ms (Wert als Vielfaches von 30 angeben) Mit 60 Schritten: 60ms–32760ms (Wert als Vielfaches von 60 angeben) Beschleunigungs-/Bremszeit (FP0, FP-e): 30ms–32760ms Beschleunigungszeit (FP0R): 1ms–32760ms
diDecelerationTime		Bremszeit (FP0R): 1ms–32760ms
diCreepSpeed		Suchgeschwindigkeit (FP0R): 1 bis 50000 (1Hz–50kHz)

dutChannelConfiguration	Vordefinierter System-SDT für die Kanalkonfiguration: PulseOutput_Channel_Configuration_DUT	
Ausgangsvariable	**Datentyp**	**Beschreibung**
bError	BOOL	TRUE, wenn ein zugewiesener Eingangswert falsch ist. Der Funktionsbaustein wird nicht weiter ausgeführt.

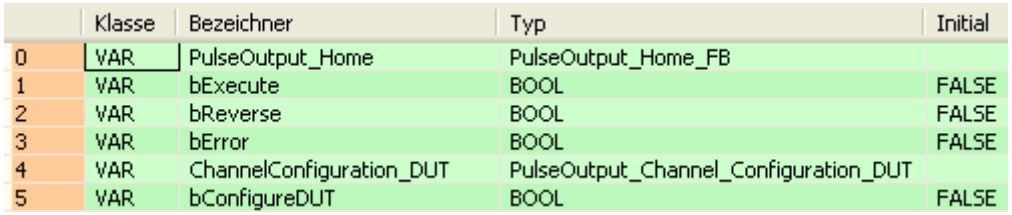
Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP)strukturierten Text (ST) programmiert. Ein Beispiel für den strukturierten Text (ST) finden Sie in der Online-Hilfe. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: PulseOutput_Channel_Configuration_DUT



	Bezeichner	Typ	Initial	Kommentar
0	iChannel	INT	0	FP-SIGMA: 0, 2
1	bOutput_Pulse_ForwardTrue	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
2	bOutput_Pulse_ForwardFalse	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
3	bAccelerationSteps60	BOOL	TRUE	FP-SIGMA, FP-X: Number of acceleration/deceleration steps: 60 (else 30 steps)
4	bDutyRatio25	BOOL	TRUE	FP-SIGMA, FP-X: Duty ratio (for pulse duration and period): 25% (else 50%)
5	bFrequencyRange_48Hz_100kHz	BOOL	FALSE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 48Hz–100kHz
6	bFrequencyRange_191Hz_100kHz	BOOL	TRUE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 191Hz–100kHz
7	bPulseWidth80µs	BOOL	FALSE	FP0, FP-e Home, Trapezoidal: 80µs (else 50%)
8	iDutyRatioIn10PercentSteps	INT	0	FP0, FP-e Jog: 1-9: duty ratio in increments of 10%, 0: fixed pulse width of 80µs
9	bEnableHomeOnlyAfterNearHomeDeceleration	BOOL	FALSE	FP0R: Type 1 (else type 0), FP-SIGMA, FP-X: Type 2 (else type 1)
10	iHomeInputDeviationCounterClearSignalOutputTime_in_0p5ms	INT	0	FP0R, FP-SIGMA, FP-X: 0 to 200 [x0.5ms]
11	bCalculationOnly	BOOL	FALSE	FP0R: Jog, Trapezoidal: Output operation calculation only (else pulse output)
12	bTrapezoidalMaximumTargetSpeed50kHz	BOOL	FALSE	FP0R: Output operation: Type 1: The target speed can be up to the maximum s...
13	bExecuteInInterrupt	BOOL	FALSE	FP0R Jog positioning, trapezoidal: Execute in or called from interrupt program (el...
14	bJogWithNoCounting	BOOL	FALSE	Only pulse outputs without counting, no target value match. FP-SIGMA, FP-X: b...
15	bContinueAfterDone	BOOL	FALSE	FP-SIGMA circular pulse output: 0=Execution stops when target value has been ...

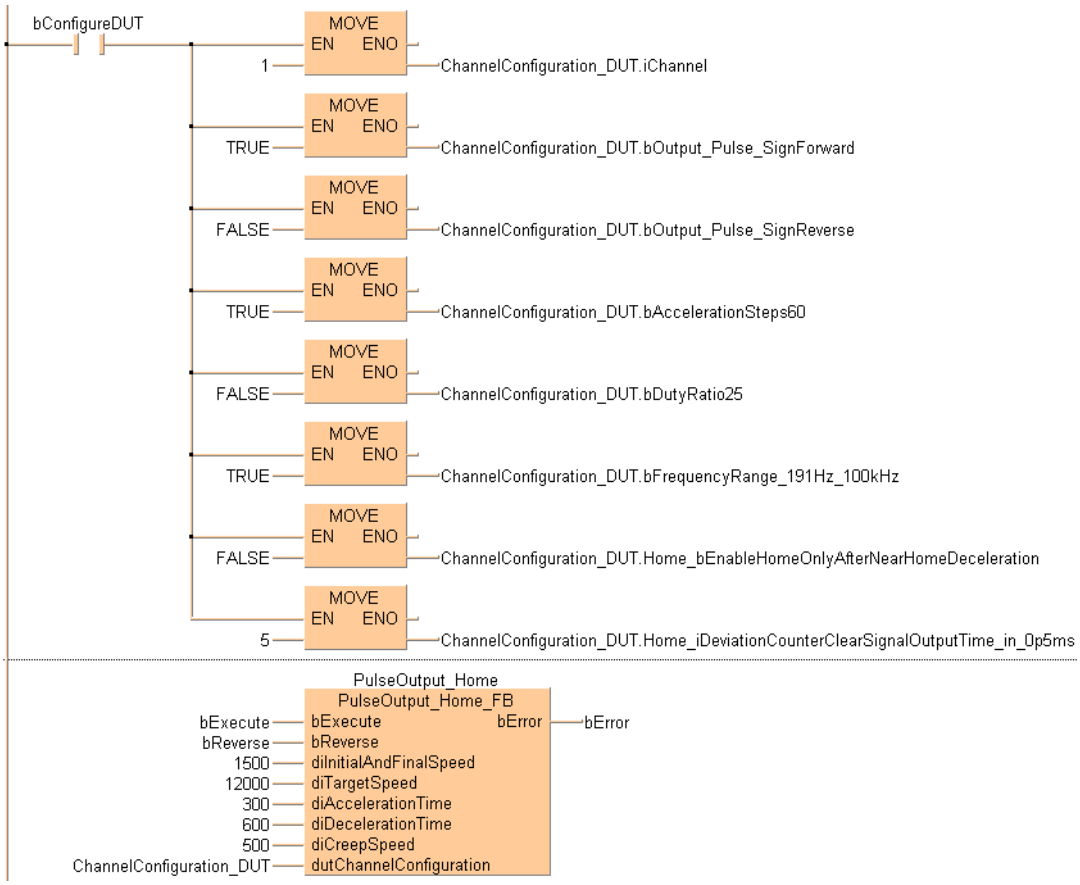
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



	Klasse	Bezeichner	Typ	Initial
0	VAR	PulseOutput_Home	PulseOutput_Home_FB	
1	VAR	bExecute	BOOL	FALSE
2	VAR	bReverse	BOOL	FALSE
3	VAR	bError	BOOL	FALSE
4	VAR	ChannelConfiguration_DUT	PulseOutput_Channel_Configuration_DUT	
5	VAR	bConfigureDUT	BOOL	FALSE

Rumpf

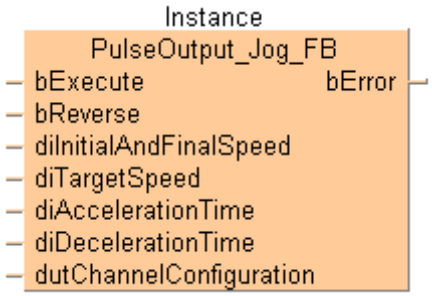
KOP


bConfigureDUT
MOVE
EN ENO
1
ChannelConfiguration_DUT.iChannel
TRUE
ChannelConfiguration_DUT.bOutput_Pulse_SignForward
FALSE
ChannelConfiguration_DUT.bOutput_Pulse_SignReverse
TRUE
ChannelConfiguration_DUT.bAccelerationSteps60
FALSE
ChannelConfiguration_DUT.bDutyRatio25
TRUE
ChannelConfiguration_DUT.bFrequencyRange_191Hz_100kHz
FALSE
ChannelConfiguration_DUT.Home_bEnableHomeOnlyAfterNearHomeDeceleration
5
ChannelConfiguration_DUT.Home_iDeviationCounterClearSignalOutputTime_in_0p5ms
PulseOutput_Home
PulseOutput_Home_FB
bExecute
bReverse
1500
12000
300
600
500
ChannelConfiguration_DUT
bExecute
bReverse
diInitialAndFinalSpeed
diTargetSpeed
diAccelerationTime
diDecelerationTime
diCreepSpeed
dutChannelConfiguration
bError
bError

PulseOutput_Jog_FB

Tipp-Betrieb

Erklärung Dieser Befehl wird für den Tipp-Betrieb verwendet. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F172_PulseOutput_Jog (s. S. 919). Prüfen Sie mit PulseInfo_IsActive (s. S. 1134), ob der Kontrollmerker für den gewählten Kanal FALSE ist.

SPS Typen s. S. 1195

Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Mit Flanke oder dauerhaft, wenn eine Änderung der Geschwindigkeit erforderlich ist
bReverse		Bewegungsrichtung: Vorwärts = FALSE, Rückwärts = TRUE
diInitialAndFinalSpeed	DINT	Anfangs- und Endgeschwindigkeit (FP0R): 1 bis 50000 (1Hz–50kHz)
diTargetSpeed		Sollgeschwindigkeit: Setzen Sie diesen Wert gemäß dem in PulseOutput_Channel_Configuration_DUT festgelegten Frequenzbereich: FPΣ, FP-X: 1 bis 9800 (1,5Hz–9,8kHz) 48 bis 100000 (48Hz–100kHz) 191 bis 100000 (191–100kHz) FP0R: 1 bis 50000 (1Hz–50kHz) FP0, FP-e: 40 bis 5000 (40Hz–5kHz)
diAccelerationTime		Beschleunigungszeit (FP0R): 1ms–32760ms (bis zur maximalen Geschwindigkeit)
diDecelerationTime		Bremszeit (FP0R): 1ms–32760ms (ab der maximalen Geschwindigkeit)
dutChannelConfiguration	Vordefinierter System-SDT für die Kanalkonfiguration: PulseOutput_Channel_Configuration_DUT	
Ausgangsvariable	**Datentyp**	**Beschreibung**
bError	BOOL	TRUE, wenn ein zugewiesener Eingangswert falsch ist. Der Funktionsbaustein wird nicht weiter ausgeführt.

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP)strukturierten Text (ST) programmiert. Ein Beispiel für den strukturierten Text (ST) finden Sie in der Online-Hilfe. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: PulseOutput_Channel_Configuration_DUT

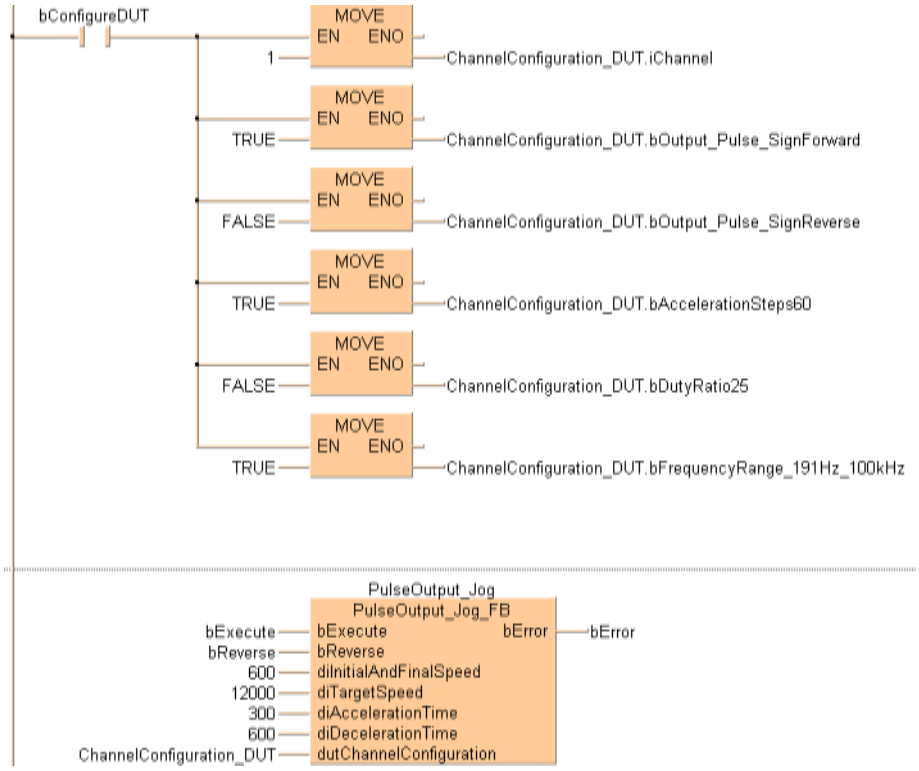
	Bezeichner	Typ	Initial	Kommentar
0	iChannel	INT	0	FP-SIGMA: 0, 2
1	bOutput_Pulse_ForwardTrue	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
2	bOutput_Pulse_ForwardFalse	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
3	bAccelerationSteps60	BOOL	TRUE	FP-SIGMA, FP-X: Number of acceleration/deceleration steps: 60 (else 30 steps)
4	bDutyRatio25	BOOL	TRUE	FP-SIGMA, FP-X: Duty ratio (for pulse duration and period): 25% (else 50%)
5	bFrequencyRange_48Hz_100kHz	BOOL	FALSE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 48Hz–100kHz
6	bFrequencyRange_191Hz_100kHz	BOOL	TRUE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 191Hz–100kHz
7	bPulseWidth80µs	BOOL	FALSE	FP0, FP-e Home, Trapezoidal: 80µs (else 50%)
8	iDutyRatioIn10PercentSteps	INT	0	FP0, FP-e Jog: 1-9: duty ratio in increments of 10%, 0: fixed pulse width of 80µs
9	bEnableHomeOnlyAfterNearHomeDeceleration	BOOL	FALSE	FP0R: Type 1 (else type 0), FP-SIGMA, FP-X: Type 2 (else type 1)
10	iHomeInputDeviationCounterClearSignalOutputTime_in_0p5ms	INT	0	FP0R, FP-SIGMA, FP-X: 0 to 200 [x0.5ms]
11	bCalculationOnly	BOOL	FALSE	FP0R: Jog, Trapezoidal: Output operation calculation only (else pulse output)
12	bTrapezoidalMaximumTargetSpeed50kHz	BOOL	FALSE	FP0R: Output operation: Type 1: The target speed can be up to the maximum s...
13	bExecuteInInterrupt	BOOL	FALSE	FP0R Jog positioning, trapezoidal: Execute in or called from interrupt program (el...
14	bJogWithNoCounting	BOOL	FALSE	Only pulse outputs without counting, no target value match. FP-SIGMA, FP-X: b...
15	bContinueAfterDone	BOOL	FALSE	FP-SIGMA circular pulse output: 0=Execution stops when target value has been ...

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	PulseOutput_Jog	PulseOutput_Jog_FB	
1	VAR	bExecute	BOOL	FALSE
2	VAR	bReverse	BOOL	FALSE
3	VAR	ChannelConfiguration_DUT	PulseOutput_Channel_Configuration_DUT	
4	VAR	bError	BOOL	FALSE
5	VAR	bConfigureDUT	BOOL	FALSE

Rumpf

KOP



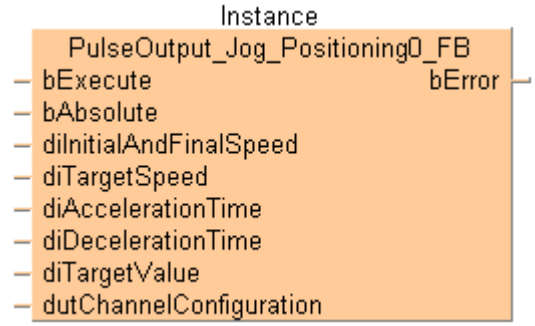
PulseOutput_Jog_Positioning0_FB

Tipp-Betrieb und Positionierung

Erklärung

Dieser Befehl wird für den Tipp-Betrieb verwendet. Wenn der Positionierungstrigger-Eingang auf TRUE gesetzt wird, wird die zuvor festgelegte Zahl von Pulsen ausgegeben. Ehe der Sollwert erreicht ist und die Pulsausgabe endet, wird eine Abbremsung ausgeführt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.

Die Geschwindigkeit lässt sich nur innerhalb des für die Sollgeschwindigkeit festgelegten Bereichs ändern.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F171_PulseOutput_Jog_Positioning (s. S. 914). Prüfen Sie mit PulseInfo_IsActive (s. S. 1134), ob der Kontrollmerker für den gewählten Kanal FALSE ist. Mit PulseControl_PulseOutputStop (s. S. 1125) beenden Sie die Pulsausgabe am angegebenen Kanal. Mit PulseControl_DeceleratedStop (s. S. 1116) führen Sie einen gebremsten Halt durch.

SPS Typen s. S. 1195

Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Mit Flanke oder dauerhaft, wenn eine Änderung der Geschwindigkeit erforderlich ist
bAbsolute	BOOL:=FALSE	Nur Relativwertpositionierung möglich; muss immer FALSE sein, andernfalls wird ein Fehler ausgegeben.
diInitialAndFinalSpeed	DINT	Anfangs- und Endgeschwindigkeit (FP0R): 1 bis 50000 (1Hz–50kHz)
diTargetSpeed		Sollgeschwindigkeit: Setzen Sie diesen Wert gemäß dem in PulseOutput_Channel_Configuration_DUT festgelegten Frequenzbereich: FPΣ, FP-X: 1 bis 9800 (1,5Hz–9,8kHz) 48 bis 100000 (48Hz–100kHz) 191 bis 100000 (191–100kHz) FP0R: 1 bis 50000 (1Hz–50kHz) FP0, FP-e: 40 bis 5000 (40Hz–5kHz)
diAccelerationTime		Beschleunigungszeit (FP0R): 1ms–32760ms (bis zur maximalen Geschwindigkeit)
diDecelerationTime		Bremszeit (FP0R): 1ms–32760ms (ab der maximalen Geschwindigkeit)
diTargetValue		Sollwert [Pulse]: -2147483648–2147483647
dutChannelConfiguration	Vordefinierter System-SDT für die Kanalkonfiguration: PulseOutput_Channel_Configuration_DUT	
Ausgangsvariable	**Datentyp**	**Beschreibung**

bError	BOOL	TRUE, wenn ein zugewiesener Eingangswert falsch ist. Der Funktionsbaustein wird nicht weiter ausgeführt. TRUE, wenn der zugewiesene Kanal nicht in den Systemregistern aktiviert ist, oder wenn **bAbsolute** TRUE ist

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP)strukturierten Text (ST) programmiert. Ein Beispiel für den strukturierten Text (ST) finden Sie in der Online-Hilfe. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

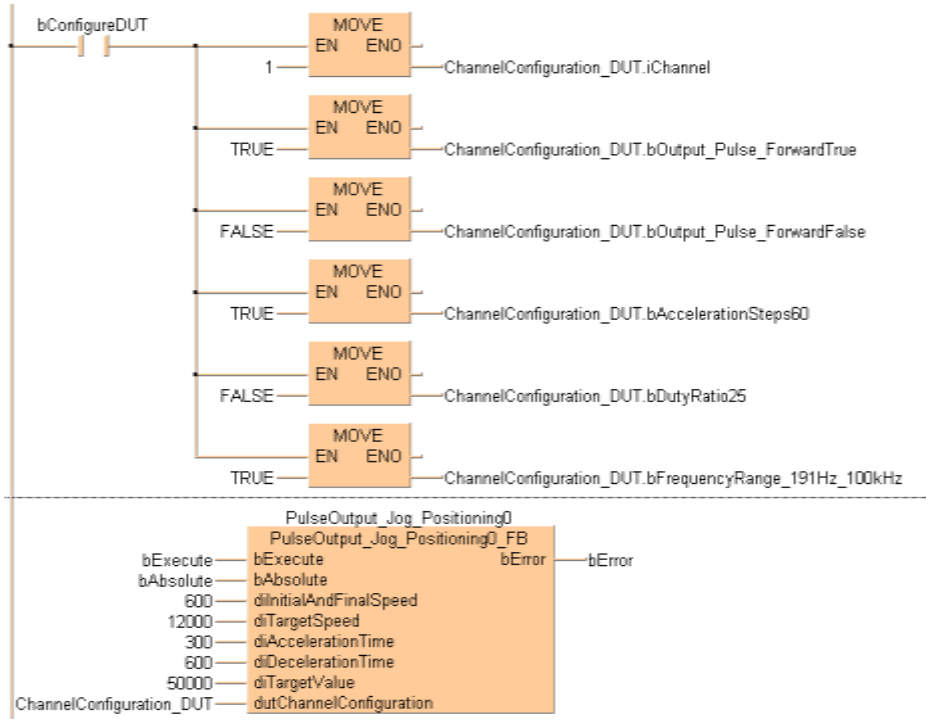
SDT Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: PulseOutput_Channel_Configuration_DUT

	Bezeichner	Typ	Initial	Kommentar
0	iChannel	INT	0	FP-SIGMA: 0, 2
1	bOutput_Pulse_ForwardTrue	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
2	bOutput_Pulse_ForwardFalse	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
3	bAccelerationSteps60	BOOL	TRUE	FP-SIGMA, FP-X: Number of acceleration/deceleration steps: 60 (else 30 steps)
4	bDutyRatio25	BOOL	TRUE	FP-SIGMA, FP-X: Duty ratio (for pulse duration and period): 25% (else 50%)
5	bFrequencyRange_48Hz_100kHz	BOOL	FALSE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 48Hz–100kHz
6	bFrequencyRange_191Hz_100kHz	BOOL	TRUE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 191Hz–100kHz
7	bPulseWidth80µs	BOOL	FALSE	FP0, FP-e Home, Trapezoidal: 80µs (else 50%)
8	iDutyRatioIn10PercentSteps	INT	0	FP0, FP-e Jog: 1-9: duty ratio in increments of 10%, 0: fixed pulse width of 80µs
9	bEnableHomeOnlyAfterNearHomeDeceleration	BOOL	FALSE	FP0R: Type 1 (else type 0), FP-SIGMA, FP-X: Type 2 (else type 1)
10	iHomeInputDeviationCounterClearSignalOutputTime_in_0p5ms	INT	0	FP0R, FP-SIGMA, FP-X: 0 to 200 [x0.5ms]
11	bCalculationOnly	BOOL	FALSE	FP0R: Jog, Trapezoidal: Output operation calculation only (else pulse output)
12	bTrapezoidalMaximumTargetSpeed50kHz	BOOL	FALSE	FP0R: Output operation: Type 1: The target speed can be up to the maximum s...
13	bExecuteInInterrupt	BOOL	FALSE	FP0R Jog positioning, trapezoidal: Execute in or called from interrupt program (el...
14	bJogWithNoCounting	BOOL	FALSE	Only pulse outputs without counting, no target value match. FP-SIGMA, FP-X: b...
15	bContinueAfterDone	BOOL	FALSE	FP-SIGMA circular pulse output: 0=Execution stops when target value has been ...

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	PulseOutput_Jog_Positioning0	PulseOutput_Jog_Positioning0_FB	
1	VAR	bExecute	BOOL	FALSE
2	VAR	ChannelConfiguration_DUT	PulseOutput_Channel_Configuration_DUT	
3	VAR	bError	BOOL	FALSE
4	VAR	bConfigureDUT	BOOL	FALSE
5	VAR	bAbsolute	BOOL	FALSE

KOP



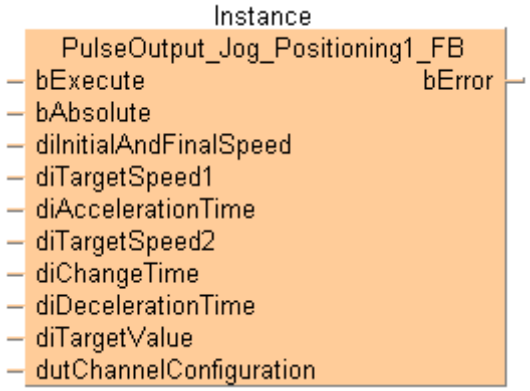
PulseOutput_Jog_Positioning1_FB

Tipp-Betrieb und Positionierung

Erklärung

Dieser Befehl wird für den Tipp-Betrieb verwendet. Wenn der Positionierungstrigger-Eingang auf TRUE gesetzt wird, wird die zuvor festgelegte Zahl von Pulsen ausgegeben. Ehe der Sollwert erreicht ist und die Pulsausgabe endet, wird eine Abbremsung ausgeführt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.

Die Sollgeschwindigkeit lässt sich einmal ändern, wenn der Positionierungstrigger-Eingang auf TRUE gesetzt wird.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F171_PulseOutput_Jog_Positioning (s. S. 914)

Prüfen Sie mit PulseInfo_IsActive (s. S. 1134), ob der Kontrollmerker für den gewählten Kanal FALSE ist.

Mit PulseControl_PulseOutputStop (s. S. 1125) beenden Sie die Pulsausgabe am angegebenen Kanal. Mit PulseControl_DeceleratedStop (s. S. 1116) führen Sie einen gebremsten Halt durch.

SPS Typen

s. S. 1195

Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Mit Flanke oder dauerhaft, wenn eine Änderung der Geschwindigkeit erforderlich ist
bAbsolute	BOOL:=FALSE	Nur Relativwertpositionierung möglich; muss immer FALSE sein, andernfalls wird ein Fehler ausgegeben.
diInitialAndFinalSpeed	DINT	Anfangs- und Endgeschwindigkeit = 1–50000 (1Hz–50kHz)
diTargetSpeed1		Sollgeschwindigkeit = 1–50000 (1Hz–50kHz)
diAccelerationTime		Beschleunigungszeit = 1ms–32760ms
diTargetSpeed2		Sollgeschwindigkeit = 1–50000 (1Hz–50kHz)
diChangeTime		Wechselzeit = 1–32760ms
diDecelerationTime		Bremszeit = 1–32760ms
diTargetValue		Sollwert [Pulse]: -2147483648–2147483647
dutChannelConfiguration	Vordefinierter System-SDT für die Kanalkonfiguration: PulseOutput_Channel_Configuration_DUT	

Ausgangsvariable	Datentyp	Beschreibung

bError	BOOL	TRUE, wenn ein zugewiesener Eingangswert falsch ist. Der Funktionsbaustein wird nicht weiter ausgeführt. TRUE, wenn der zugewiesene Kanal nicht in den Systemregistern aktiviert ist, oder wenn **bAbsolute** TRUE ist

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP)strukturierten Text (ST) programmiert. Ein Beispiel für den strukturierten Text (ST) finden Sie in der Online-Hilfe. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

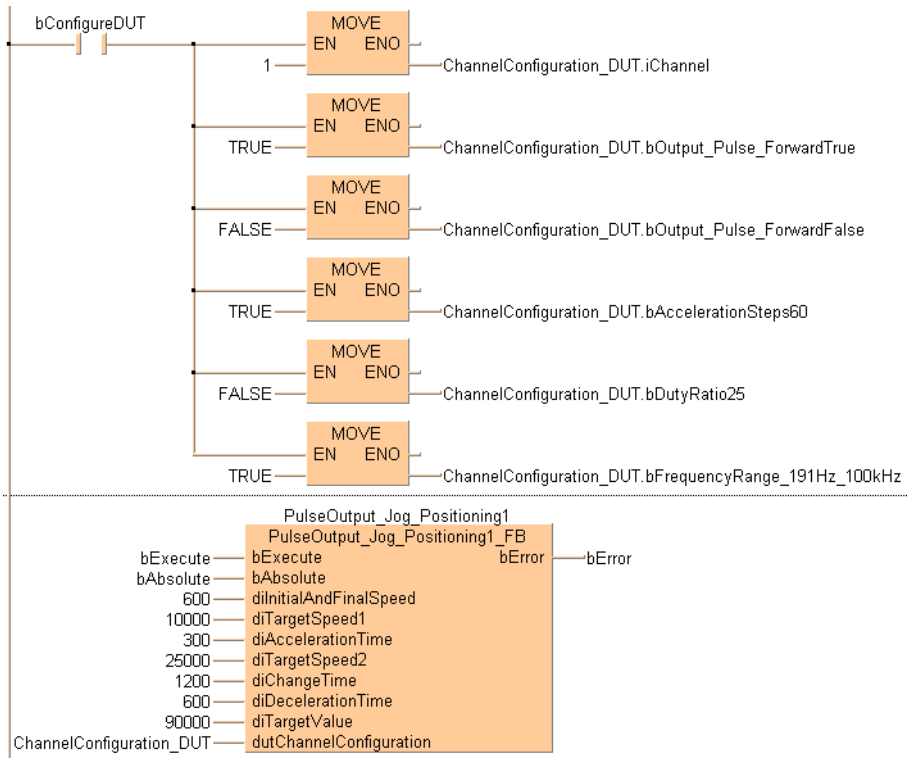
SDT Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: PulseOutput_Channel_Configuration_DUT

	Bezeichner	Typ	Initial	Kommentar
0	iChannel	INT	0	FP-SIGMA: 0, 2
1	bOutput_Pulse_ForwardTrue	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
2	bOutput_Pulse_ForwardFalse	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
3	bAccelerationSteps60	BOOL	TRUE	FP-SIGMA, FP-X: Number of acceleration/deceleration steps: 60 (else 30 steps)
4	bDutyRatio25	BOOL	TRUE	FP-SIGMA, FP-X: Duty ratio (for pulse duration and period): 25% (else 50%)
5	bFrequencyRange_48Hz_100kHz	BOOL	FALSE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 48Hz–100kHz
6	bFrequencyRange_191Hz_100kHz	BOOL	TRUE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 191Hz–100kHz
7	bPulseWidth80µs	BOOL	FALSE	FP0, FP-e Home, Trapezoidal: 80µs (else 50%)
8	iDutyRatioIn10PercentSteps	INT	0	FP0, FP-e Jog: 1-9: duty ratio in increments of 10%, 0: fixed pulse width of 80µs
9	bEnableHomeOnlyAfterNearHomeDeceleration	BOOL	FALSE	FP0R: Type 1 (else type 0), FP-SIGMA, FP-X: Type 2 (else type 1)
10	iHomeInputDeviationCounterClearSignalOutputTime_in_0p5ms	INT	0	FP0R, FP-SIGMA, FP-X: 0 to 200 [x0.5ms]
11	bCalculationOnly	BOOL	FALSE	FP0R: Jog, Trapezoidal: Output operation calculation only (else pulse output)
12	bTrapezoidalMaximumTargetSpeed50kHz	BOOL	FALSE	FP0R: Output operation: Type 1: The target speed can be up to the maximum s...
13	bExecuteInInterrupt	BOOL	FALSE	FP0R Jog positioning, trapezoidal: Execute in or called from interrupt program (el...
14	bJogWithNoCounting	BOOL	FALSE	Only pulse outputs without counting, no target value match. FP-SIGMA, FP-X: b...
15	bContinueAfterDone	BOOL	FALSE	FP-SIGMA circular pulse output: 0=Execution stops when target value has been ...

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	PulseOutput_Jog_Positioning1	PulseOutput_Jog_Positioning1_FB	
1	VAR	bExecute	BOOL	FALSE
2	VAR	ChannelConfiguration_DUT	PulseOutput_Channel_Configuration_DUT	
3	VAR	bError	BOOL	FALSE
4	VAR	bConfigureDUT	BOOL	FALSE
5	VAR	bAbsolute	BOOL	FALSE

KOP

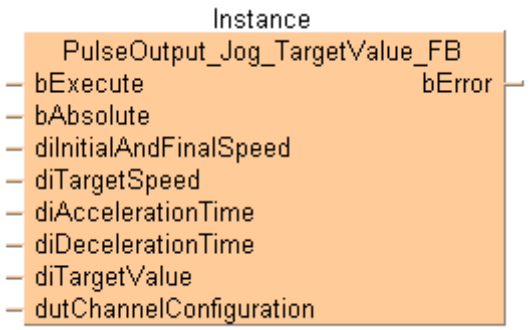


PulseOutput_Jog_TargetValue_FB

Tipp-Betrieb mit Sollwert

Erklärung

Dieser Befehl wird für den Tipp-Betrieb verwendet. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist. Die Pulsausgabe stoppt, wenn der Sollwert erreicht ist.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F172_PulseOutput_Jog (s. S. 919). Prüfen Sie mit PulseInfo_IsActive (s. S. 1134), ob der Kontrollmerker für den gewählten Kanal FALSE ist.

SPS Typen s. S. 1195

Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Mit Flanke oder dauerhaft, wenn eine Änderung der Geschwindigkeit erforderlich ist
bAbsolute		FP0R: Absolutwert-Steuerung = TRUE, Relativwertpositionierung = FALSE
diInitialAndFinalSpeed	DINT	FP0R: Anfangs- und Endgeschwindigkeit = 1–50000 (1Hz–50kHz)
diTargetSpeed		Sollgeschwindigkeit: Setzen Sie diesen Wert gemäß dem in PulseOutput_Channel_Configuration_DUT festgelegten Frequenzbereich: FPΣ, FP-X: 1–9800 (1,5Hz–9,8kHz) 48–100000 (48Hz–100kHz) 191–100000 (191–100kHz) FP0R: 1–50000 (1Hz–50kHz) FP0, FP-e: 40–5000 (40Hz–5kHz)
diAccelerationTime		Beschleunigungszeit (FP0R): 1ms–32760ms (bis zur maximalen Geschwindigkeit)
diDecelerationTime		Bremszeit (FP0R): 1ms–32760ms (ab der maximalen Geschwindigkeit)
diTargetValue		Sollwert [Pulse]: -2147483648–2147483647
dutChannelConfiguration	Vordefinierter System-SDT für die Kanalkonfiguration: PulseOutput_Channel_Configuration_DUT	

Ausgangsvariable	Datentyp	Beschreibung
bError	BOOL	TRUE, wenn ein zugewiesener Eingangswert falsch ist. Der Funktionsbaustein wird nicht weiter ausgeführt. Weitere Fehlerzustände für die FPΣ, FP-X : TRUE, wenn der zugewiesene Kanal nicht in den Systemregistern aktiviert ist, oder wenn **bAbsolute** TRUE ist

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP)strukturierten Text (ST) programmiert. Ein Beispiel für den strukturierten Text (ST) finden Sie in der Online-Hilfe. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: PulseOutput_Channel_Configuration_DUT

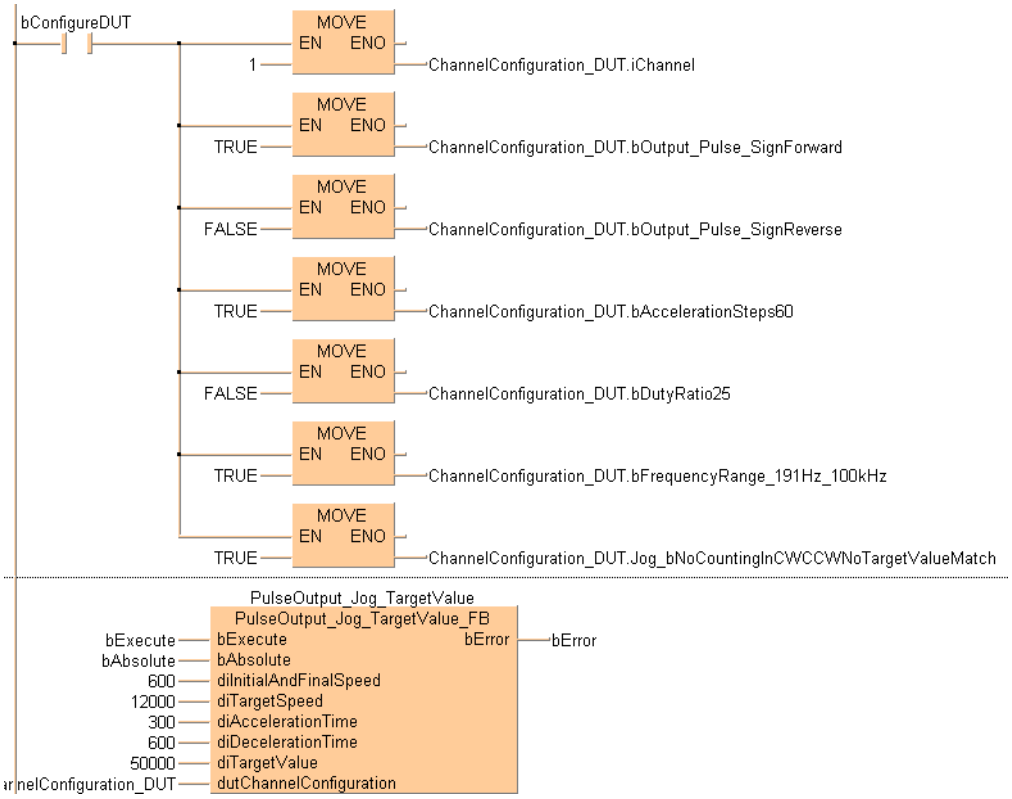
	Bezeichner	Typ	Initial	Kommentar
0	iChannel	INT	0	FP-SIGMA: 0, 2
1	bOutput_Pulse_ForwardTrue	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
2	bOutput_Pulse_ForwardFalse	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
3	bAccelerationSteps60	BOOL	TRUE	FP-SIGMA, FP-X: Number of acceleration/deceleration steps: 60 (else 30 steps)
4	bDutyRatio25	BOOL	TRUE	FP-SIGMA, FP-X: Duty ratio (for pulse duration and period): 25% (else 50%)
5	bFrequencyRange_48Hz_100kHz	BOOL	FALSE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 48Hz–100kHz
6	bFrequencyRange_191Hz_100kHz	BOOL	TRUE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 191Hz–100kHz
7	bPulseWidth80µs	BOOL	FALSE	FP0, FP-e Home, Trapezoidal: 80µs (else 50%)
8	iDutyRatioIn10PercentSteps	INT	0	FP0, FP-e Jog: 1-9: duty ratio in increments of 10%, 0: fixed pulse width of 80µs
9	bEnableHomeOnlyAfterNearHomeDeceleration	BOOL	FALSE	FP0R: Type 1 (else type 0), FP-SIGMA, FP-X: Type 2 (else type 1)
10	iHomeInputDeviationCounterClearSignalOutputTime_in_0p5ms	INT	0	FP0R, FP-SIGMA, FP-X: 0 to 200 [x0.5ms]
11	bCalculationOnly	BOOL	FALSE	FP0R: Jog, Trapezoidal: Output operation calculation only (else pulse output)
12	bTrapezoidalMaximumTargetSpeed50kHz	BOOL	FALSE	FP0R: Output operation: Type 1: The target speed can be up to the maximum s...
13	bExecuteInInterrupt	BOOL	FALSE	FP0R Jog positioning, trapezoidal: Execute in or called from interrupt program (el...
14	bJogWithNoCounting	BOOL	FALSE	Only pulse outputs without counting, no target value match. FP-SIGMA, FP-X: b...
15	bContinueAfterDone	BOOL	FALSE	FP-SIGMA circular pulse output: 0=Execution stops when target value has been ...

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	PulseOutput_Jog_TargetValue	PulseOutput_Jog_TargetValue_FB	
1	VAR	bExecute	BOOL	FALSE
2	VAR	bAbsolute	BOOL	FALSE
3	VAR	ChannelConfiguration_DUT	PulseOutput_Channel_Configuration_DUT	
4	VAR	bError	BOOL	FALSE
5	VAR	bConfigureDUT	BOOL	FALSE

Rumpf

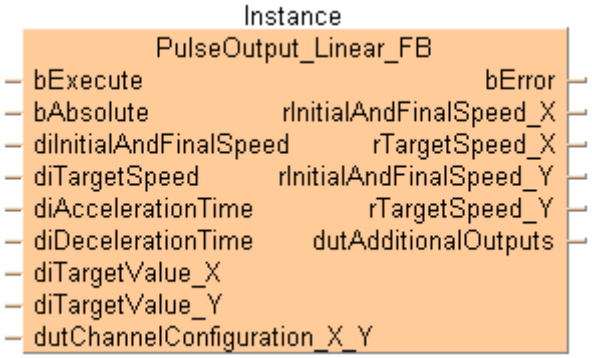
KOP



PulseOutput_Linear_FB

Linearinterpolation

Erklärung Durch eine zweikanalige Pulsausgabe wird eine geradlinige Verfahrstrecke erzeugt. Die Parameter für die Pulsausgabe werden in einem Funktionsbaustein und einem SDT festgelegt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F175_PulseOutput_Linear (s. S. 932)

Prüfen Sie mit PulseInfo_IsActive (s. S. 1134), ob der Kontrollmerker für den gewählten Kanal FALSE ist.

SPS Typen s. S. 1195

Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Mit Flanke oder dauerhaft, wenn eine Änderung der Geschwindigkeit erforderlich ist
bAbsolute		Absolutwert-Steuerung = TRUE, Relatives Positionieren = FALSE
diInitialAndFinalSpeed	DINT	Anfangs- und Endgeschwindigkeit: Resultierende Geschwindigkeit = 1–50000 (1Hz–50kHz)
diTargetSpeed		Sollgeschwindigkeit: Resultierende Geschwindigkeit = 1–50000 (1Hz–50kHz)
diAccelerationTime		Beschleunigungs-/Bremszeit (FPΣ, FP-X): 0ms–32767ms Beschleunigungszeit (FP0R): 0ms–32767ms
diDecelerationTime		Bremszeit (FP0R): 0ms–32767ms
diTargetValue_X		Sollwert (x-Achse) [Pulse] -8388608–8388607
diTargetValue_Y		Sollwert (y-Achse) [Pulse] -8388608–8388607
dutChannelConfiguration_X_Y	Vordefinierter System-SDT für die Kanalkonfiguration: PulseOutput_Channel_Configuration_DUT Für die Interpolation verwenden Sie paarweise Kanal 0 und 1 oder Kanal 2 und 3. Sie können nur 0 oder 2 angeben (für C14T: nur 0).	

Ausgangsvariable	Datentyp	Beschreibung
bError	BOOL	TRUE, wenn ein zugewiesener Eingangswert falsch ist. Der Funktionsbaustein wird nicht weiter ausgeführt. Wird nur gesetzt, wenn die globale Konstante MC_PulseOutput_Library_Basic_bCheckInputs auf TRUE steht.
riInitialAndFinalSpeed_X	REAL	Anfangs- und Restgeschwindigkeit (x-Achse) [Hz]
riTargetSpeed_X		Sollwert (x-Achse) [Hz]
riInitialAndFinalSpeed_Y		Anfangs- und Restgeschwindigkeit (y-Achse) [Hz]

riTargetSpeed_Y		Sollwert (y-Achse) [Hz]
dutAdditionalOutputs	FPΣ, FP-X: PulseOutput_Linear_AdditionalOutputs_DUT	

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP)strukturierten Text (ST) programmiert. Ein Beispiel für den strukturierten Text (ST) finden Sie in der Online-Hilfe. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

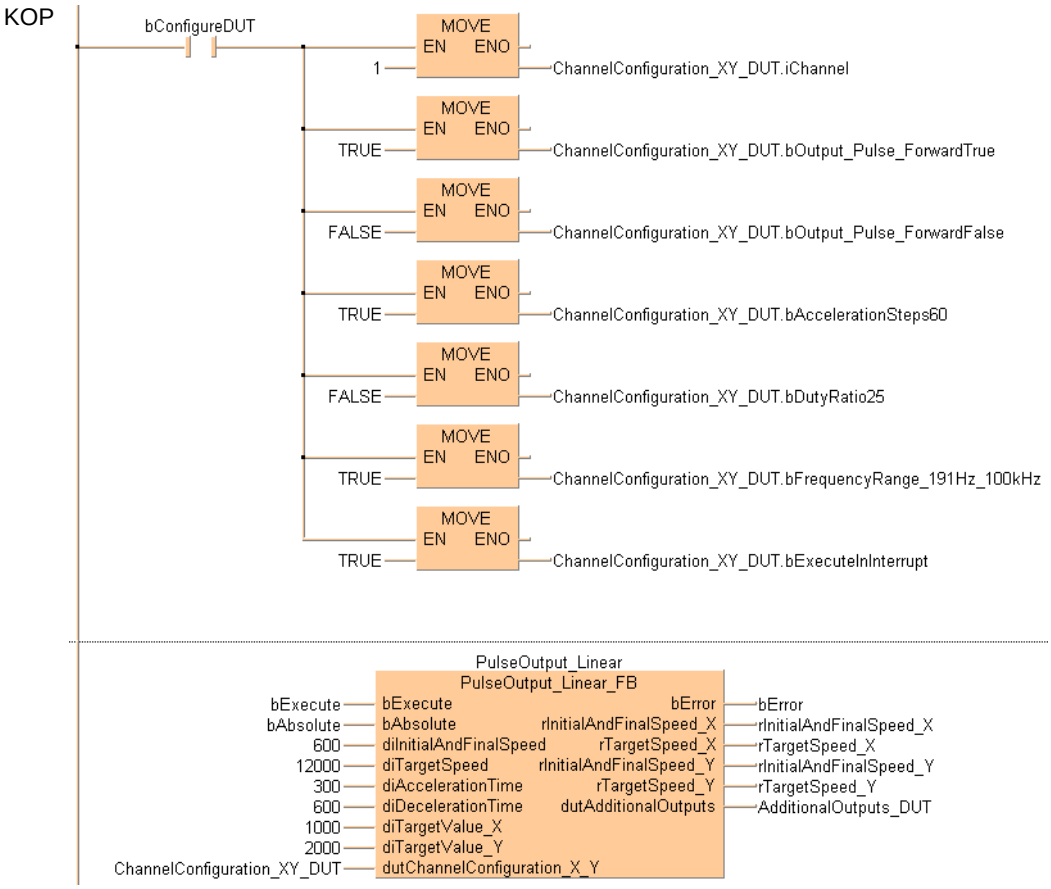
SDT Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: PulseOutput_Channel_Configuration_DUT

	Bezeichner	Typ	Initial	Kommentar
0	iChannel	INT	0	FP-SIGMA: 0, 2
1	bOutput_Pulse_ForwardTrue	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
2	bOutput_Pulse_ForwardFalse	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
3	bAccelerationSteps60	BOOL	TRUE	FP-SIGMA, FP-X: Number of acceleration/deceleration steps: 60 (else 30 steps)
4	bDutyRatio25	BOOL	TRUE	FP-SIGMA, FP-X: Duty ratio (for pulse duration and period): 25% (else 50%)
5	bFrequencyRange_48Hz_100kHz	BOOL	FALSE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 48Hz–100kHz
6	bFrequencyRange_191Hz_100kHz	BOOL	TRUE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 191Hz–100kHz
7	bPulseWidth80µs	BOOL	FALSE	FP0, FP-e Home, Trapezoidal: 80µs (else 50%)
8	iDutyRatioIn10PercentSteps	INT	0	FP0, FP-e Jog: 1-9: duty ratio in increments of 10%, 0: fixed pulse width of 80µs
9	bEnableHomeOnlyAfterNearHomeDeceleration	BOOL	FALSE	FP0R: Type 1 (else type 0), FP-SIGMA, FP-X: Type 2 (else type 1)
10	iHomeInputDeviationCounterClearSignalOutputTime_in_0p5ms	INT	0	FP0R, FP-SIGMA, FP-X: 0 to 200 [x0.5ms]
11	bCalculationOnly	BOOL	FALSE	FP0R: Jog, Trapezoidal: Output operation calculation only (else pulse output)
12	bTrapezoidalMaximumTargetSpeed50kHz	BOOL	FALSE	FP0R: Output operation: Type 1: The target speed can be up to the maximum s...
13	bExecuteInInterrupt	BOOL	FALSE	FP0R Jog positioning, trapezoidal: Execute in or called from interrupt program (el...
14	bJogWithNoCounting	BOOL	FALSE	Only pulse outputs without counting, no target value match. FP-SIGMA, FP-X: b...
15	bContinueAfterDone	BOOL	FALSE	FP-SIGMA circular pulse output: 0=Execution stops when target value has been ...

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	PulseOutput_Linear	PulseOutput_Linear_FB	
1	VAR	bExecute	BOOL	FALSE
2	VAR	bAbsolute	BOOL	FALSE
3	VAR	ChannelConfiguration_XY_DUT	PulseOutput_Channel_Configuration_DUT	
4	VAR	bError	BOOL	FALSE
5	VAR	rInitialAndFinalSpeed_X	REAL	0
6	VAR	rTargetSpeed_X	REAL	0
7	VAR	rInitialAndFinalSpeed_Y	REAL	0
8	VAR	rTargetSpeed_Y	REAL	0
9	VAR	AdditionalOutputs_DUT	PulseOutput_Linear_AdditionalOutputs_DUT	
10	VAR	bConfigureDUT	BOOL	FALSE

KOP

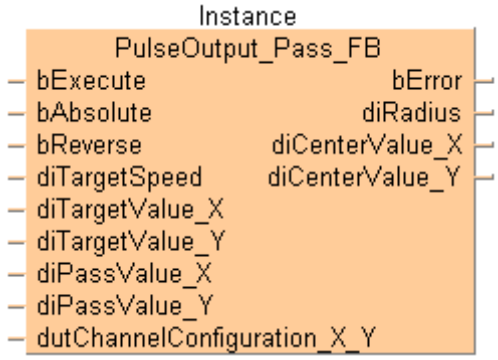

bConfigureDUT
MOVE
EN ENO
1
ChannelConfiguration_XY_DUT.iChannel
TRUE
ChannelConfiguration_XY_DUT.bOutput_Pulse_ForwardTrue
FALSE
ChannelConfiguration_XY_DUT.bOutput_Pulse_ForwardFalse
ChannelConfiguration_XY_DUT.bAccelerationSteps60
ChannelConfiguration_XY_DUT.bDutyRatio25
ChannelConfiguration_XY_DUT.bFrequencyRange_191Hz_100kHz
ChannelConfiguration_XY_DUT.bExecuteInInterrupt
PulseOutput_Linear
PulseOutput_Linear_FB
bExecute
bAbsolute
600
12000
300
600
1000
2000
ChannelConfiguration_XY_DUT
diInitialAndFinalSpeed
diTargetSpeed
diAccelerationTime
diDecelerationTime
diTargetValue_X
diTargetValue_Y
dutChannelConfiguration_X_Y
bError
rInitialAndFinalSpeed_X
rTargetSpeed_X
rInitialAndFinalSpeed_Y
rTargetSpeed_Y
dutAdditionalOutputs
AdditionalOutputs_DUT

PulseOutput_Pass_FB

Kreisinterpolation (Drei-Punkte-Verfahren)

Erklärung

Durch eine zweikanalige Pulsausgabe wird eine bogenförmige Verfahrstrecke erzeugt. Die Parameter für die Pulsausgabe werden in einem Funktionsbaustein und einem SDT festgelegt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F176_PulseOutput_Pass (s. S. 941)

Prüfen Sie mit PulseInfo_IsActive (s. S. 1134), ob der Kontrollmerker für den gewählten Kanal FALSE ist.

SPS Typen

s. S. 1195

Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Aktivierung des Funktionsbausteins
bAbsolute		Absolutwert-Steuerung = TRUE, Relativwertpositionierung = FALSE
bCounterclockwise		Verkettungsmodus: Rückwärts = TRUE, Vorwärts = FALSE
diTargetSpeed	DINT	Sollgeschwindigkeit: Resultierende Geschwindigkeit an beiden Achsen = 100–20000 (100Hz–20kHz)
diTargetValue_X		Sollwert [Pulse]: -8388608–8388607
diTargetValue_Y		
diPassValue_X		
diPassValue_Y		
dutChannelConfiguration_X_Y	Vordefinierter System-SDT für die Kanalkonfiguration: PulseOutput_Channel_Configuration_DUT Kanal: 0, 2	
Ausgangsvariable	**Datentyp**	**Beschreibung**
bError	BOOL	TRUE, wenn ein zugewiesener Eingangswert falsch ist. Der Funktionsbaustein wird nicht weiter ausgeführt.
diRadius	DINT	Radius [Pulse]
diCenterValue_X		Mittelpunkt (x-Achse) [Pulse] = -8388608–8388607
diCenterValue_Y		Mittelpunkt (y-Achse) [Pulse] = -8388608–8388607

Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP)strukturierten Text (ST) programmiert. Ein Beispiel für den strukturierten Text (ST) finden Sie in der Online-Hilfe. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

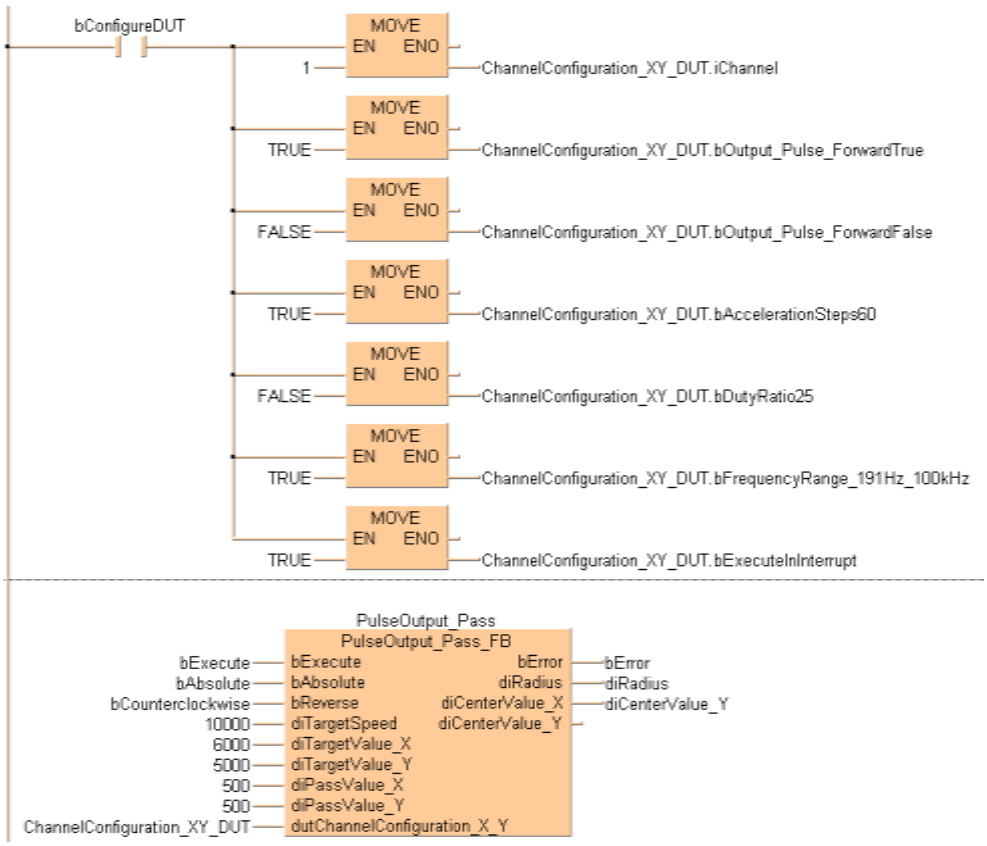
SDT Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: PulseOutput_Channel_Configuration_DUT

	Bezeichner	Typ	Initial	Kommentar
0	iChannel	INT	0	FP-SIGMA: 0, 2
1	bOutput_Pulse_ForwardTrue	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
2	bOutput_Pulse_ForwardFalse	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
3	bAccelerationSteps60	BOOL	TRUE	FP-SIGMA, FP-X: Number of acceleration/deceleration steps: 60 (else 30 steps)
4	bDutyRatio25	BOOL	TRUE	FP-SIGMA, FP-X: Duty ratio (for pulse duration and period): 25% (else 50%)
5	bFrequencyRange_48Hz_100kHz	BOOL	FALSE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 48Hz–100kHz
6	bFrequencyRange_191Hz_100kHz	BOOL	TRUE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 191Hz–100kHz
7	bPulseWidth80µs	BOOL	FALSE	FP0, FP-e Home, Trapezoidal: 80µs (else 50%)
8	iDutyRatioIn10PercentSteps	INT	0	FP0, FP-e Jog: 1-9: duty ratio in increments of 10%, 0: fixed pulse width of 80µs
9	bEnableHomeOnlyAfterNearHomeDeceleration	BOOL	FALSE	FP0R: Type 1 (else type 0), FP-SIGMA, FP-X: Type 2 (else type 1)
10	iHomeInputDeviationCounterClearSignalOutputTime_in_0p5ms	INT	0	FP0R, FP-SIGMA, FP-X: 0 to 200 [x0.5ms]
11	bCalculationOnly	BOOL	FALSE	FP0R: Jog, Trapezoidal: Output operation calculation only (else pulse output)
12	bTrapezoidalMaximumTargetSpeed50kHz	BOOL	FALSE	FP0R: Output operation: Type 1: The target speed can be up to the maximum s...
13	bExecuteInInterrupt	BOOL	FALSE	FP0R Jog positioning, trapezoidal: Execute in or called from interrupt program (el...
14	bJogWithNoCounting	BOOL	FALSE	Only pulse outputs without counting, no target value match. FP-SIGMA, FP-X: b...
15	bContinueAfterDone	BOOL	FALSE	FP-SIGMA circular pulse output: 0=Execution stops when target value has been ...

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	PulseOutput_Pass	PulseOutput_Pass_FB	
1	VAR	bExecute	BOOL	FALSE
2	VAR	bAbsolute	BOOL	FALSE
3	VAR	bContinueAfterDone	BOOL	FALSE
4	VAR	bCounterclockwise	BOOL	FALSE
5	VAR	ChannelConfiguration_XY_DUT	PulseOutput_Channel_Configuration_DUT	
6	VAR	bError	BOOL	FALSE
7	VAR	diRadius	DINT	0
8	VAR	diCenterValue_X	DINT	0
9	VAR	diCenterValue_Y	DINT	0
10	VAR	bConfigureDUT	BOOL	FALSE

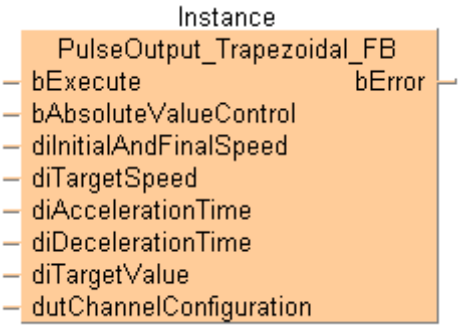
KOP



PulseOutput_Trapezoidal_FB

AUTO-TRAPEZ-Funktion

Erklärung Anhand der Parameter im Funktionsbaustein und dem angegebenen strukturierten Datentyp wird eine Auto-Trapezsteuerung durchgeführt. Die Pulse werden vom angegebenen Kanal ausgegeben, wenn der Kontrollmerker für diesen Kanal FALSE und die Ausführungsbedingung TRUE ist.



Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe:

- FPΣ, FP-X, FP0R: F171_PulseOutput_Trapezoidal (s. S. 904)
- FP0, FP-e: F168_PulseOutput_Trapezoidal (s. S. 891)

Prüfen Sie mit PulseInfo_IsActive (s. S. 1134), ob der Kontrollmerker für den gewählten Kanal FALSE ist.

SPS Typen s. S. 1195

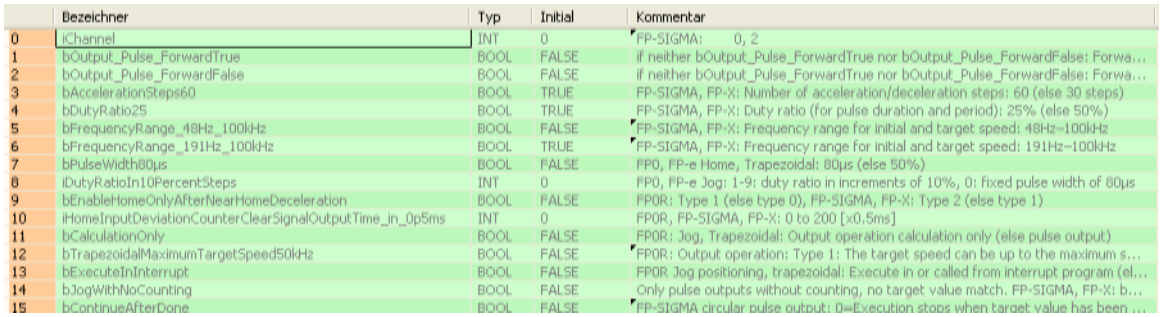
Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	FP-SIGMA, FP-X, FP0, FP-e: Nur mit Flankentrigger FP0R: Mit Flanke oder dauerhaft, wenn eine Änderung der Geschwindigkeit erforderlich ist
bAbsoluteValueControl		Absolutwert-Steuerung = TRUE, Relativwertpositionierung = FALSE
diInitialAndFinalSpeed	DINT	Anfangs- und Endgeschwindigkeit: Setzen Sie diesen Wert gemäß dem in PulseOutput_Channel_Configuration_DUT festgelegten Frequenzbereich: FPΣ, FP-X: 1–9800 (1,5Hz–9,8kHz) 48–100000 (48Hz–100kHz) 191–100000 (191–100kHz) FP0R: 1–50000 (1Hz–50kHz) FP0, FP-e: 40–5000 (40Hz–5kHz)
diTargetSpeed		Sollgeschwindigkeit: Setzen Sie diesen Wert gemäß dem in PulseOutput_Channel_Configuration_DUT festgelegten Frequenzbereich: FPΣ, FP-X: 1–9800 (1,5Hz–9,8kHz) 48–100000 (48Hz–100kHz) 191–100000 (191–100kHz) FP0R: 1–50000 (1Hz–50kHz) FP0, FP-e: 40–5000 (40Hz–5kHz)
diAccelerationTime		Beschleunigungs-/Bremszeit (FPΣ, FP-X): Mit 30 Schritten: 30ms–32760ms (Wert als Vielfaches von 30 angeben) Mit 60 Schritten: 60ms–32760ms (Wert als Vielfaches von 60 angeben) Beschleunigungs-/Bremszeit (FP0, FP-e): 30ms–32760ms Beschleunigungszeit (FP0R): 1ms–32760ms
diDecelerationTime		Bremszeit (FP0R): 1ms–32760ms
diTargetValue		Sollwert [Pulse]: -2147483648–2147483647

dutChannelConfiguration	Vordefinierter System-SDT für die Kanalkonfiguration: PulseOutput_Channel_Configuration_DUT	
Ausgangsvariable	**Datentyp**	**Beschreibung**
bError	BOOL	TRUE, wenn ein zugewiesener Eingangswert falsch ist. Der Funktionsbaustein wird nicht weiter ausgeführt.

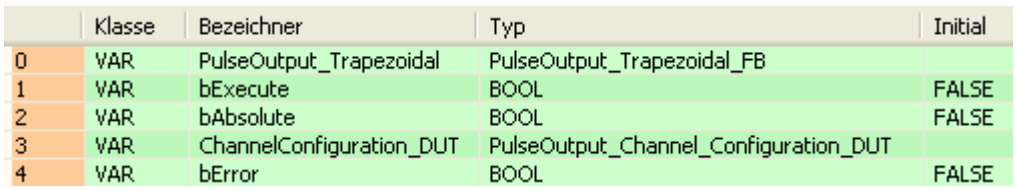
Beispiel In diesem Beispiel wird die Funktion im Kontaktplan (KOP)strukturierten Text (ST) programmiert. Ein Beispiel für den strukturierten Text (ST) finden Sie in der Online-Hilfe. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: PulseOutput_Channel_Configuration_DUT



	Bezeichner	Typ	Initial	Kommentar
0	iChannel	INT	0	FP-SIGMA: 0, 2
1	bOutput_Pulse_ForwardTrue	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
2	bOutput_Pulse_ForwardFalse	BOOL	FALSE	if neither bOutput_Pulse_ForwardTrue nor bOutput_Pulse_ForwardFalse: Forwa...
3	bAccelerationSteps60	BOOL	TRUE	FP-SIGMA, FP-X: Number of acceleration/deceleration steps: 60 (else 30 steps)
4	bDutyRatio25	BOOL	TRUE	FP-SIGMA, FP-X: Duty ratio (for pulse duration and period): 25% (else 50%)
5	bFrequencyRange_48Hz_100kHz	BOOL	FALSE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 48Hz–100kHz
6	bFrequencyRange_191Hz_100kHz	BOOL	TRUE	FP-SIGMA, FP-X: Frequency range for initial and target speed: 191Hz–100kHz
7	bPulseWidth80µs	BOOL	FALSE	FP0, FP-e Home, Trapezoidal: 80µs (else 50%)
8	iDutyRatioIn10PercentSteps	INT	0	FP0, FP-e Jog: 1-9: duty ratio in increments of 10%, 0: fixed pulse width of 80µs
9	bEnableHomeOnlyAfterNearHomeDeceleration	BOOL	FALSE	FP0R: Type 1 (else type 0), FP-SIGMA, FP-X: Type 2 (else type 1)
10	iHomeInputDeviationCounterClearSignalOutputTime_in_0p5ms	INT	0	FP0R, FP-SIGMA, FP-X: 0 to 200 [x0.5ms]
11	bCalculationOnly	BOOL	FALSE	FP0R: Jog, Trapezoidal: Output operation calculation only (else pulse output)
12	bTrapezoidalMaximumTargetSpeed50kHz	BOOL	FALSE	FP0R: Output operation: Type 1: The target speed can be up to the maximum s...
13	bExecuteInInterrupt	BOOL	FALSE	FP0R Jog positioning, trapezoidal: Execute in or called from interrupt program (el...
14	bJogWithNoCounting	BOOL	FALSE	Only pulse outputs without counting, no target value match. FP-SIGMA, FP-X: b...
15	bContinueAfterDone	BOOL	FALSE	FP-SIGMA circular pulse output: 0=Execution stops when target value has been ...

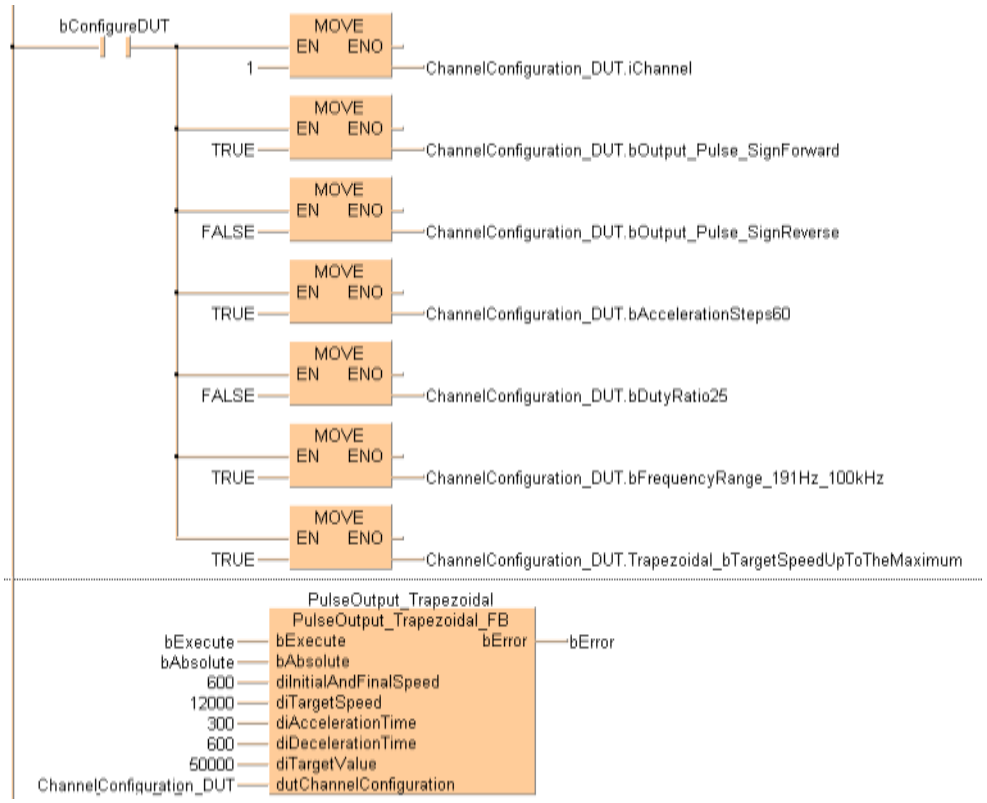
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



	Klasse	Bezeichner	Typ	Initial
0	VAR	PulseOutput_Trapezoidal	PulseOutput_Trapezoidal_FB	
1	VAR	bExecute	BOOL	FALSE
2	VAR	bAbsolute	BOOL	FALSE
3	VAR	ChannelConfiguration_DUT	PulseOutput_Channel_Configuration_DUT	
4	VAR	bError	BOOL	FALSE

Rumpf

KOP


bConfigureDUT
MOVE
EN ENO
1
ChannelConfiguration_DUT.iChannel
TRUE
ChannelConfiguration_DUT.bOutput_Pulse_SignForward
FALSE
ChannelConfiguration_DUT.bOutput_Pulse_SignReverse
ChannelConfiguration_DUT.bAccelerationSteps60
ChannelConfiguration_DUT.bDutyRatio25
ChannelConfiguration_DUT.bFrequencyRange_191Hz_100kHz
ChannelConfiguration_DUT.Trapezoidal_bTargetSpeedUpToTheMaximum
PulseOutput_Trapezoidal
PulseOutput_Trapezoidal_FB
bExecute
bAbsolute
600
diInitialAndFinalSpeed
12000
diTargetSpeed
300
diAccelerationTime
600
diDecelerationTime
50000
diTargetValue
ChannelConfiguration_DUT
dutChannelConfiguration
bError

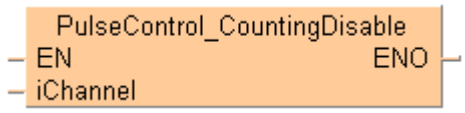
38.2 Pulssteuerungsbefehle

In diesem Abschnitt:

PulseControl_Counting Disable

Zähler am Pulsausgabekanal deaktivieren

Erklärung Dieser Befehl deaktiviert den Zähler am durch **iChannel** festgelegten Kanal. Bit 1 des Pulsausgabe-Steuercodes (s. S. 865) wird auf TRUE gesetzt.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseInfo_IsCountingDisabled (s. S. 1136)
- PulseControl_CountingEnable (s. S. 1115)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn der Pulszähler ausgeschaltet ist

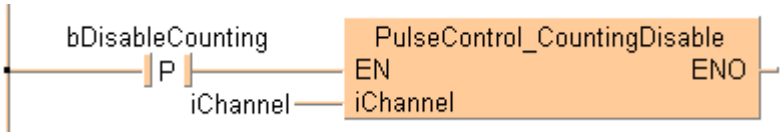
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bDisableCounting	BOOL	FALSE

Rumpf

KOP



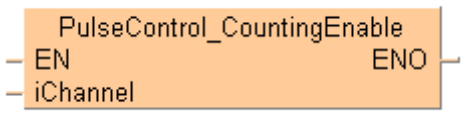
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bDisableCounting) then
    PulseControl_CountingDisable(iChannel := iChannel);
end_if;
```

PulseControl_CountingEnable

Zähler am Pulsausgabekanal einschalten

Erklärung Dieser Befehl aktiviert den Zähler an einem mit **iChannel** festgelegten Pulsausgabekanal, der zuvor mit PulseControl_CountingDisable (s. S. 1114) deaktiviert worden ist. Bit 1 des Pulsausgabe-Steuercodes (s. S. 865) wird auf FALSE gesetzt.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseControl_CountingDisable (s. S. 1114)
- PulseInfo_IsCountingDisabled (s. S. 1136)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

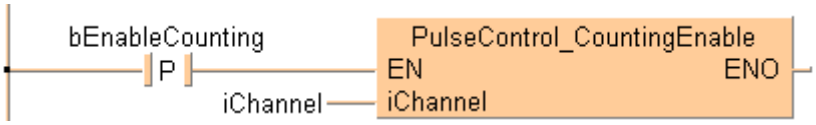
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bEnableCounting	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf

KOP



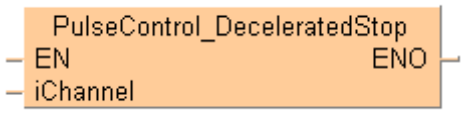
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bEnableCounting) then
    PulseControl_CountingEnable(iChannel := iChannel);
end_if;
```

PulseControl_DeceleratedStop

Gebremsten Halt ausfÃ¼hren

Erklärung Dieser Befehl führt am durch **iChannel** festgelegten Kanal einen gebremsten Halt aus. Wenn während der Beschleunigung das Signal für einen gebremsten Halt ausgelöst wird, wird der Bremsvorgang mit derselben Geschwindigkeit ausgeführt wie der normale Bremsvorgang, der von der Sollgeschwindigkeit ausgeht.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseControl_PulseOutputStop (s. S. 1125)
- PulseInfo_IsPulseOutputStopped (s. S. 1139)
- FP-Befehle: Pulsausgabe-Steuercode schreiben (s. S. 865) (FP0R)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

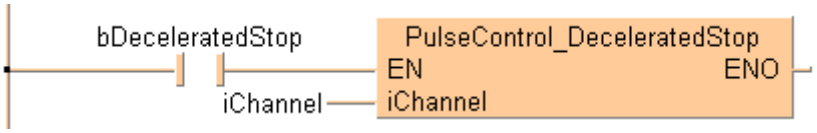
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bDeceleratedStop	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf

KOP



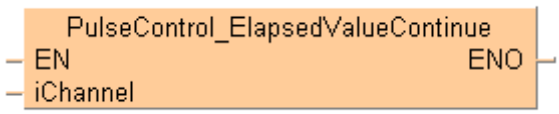
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if (bDeceleratedStop) then
    PulseControl_DeceleratedStop(iChannel := iChannel);
end_if;
```

PulseControl_ElapsedValueContinue

Pulszählung nach Reset fortsetzen

Erklärung Dieser Befehl setzt das Zählen auf einem mit **iChannel** festgelegten Kanal fort, dessen Istwert zuvor mit PulseControl_ElapsedValueReset (s. S. 1119) zurückgesetzt wurde. Bit 0 des Steuercodes der Pulsausgabe (s. S. 865) wird auf FALSE gesetzt.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseInfo_IsElapsedValueReset (s. S. 1137)
- PulseInfo_ReadElapsedValue (s. S. 1144)
- PulseControl_ElapsedValueContinue (s. S. 1119)
- PulseControl_WriteElapsedValue (s. S. 1127)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

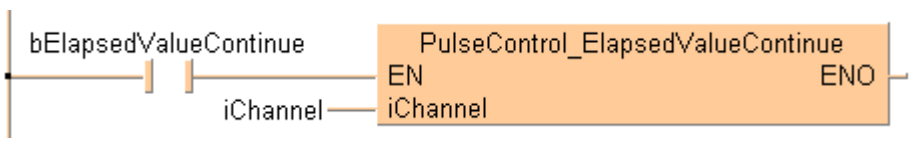
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bElapsedValueContinue	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf

KOP



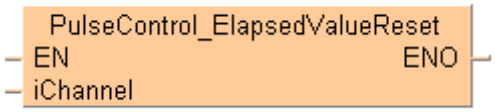
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if (bElapsedValueContinue) then
    PulseControl_ElapsedValueContinue(iChannel := iChannel);
end_if;
```

PulseControl_ElapsedValueReset

Istwert auf 0 setzen

Erklärung Dieser Befehl setzt den Istwert am durch iChannel festgelegten Pulsausgabekanal auf 0. Bit 0 des Pulsausgabe-Steuercodes (s. S. 865) wird auf TRUE gesetzt. Mit PulseControl_ElapsedValueContinue (s. S. 1118) können Sie die Pulszählung fortsetzen. Mit PulseInfo_IsElapsedValueReset (s. S. 1137) prüfen Sie den aktuellen Status. Durch das Zurücksetzen des Istwerts wird die Pulsausgabe nicht unterbrochen.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseInfo_IsElapsedValueReset (s. S. 1137)
- PulseControl_ElapsedValueContinue (s. S. 1118)
- PulseControl_WriteElapsedValue (s. S. 1127)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

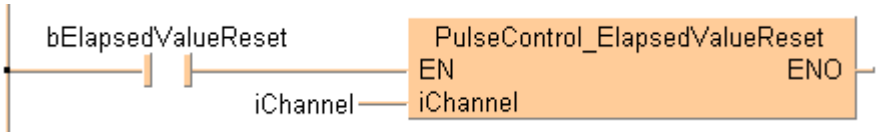
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bElapsedValueReset	BOOL	FALSE
1	VAR	iChannel_Kopie	INT	0

Rumpf

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

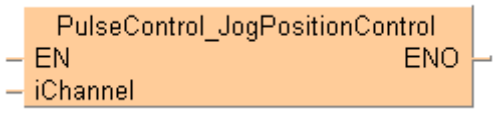
```
if (bElapsedValueReset) then
    PulseControl_ElapsedValueReset(iChannel := iChannel);
end_if;
```

PulseControl_JogPositionControl

Positionierung starten

Erklärung

Dieser Befehl setzt Bit 6 des Pulsausgabe-Steuercodes (s. S. 865) auf TRUE und zurück auf FALSE, um die Positionierung am durch **iChannel** festgelegten Kanal zu starten. Der Positionierungstrigger wird von den Tipp-Betriebsbefehlen PulseOutput_Jog_Positioning0_FB (s. S. 1099) und PulseOutput_Jog_Positioning1_FB (s. S. 1101) verwendet.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- FP-Befehle: F171_PulseOutput_Jog_Positioning (s. S. 914)

SPS Typen

s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

Beispiel

In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

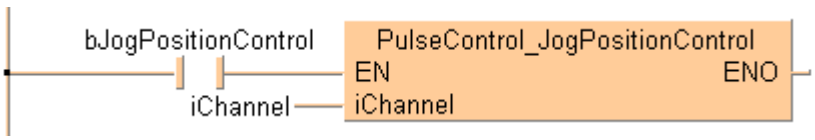
POE-Kopf

Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bJogPositionControl	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf

KOP



ST

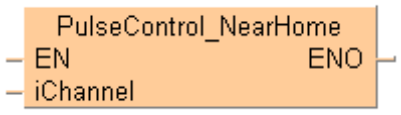
Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if (bJogPositionControl) then
    PulseControl_JogPositionControl(iChannel := iChannel);
end_if;
```

PulseControl_NearHome

Abbremsung startet am Referenzpunkt-Sucheingang

Erklärung Dieser Befehl startet die Abbremsung am durch **iChannel** festgelegten, wenn der Referenzpunkt-Sucheingang aktiviert wird, indem Bit 4 des Pulsausgabe-Steuercodes (s. S. 865) auf TRUE und wieder zurück auf FALSE gesetzt wird. Mit PulseInfo_IsHomeInputTrue (s. S. 1138) prüfen Sie, ob der Referenzpunkteingang TRUE ist.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091), PulseOutput_Home_FB (s. S. 1094)
- FP-Befehle:

F168_PulseOutput_Home (s. S. 894) (FP0, FP-e)

F171_PulseOutput_Home (s. S. 910) (FP_{Σ}, FP-X)

F177_PulseOutput_Home (s. S. 945) (FP0R)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

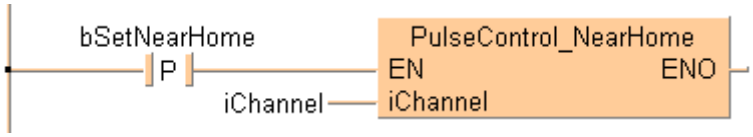
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bSetNearHome	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf

KOP



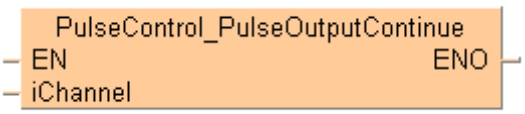
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bSetNearHome) then
    PulseControl_NearHome(iChannel := iChannel);
end_if;
```

PulseControl_PulseOutput Continue

Pulsausgabe fortsetzen

Erklärung Dieser Befehl setzt die Pulsausgabe am durch **iChannel** festgelegten Kanal fort, nachdem die Pulsausgabe mit PulseControl_PulseOutputStop (s. S. 1125) gestoppt wurde. Bit 3 des Pulsausgabe-Steuercodes (s. S. 865) wird auf FALSE gesetzt.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseControl_PulseOutputStop (s. S. 1125)
- PulseInfo_IsPulseOutputStopped (s. S. 1139)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

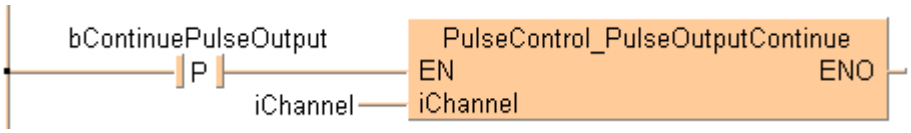
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bContinuePulseOutput	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf

KOP



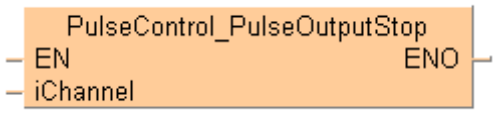
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bContinuePulseOutput) then
    PulseControl_PulseOutputContinue(iChannel := iChannel);
end_if;
```

PulseControl_PulseOutputStop

Pulsausgabe anhalten

Erklärung Dieser Befehl hält die Pulsausgabe am durch **iChannel** festgelegten Kanal an, indem Bit 3 des Pulsausgabe-Steuercodes (s. S. 865) auf TRUE gesetzt wird. Mit PulseControl_PulseOutputContinue (s. S. 1124) setzen Sie die Pulsausgabe nach der Unterbrechung wieder fort.



Siehe auch:

- PulseInfo_IsPulseOutputStopped (s. S. 1139)
- PulseControl_PulseOutputContinue (s. S. 1124)
- Pulsausgabe beenden

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

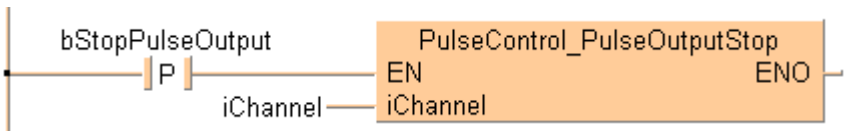
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Class	Identifier	Type	Initial
0	VAR	bStopPulseOutput	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf

KOP



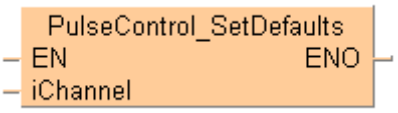
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bStopPulseOutput) then
    PulseControl_PulseOutputStop(iChannel := iChannel);
end_if;
```

PulseControl_SetDefaults

Standardwerte für Pulsausgabekanal setzen

Erklärung Dieser Befehl setzt alle Bits des Pulsausgabe-Steuercodes (s. S. 865) am durch **iChannel** festgelegten Kanal auf 0. 0 ist die Standardeinstellung.



Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

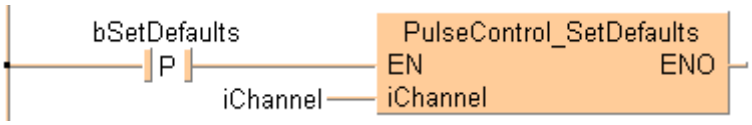
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bSetDefaults	BOOL	FALSE
1	VAR	iChannel	INT	0

Rumpf

KOP



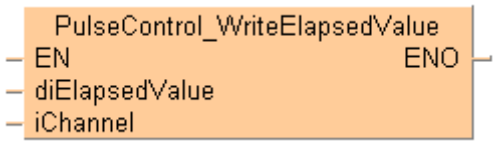
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bSetDefaults) then
    PulseControl_SetDefaults(iChannel := iChannel);
end_if;
```

PulseControl_WriteElapsedValue

Istwert in Pulsausgabekanal schreiben

Erklärung Dieser Befehl schreibt einen Istwert in den durch **iChannel** festgelegten Pulsausgabekanal.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseInfo_IsElapsedValueReset (s. S. 1137)
- PulseControl_ElapsedValueContinue (s. S. 1118)
- PulseInfo_ReadElapsedValue (s. S. 1144)
- Istwert schreiben und lesen (s. S. 870)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
diElapsedValue	DINT	In Kanal **iChannel** zu schreibender Istwert.
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

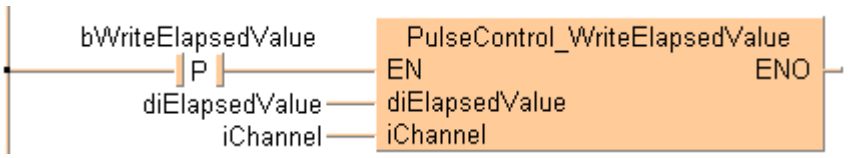
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	bWriteElapsedValue	BOOL	FALSE
1	VAR	diElapsedValue	DINT	5000
2	VAR	iChannel	INT	0

Rumpf

KOP



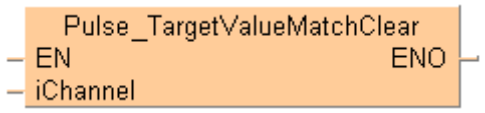
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
if DF(bWriteElapsedValue) then
    PulseControl_WriteElapsedValue(ElapsedValue := diElapsedValue,
    iChannel := iChannel);
end_if;
```

Pulse_TargetValueMatchClear

Zählervergleichsfunktion löschen

Erklärung Dieser Befehl entfernt die Zählervergleichsfunktion an dem durch **iChannel** festgelegten Kanal.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseInfo_IsTargetValueMatchActive (s. S. 1140)
- PulseInfo_ReadTargetValueMatchValue (s. S. 1146)

Wenn Sie einen Freigabeeingang und -ausgang zum Befehl hinzufügen möchten, wählen Sie [Mit EN/ENO] im Fenster "Befehle" (in den Editoren KOP, FBS und AWL). Einen zuvor verwendeten Befehl können Sie auch im Kontextmenü unter "Zuletzt verwendet" auswählen oder mit <Strg>+<Umschalt>+<v> im Programmierfenster einfügen.

SPS Typen s. S. 1207

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

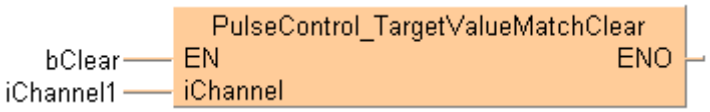
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Class	Identifier	Type	Initial
0	VAR	iChannel1	INT	1
1	VAR	bClear	BOOL	FALSE

Rumpf

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
Pulse_TargetValueMatchClear(iChannel := iChannel);
```

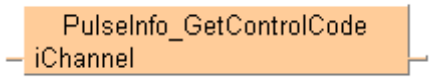
38.3 Informationsbefehle für Pulsausgabe

In diesem Abschnitt:

PulseInfo_GetControlCode

Steuercode des Pulsausgabekanals zurückgeben

Erklärung Dieser Befehl gibt den Steuercode (s. S. 865) des durch **iChannel** festgelegten Pulsausgabekanals zurück.



Siehe auch: Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	WORD	Speichert den Steuercode

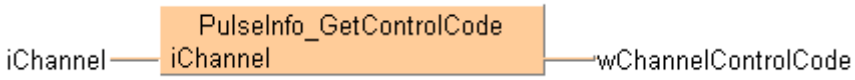
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	wChannelControlCode	WORD	0

Rumpf

KOP



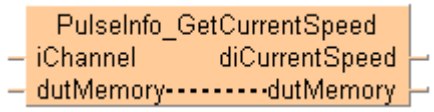
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
wChannelControlCode := PulseInfo_GetControlCode (iChannel := iChannel);
```

PulseInfo_GetCurrentSpeed

Aktuelle Pulsausgabefrequenz zurückgeben

Erklärung Dieser Befehl gibt die aktuelle Pulsausgabegeschwindigkeit in Hz des durch **iChannel** festgelegten Pulsausgabekanals zurück.



Siehe auch: Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)

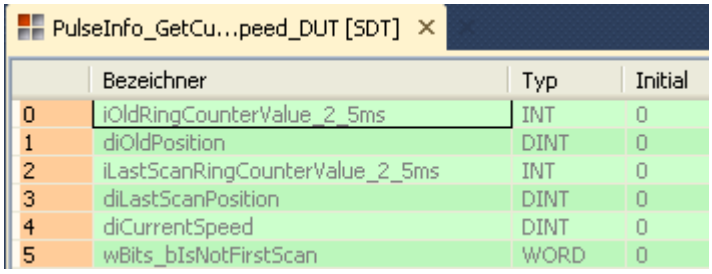
SPS Typen s. S. 1195

Datentypen

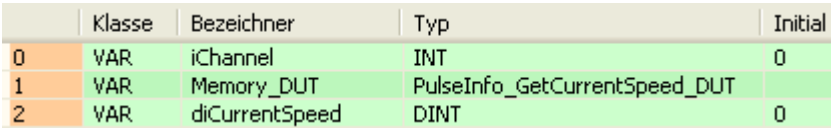
Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
dutMemory	PulseInfo_GetCurrentSpeed_DUT	
diCurrentSpeed	DINT	Aktuelle Pulsausgabegeschwindigkeit in Hz

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

SDT Verwenden Sie folgenden, vordefinierten strukturierten Datentyp: PulseInfo_GetCurrentSpeed_DUT

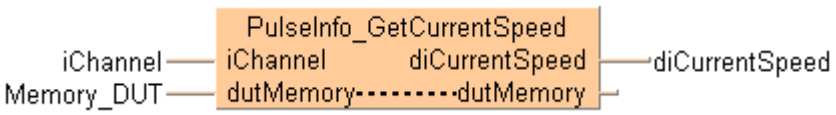


POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



Rumpf

KOP



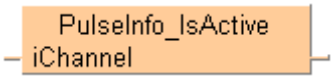
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
PulseInfo_GetCurrentSpeed(iChannel := iChannel,
          dutMemory := Memory_DUT,
          diCurrentSpeed => diCurrentSpeed);
```

PulseInfo_IsActive

Check if pulse output is active

Erklärung Dieser Befehl wertet den Kontrollmerker der Pulsausgabe aus und gibt TRUE zurück, wenn der durch **iChannel** festgelegte Pulsausgabekanal aktiv ist.



Siehe auch: Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn die Pulsausgabe aktiv ist

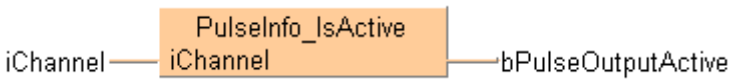
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bPulseOutputActive	BOOL	FALSE

Rumpf

KOP



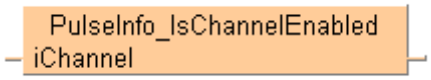
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bPulseOutputActive := PulseInfo_IsActive(iChannel := iChannel);
```

PulseInfo_IsChannelEnabled

Aktiven Status des Pulsausgabekanals prüfen

Erklärung Dieser Befehl gibt TRUE zurück, wenn der durch **iChannel** festgelegte Pulsausgabekanal in den Systemregistern aktiviert ist und durch den ausgewählten SPS-Typ unterstützt wird.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- Erforderliche Systemregistereinstellungen

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn der durch **iChannel** festgelegte Kanal aktiv ist

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bChannelEnabled	BOOL	FALSE

KOP

ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bChannelEnabled := PulseInfo_IsChannelEnabled(iChannel := iChannel);
```

PulseInfo_IsCounting Disabled

Inaktiven Status des Pulszählers prüfen

Erklärung Dieser Befehl gibt TRUE zurück, wenn der Zähler am durch **iChannel** festgelegten Kanal deaktiviert wurde.

Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091), PulseControl_CountingDisable (s. S. 1114), PulseControl_CountingEnable (s. S. 1115)
- FP-Befehle: Zählen aktivieren/deaktivieren (s. S. 865)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bCountingDisabled	BOOL	FALSE

Rumpf

KOP

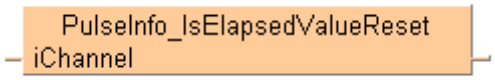
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bCountingDisabled := PulseInfo_IsCountingDisabled(iChannel := iChannel);
```

PulseInfo_IsElapsedValueReset

Istwert = 0 prüfen

Erklärung Dieser Befehl gibt TRUE zurück, wenn der Istwert des durch **iChannel** festgelegten Pulsausgabekanals auf 0 zurückgesetzt wurde.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseInfo_ReadElapsedValue (s. S. 1144)
- PulseControl_ElapsedValueReset (s. S. 1119)
- PulseControl_ElapsedValueContinue (s. S. 1118)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn der durch **iChannel** festgelegte Kanal zurückgesetzt wurde

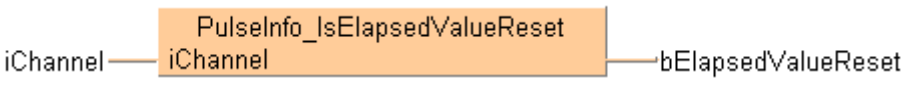
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bElapsedValueReset	BOOL	FALSE

Rumpf

KOP



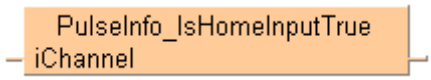
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bElapsedValueReset := PulseInfo_IsElapsedValueReset (iChannel := iChannel);
```

PulseInfo_IsHomeInputTrue

Status TRUE von Referenzpunkteingang prüfen

Erklärung Dieser Befehl gibt TRUE zurück, wenn der Referenzpunkteingang am durch **iChannel** festgelegten Kanal auf TRUE steht.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseOutput_Home_FB (s. S. 1094)

SPS Typen s. S. 1195

Datentypen

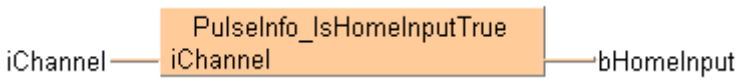
Variable	Datentyp	Beschreibung
Ausgangsvariable	BOOL	TRUE, wenn der Referenzpunkteingang erreicht ist

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bHomeInput	BOOL	FALSE

KOP



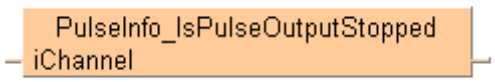
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bHomeInput := PulseInfo_IsHomeInputTrue (iChannel := iChannel);
```

PulseInfo_IsPulseOutputStopped

Ende der Pulsausgabe prüfen

Erklärung Dieser Befehl gibt TRUE zurück, wenn die Pulsausgabe beendet wurde, z.B. mit PulseControl_DeceleratedStop (s. S. 1116) oder PulseControl_PulseOutputStop (s. S. 1125). Mit PulseControl_PulseOutputContinue (s. S. 1124) aktivieren Sie die Pulsausgabe wieder.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- Pulsausgabe beenden

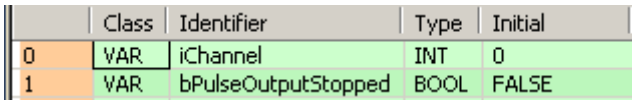
SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn die Pulsausgabe beendet ist

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

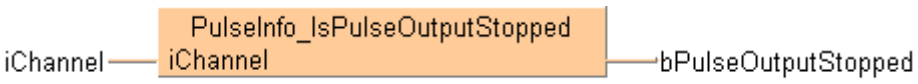
POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.



	Class	Identifier	Type	Initial
0	VAR	iChannel	INT	0
1	VAR	bPulseOutputStopped	BOOL	FALSE

Rumpf

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bPulseOutputStopped := PulseInfo_IsPulseOutputStopped(iChannel :=
iChannel);
```

PulseInfo_IsTarget ValueMatchActive

Aktiven Status der Zählervergleichsfunktion prüfen

Erklärung Dieser Befehl gibt TRUE zurück, wenn die Zählervergleichsfunktion (s. S. 1148) an dem durch **iChannel** festgelegten Kanal aktiv ist.

Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseControl_TargetValueMatchClear (s. S. 1129)
- PulseInfo_ReadTargetValueMatchValue (s. S. 1146)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	BOOL	TRUE, wenn die Zählervergleichsfunktion aktiv ist

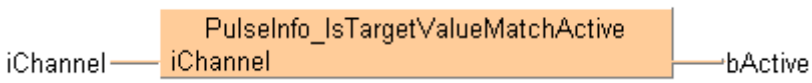
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	bActive	BOOL	FALSE

Rumpf

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
bActive := PulseInfo_IsTargetValueMatchActive(iChannel := iChannel);
```

PulseInfo_ReadAccelerationForbiddenAreaStartingPosition

Beschleunigungsgrenzwert lesen

Erklärung Dieser Befehl liest den Anfangswert eines nicht für die Beschleunigung zulässigen Bereichs. Wenn der Istwert diesen Wert während der Beschleunigungsphase erreicht, wird die Pulsauausgabe nicht weiter beschleunigt.

Siehe auch: Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	DINT	Speichert den Beschleunigungsgrenzwert

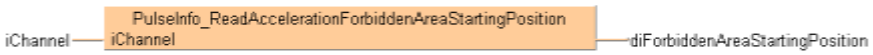
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	diForbiddenAreaStartingPosition	DINT	0

Rumpf

KOP



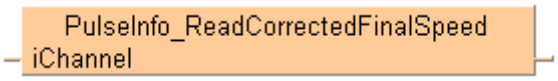
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
diForbiddenAreaStartingPosition :=
PulseInfo_ReadAccelerationForbiddenAreaStartingPosition(iChannel :=
iChannel);
```

PulseInfo_ReadCorrectedFinalSpeed

Korrigierte Endgeschwindigkeit lesen

Erklärung Dieser Befehl gibt den Wert für die korrigierte Endgeschwindigkeit am durch **iChannel** festgelegten Kanal zurück.



Siehe auch: Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable		Speichert den Wert der korrigierten Endgeschwindigkeit

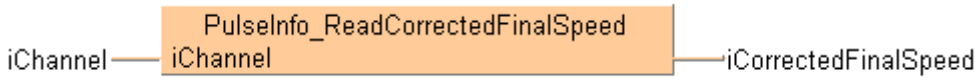
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	iCorrectedFinalSpeed	INT	0

Rumpf

KOP



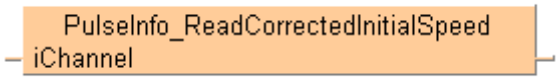
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
iCorrectedFinalSpeed := PulseInfo_ReadCorrectedFinalSpeed(iChannel :=
iChannel);
```

PulseInfo_ReadCorrectedInitialSpeed

Korrigierte Anfangsgeschwindigkeit lesen

Erklärung Dieser Befehl gibt den Wert für die korrigierte Anfangsgeschwindigkeit am durch **iChannel** festgelegten Kanal zurück.



Siehe auch: Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable		Speichert den Wert der korrigierten Anfangsgeschwindigkeit

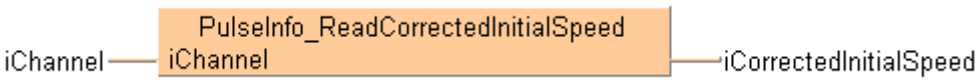
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	iCorrectedInitialSpeed	INT	0

Rumpf

KOP



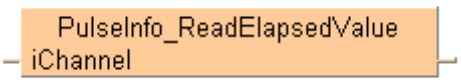
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
iCorrectedInitialSpeed := PulseInfo_ReadCorrectedInitialSpeed(iChannel :=
iChannel);
```

PulseInfo_ReadElapsedValue

Istwert von Pulsausgabekanal lesen

Erklärung Dieser Befehl liest den Istwert aus dem durch **iChannel** festgelegten Pulsausgabekanal. Mit PulseControl_WriteElapsedValue (s. S. 1127) ändern Sie den Istwert und mit PulseControl_ElapsedValueReset (s. S. 1119) setzen Sie den Istwert auf 0.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- PulseInfo_IsElapsedValueReset (s. S. 1137)
- PulseControl_ElapsedValueContinue (s. S. 1118)
- PulseControl_WriteElapsedValue (s. S. 1127)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	DINT	Speichert den Istwert des durch **iChannel** festgelegten Kanals

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	diElapsedValue	DINT	0

Rumpf

KOP

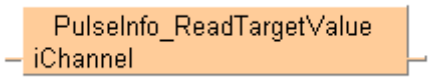
ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
diElapsedValue := PulseInfo_ReadElapsedValue(iChannel := iChannel);
```

PulseInfo_ReadTargetValue

Reads the target value from a pulse output channel

Erklärung Dieser Befehl liest den Sollwert von dem durch **iChannel** festgelegten Pulsausgabekanal.



Siehe auch:

- Tool-Befehle:

- Tool-Befehle für die schnellen Zähler (s. S. 1061)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	DINT	Speichert den Sollwert des durch **iChannel** festgelegten Kanals

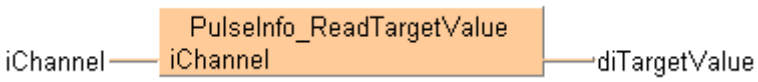
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ
0	VAR	iChannel	INT
1	VAR	diTargetValue	DINT

Rumpf

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
diTargetValue := PulseInfo_ReadTargetValue (iChannel := iChannel);
```

PulseInfo_ReadTargetValueMatchValue

ulseInfo_ReadTargetValueMatchValue,Sollwert-Pulsauswertung von Pulsausgabekanal lesen

Erklärung Dieser Befehl gibt die Sollwert der Ausgangssteuerung am durch **iChannel** festgelegten Pulsausgabekanal zurück. Der Sollwert aus der Ausgangssteuerung wird von den Zählervergleichsfunktionen verwendet.

Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- Pulse_TargetValueMatch_Set (s. S. 1150)
- Pulse_TargetValueMatch_Reset (s. S. 1148)
- Pulse_TargetValueMatch_Clear (s. S. 1129)
- Info_IsTargetValueMatch_Active (s. S. 1140)

SPS Typen s. S. 1195

Datentypen

Variable	Datentyp	Beschreibung
iChannel	INT	Pulsausgabekanal FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
Ausgangsvariable	DINT	Speichert den Sollwert der Ausgangssteuerung

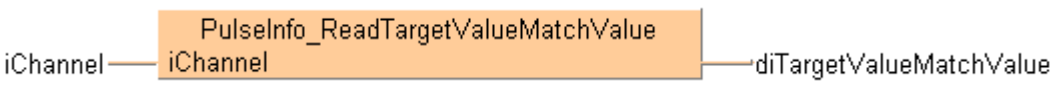
Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR	iChannel	INT	0
1	VAR	diTargetValueMatchValue	DINT	0

Rumpf

KOP



ST Beim Programmieren mit strukturiertem Text geben Sie Folgendes ein:

```
diTargetValueMatchValue := PulseInfo_ReadTargetValueMatchValue(iChannel :=
iChannel);
```

38.4 Zählervergleichsfunktionen der Pulsausgabe

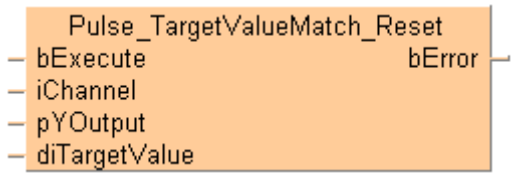
In diesem Abschnitt:

Pulse_TargetValue Match_Reset

Zählervergleichsausgang zurücksetzen (Pulsausgabe)

Erklärung

Wenn der Istwert den Sollwert **diTargetValue** des durch **iChannel** festgelegten Pulsausgabekanals erreicht, wird der durch **pYOutput** festgelegte Ausgang sofort auf FALSE gesetzt.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- F167_PulseOutput_Reset (s. S. 888)

Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F167_PulseOutput_Reset (s. S. 888)

Um die Kombination von Kanal und Y-Ausgang zu prüfen, benötigt der Compiler das folgende Benennungsmuster für globale Variablen: `%s`Pulse_TargetValueMatch_`Kanal%d_Y%d`_`%s`
Verwenden Sie für globale Variablen in Zählervergleichsfunktionen grundsätzlich dieses Muster:

- `Channel%d` muss ein Pulsausgabekanal sein, der in den Systemregistern aktiviert ist.
- `Y%d` muss eine explizite, von der SPS unterstützte Ausgangsadresse sein.

FP-Σ, FP0, FP-e:	Y0–Y7
FP-Σ (V3.1 oder neuer), FP0R:	Y0–Y1F
FP-X:	Y0–Y29F

- `%s` ist ein optionaler Deskriptor am Musteranfang
- `%s` ist ein optionaler Deskriptor am Musterende

Optional	Vorgegebenes Muster	Optional
g_b	Pulse_TargetValueMatch_ChannelA_Y11F	_MotorOn

Diese globale Variable erzeugt den Code für Kanal **A** und Ausgang **Y11F**.

SPS Typen

s. S. 1195

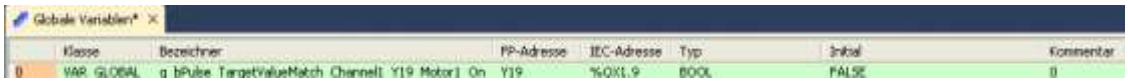
Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Eine steigende Flanke aktiviert die Funktion; Auswertung des Kontrollmerkers "Pulsausgabekanal" mit PulseInfo_IsTargetValueMatchActive (s. S. 1140)
iChannel	INT	FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
pYOutput	POINTER	Durch GetPointer erhaltenes Zeigerergebnis von einer globalen Variable, die Kanalnummer und Ausgang liefert.

Eingangsvariable	Datentyp	Beschreibung
diTargetValue	DINT	Geben Sie für den Sollwert einen 32-Bit-Datenwert innerhalb des folgenden Bereichs ein: FP0, FP-e: -838808–+8388607 FPΣ, FP-X, FP0R: -2147483467–+2147483648
Ausgangsvariable	**Datentyp**	**Beschreibung**
bError	BOOL	TRUE, wenn die Kombination von **Channel%d** und **pYOuput.iOffset** nicht mit einer gültigen Kombination von Kanalnummer und Ausgang übereinstimmt, wie sie von der globalen Variablen vorgegeben wird.

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. (ST).Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

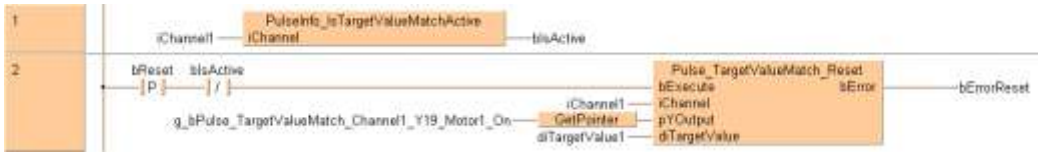


POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

	Klasse	Bezeichner	Typ	Initial
0	VAR_EXTERNAL	g_bPulse_TargetValueMatch_Channel1_YA_Horn1_On	BOOL	FALSE
1	VAR_EXTERNAL	g_bPulse_TargetValueMatch_Channel1_Y19_Motor1_On	BOOL	FALSE
2	VAR	diTargetValue0	DINT	11000
3	VAR	iChannel1	INT	1
4	VAR	diTargetValue1	DINT	22000
5	VAR	bReset	BOOL	FALSE
6	VAR	bIsActive	BOOL	FALSE
7	VAR	bErrorReset	BOOL	FALSE
8	VAR	bErrorSet	BOOL	FALSE
9	VAR	bSet	BOOL	FALSE

Rumpf Verwenden Sie PulseInfo_IsTargetValueMatchActive (s. S. 1140), um zu ermitteln, ob der durch **iChannel1** festgelegte Kanal aktiv ist. Wenn eine steigende Flanke bei **bReset** erkannt wird, und wenn **bIsActive nicht** TRUE ist, wird der Befehl ausgeführt. Die Kombination aus Kanalnummer und Ausgangskontakt wird in der globalen Variablen **g_bPulse_TargetValueMatch_Channel1_Y19_Motor1_On**überprüft. Wenn die Pulsausgabe an Kanal 1 den Sollwert **diTargetValue1** erreicht, wird der Ausgang **Y19** auf FALSE gesetzt.

KOP



ST

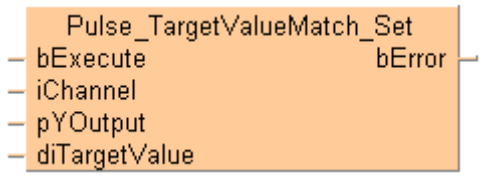
```
bIsActive:=PulseInfo_IsTargetValueMatchActive(iChannel1);
Pulse_TargetValueMatch_Reset(bExecute := DF(bReset) AND NOT bIsActive,
                iChannel := iChannel1,
                pYOutput :=
GetPointer(g_bPulse_TargetValueMatch_Channel1_Y19_Motor1_On),
                diTargetValue := diTargetValue1,
                bError => bErrorReset);
```

Pulse_TargetValue Match_Set

Zählervergleichsausgang setzen (Pulsausgabe)

Erklärung

Wenn der Istwert den Sollwert **diTargetValue** des durch **iChannel** festgelegten Pulsausgabekanals erreicht, wird der durch **pYOutput** festgelegte Ausgang sofort auf TRUE gesetzt.



Siehe auch:

- Tool-Befehle: Übersicht zu den Pulsausgabebefehlen (s. S. 1091)
- F166_PulseOutput_Set (s. S. 881)

Dieser Nicht-Inline-Befehl ist Teil der Tool-Befehle für die Pulsausgabe (s. S. 1091). Eine ausführliche Beschreibung der intern verwendeten Befehle finden Sie in der Online-Hilfe: F166_PulseOutput_Set (s. S. 881)

Um die Kombination von Kanal und Y-Ausgang zu prüfen, benötigt der Compiler das folgende Benennungsmuster für globale Variablen: %sPulse_TargetValueMatch_Kanal%d_Y%d_%s
Verwenden Sie für globale Variablen in Zählervergleichsfunktionen grundsätzlich dieses Muster:

- Channel%d muss ein Pulsausgabekanal sein, der in den Systemregistern aktiviert ist.
- Y%d muss eine explizite, von der SPS unterstützte Ausgangsadresse sein.

FP-Σ, FP0, FP-e:	Y0–Y7
FP-Σ (V3.1 oder neuer), FP0R:	Y0–Y1F
FP-X:	Y0–Y29F

- %s ist ein optionaler Deskriptor am Musteranfang
- %s ist ein optionaler Deskriptor am Musterende

Optional	Vorgegebenes Muster	Optional
g_b	Pulse_TargetValueMatch_ChannelA_Y11F	_MotorOn

Diese globale Variable erzeugt den Code für Kanal **A** und Ausgang **Y11F**.

SPS Typen

s. S. 1195

Datentypen

Eingangsvariable	Datentyp	Beschreibung
bExecute	BOOL	Eine steigende Flanke aktiviert die Funktion; Auswertung des Kontrollmerkers "Pulsausgabekanal" mit PulseInfo_IsTargetValueMatchActive (s. S. 1140)
iChannel	INT	FPΣ: 0, 2 FP-X R: 0, 1 FP-X 16K C14T: 0, 1, 2 FP-X 32K C30T, C60T: 0, 1, 2, 3 FP0R: 0, 1, 2, 3 FP0: 0, 1 FP-e: 0, 1
pYOutput	POINTER	Durch GetPointer erhaltenes Zeigerergebnis von einer globalen Variable, die Kanalnummer und Ausgang liefert.

diTargetValue	DINT	Geben Sie für den Sollwert einen 32-Bit-Datenwert innerhalb des folgenden Bereichs ein: FP0, FP-e: -838808–+8388607 FPΣ, FP-X, FP0R: -2147483467–+2147483648
Ausgangsvariable	**Datentyp**	**Beschreibung**
bError	BOOL	TRUE, wenn die Kombination von **Channel%d** und **pYOuput.iOffset** nicht mit einer gültigen Kombination von Kanalnummer und Ausgang übereinstimmt, wie sie von der globalen Variablen vorgegeben wird.

Beispiel In diesem Beispiel wird der Befehl im Kontaktplan (KOP) und im strukturierten Text (ST) verwendet. (ST).Für beide Programmiersprachen wird der gleiche POE-Kopf verwendet.

GVL In der globalen Variablenliste können Sie Variablen festlegen, die von allen POEs des Projekts verwendet werden können.

Globale Variablen*

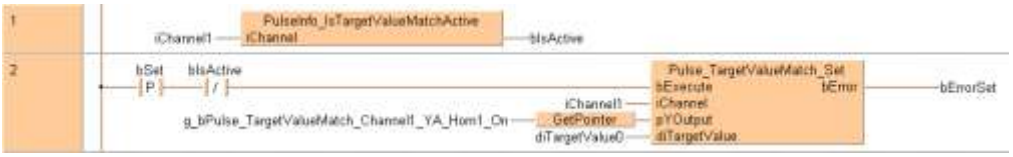
	Klasse	Bezeichner	FP-Adresse	IEC-Adresse	Typ	Initial
0	VAR_GLOBAL	g_bPulse_TargetValueMatch_Channel1_YA_Horn1_On	YA	%QX0.10	BOOL	FALSE

POE-Kopf Im POE-Kopf werden alle Ein- und Ausgangsvariablen deklariert, die für die Programmierung der Funktion verwendet werden.

Klasse	Bezeichner	Typ	Initial
VAR_EXTERNAL	g_bPulse_TargetValueMatch_Channel1_YA_Horn1_On	BOOL	FALSE
VAR	diTargetValue0	DINT	11000
VAR	iChannel1	INT	1
VAR	diTargetValue1	DINT	22000
VAR	bReset	BOOL	FALSE
VAR	bIsActive	BOOL	FALSE
VAR	bErrorReset	BOOL	FALSE
VAR	bErrorSet	BOOL	FALSE
VAR	bSet	BOOL	FALSE

Rumpf Mit PulseInfo_IsTargetValueMatchActive (s. S. 1140) ermitteln Sie, ob der durch **iChannel1** festgelegte Kanal aktiv ist. Wenn eine steigende Flanke bei **bSet** erkannt wird, und wenn **bIsActive nicht** TRUE ist, wird der Befehl ausgeführt. Die Kombination aus Kanalnummer und Ausgangskontakt wird in der globalen Variablen **g_bPulse_TargetValueMatch_Channel1_YA_Horn1_On** überprüft. Wenn die Pulsausgabe an Kanal 1 den Sollwert **diTargetValue0** erreicht, wird der Ausgang **YA** auf TRUE gesetzt.

KOP



ST

```
bIsActive:=PulseInfo_IsTargetValueMatchActive(iChannel1);
Pulse_TargetValueMatch_Set(bExecute := DF(bSet) AND NOT bIsActive,
                iChannel := iChannel1,
                pYOutput :=
GetPointer(g_bPulse_TargetValueMatch_Channel1_YA_Horn1_On),
                diTargetValue := diTargetValue0,
                bError => bErrorSet);
```

Kapitel 39

Anhang Programmierinformationen

39.1 FP TOOL Library

Die FP TOOL Library vereinfacht die Entwicklung von universellen, auf allen Panasonic-Steuerungen verwendbaren Programmen. Sie stellt Funktionen für den Zugriff auf die SPS-Speicherbereiche, Funktionen, die Informationen über Variablen liefern, und spezielle Kopierfunktionen zur Verfügung. Nachstehend finden Sie eine Auswahl dieser Funktionen. Detaillierte Informationen und Beispiele, siehe Online-Hilfe.

Das Programm kann ungünstig beeinflusst werden! Diese Funktionen können durch Zugriff auf falsche Speicherbereiche massive Probleme bereiten, wenn sie nicht entsprechend ihrer Bedeutung eingesetzt werden. Insbesondere können auch andere Programmteile davon beeinträchtigt werden.

Name	Funktion
Adressfunktionen	
Adr_Of_Var	Adresse einer Variablen am Eingang/Ausgang einer FP-Funktion
AdrLast_Of_Var	Adresse einer Variablen am Eingang/Ausgang einer FP-Funktion
Adr_Of_VarOffs	Adresse einer Variablen mit Offset am Eingang/Ausgang einer FP-Funktion
Zeigerfunktionen	
AdrDT_Of_Offs	DT-Adresse aus Adressoffset für den Eingang/Ausgang einer FP-Funktion
AdrFL_Of_Offs_l	FL-Adresse aus Adressoffset für den Eingang/Ausgang einer FP-Funktion
AreaOffs_OfVar	Liefert Speicherbereich und Adressoffset einer Variablen (mit Enable)
Is_AreaDT	Liefert TRUE wenn Speicherbereich einer Variablen DT-Bereich ist (mit Enable)
Is_AreaFL	Liefert TRUE wenn Speicherbereich einer Variablen FL-Bereich ist (mit Enable)
AreaOffs_ToVar	Kopiert den Inhalt einer Adresse, die durch Speicherbereich und Adressoffset spezifiziert ist, auf eine Variable (mit Enable)
Var_ToAreaOffs	Kopiert den Wert einer Variablen auf eine Adresse, die durch Speicherbereich und Adressoffset spezifiziert ist (mit Enable)
Funktionen, die Informationen über Variablen liefern	
Size_Of_Var	Liefert Größe einer Variablen in Worten (mit Enable)
Elem_OfArray1D	Liefert Anzahl der Elemente eines Arrays (mit Enable)
Elem_OfArray2D	Liefert Anzahl der Elemente der 1. und 2. Dimension eines Arrays (mit Enable)
Elem_OfArray3D	Liefert Anzahl der Elemente der 1., 2. und 3. Dimension eines Arrays (mit Enable)
Zusätzliche Kopierfunktionen	
	Diese Funktionen können aus Kompatibilitätsgründen noch übersetzt, jedoch nicht mehr neu über die OP/FUN/FB-Auswahl eingefügt werden.
Any16_ToBool16	Diese Funktion wird ab Version 5 von der Funktion INT_TO_BOOL16 oder WORD_TO_BOOL16 ersetzt.
Bool16_ToAny16	Diese Funktion wird ab Version 5 von der Funktion BOOL16_TO_INT oder BOOL16_TO_WORD ersetzt.
Any32_ToBool32	Diese Funktion wird ab Version 5 von der Funktion DINT_TO_BOOL32 oder DWORD_TO_BOOL32 ersetzt.
Bool32_ToAny32	Diese Funktion wird ab Version 5 von der Funktion BOOL32_TO_DINT oder BOOL32_TO_DWORD ersetzt.
Any16_ToSpecDT	Diese Funktion wird ab Version 5 von der Funktion INT_TO_SDT oder WORD_TO_SDT ersetzt.

SpecDT_ToAny16	Diese Funktion wird ab Version 5 von der Funktion SDT_TO_INT oder SDT_TO_WORD ersetzt.
Any32_ToSpecDT	Diese Funktion wird ab Version 5 von der Funktion DINT_TO_SDDT oder DWORD_TO_SDDT ersetzt.
SpecDT_ToAny32	Diese Funktion wird ab Version 5 von der Funktion SDDT_TO_DINT oder SDDT_TO_DWORD ersetzt.
Steuerfunktionen für Ablaufsprachenprogramme	
Funktionen, die alle Ablaufsprachenprogramme (AS) gleichzeitig steuern	
StartStopAllSfcs	Die Funktion hält alle Ablaufsprachenprogramme an und startet sie neu.
StartStopAllSfcsAndInitData	
Eine Funktion, die den Zustand aller Ablaufsprachenprogramme liefert	
AllSfcsStopped	Diese Funktion zeigt an, ob alle AS-Programme angehalten wurden.
Funktionen, die ein bestimmtes AS-Programm steuern	
StartStopSfc	Die Funktion hält ein bestimmtes Ablaufsprachenprogramm (AS) an und startet es neu.
StartStopSfcAndInitData	
ControlSfc	Die Funktion steuert ein bestimmtes Ablaufsprachenprogramm (AS).
ControlSfcAndData	
ActivateStepsOfStoppedSfc	Diese Funktion führt ein unterbrochenes Ablaufsprachenprogramm (AS) wieder aus.
Funktionen, die die Zustände eines bestimmten AS-Programms liefern	
SfcStopped	Diese Funktion zeigt an, ob ein bestimmtes AS-Programm angehalten wurde.
SfcTransitionsInhibited	Zeigt an, ob die Transitionen eines bestimmten Ablaufsprachenprogramms (AS) gesperrt wurden.
SfcRunning	Diese Funktion zeigt an, ob ein bestimmtes AS-Programm angehalten wurde.
SfcOutputsReset	Diese Funktion zeigt, ob die Eingänge eines bestimmten Ablaufsprachenprogramm zurückgesetzt worden sind.

39.2 Befehle für Fließkommawerte

Die FP-Befehle für Fließkommawerte sind insbesondere für Anwendungen entwickelt worden, die Variablen des Datentyps REAL erfordern. Die meisten dieser Befehle lassen sich durch die etwas flexibleren IEC-Befehle ersetzen. Dadurch lässt sich auch die Anzahl der Befehle reduzieren, die Sie kennen sollten.

Die folgenden Befehle für Fließkommawerte werden in diesem Handbuch detailliert beschrieben, weil sie sich nicht einfach durch IEC-Befehle ersetzen lassen: F327_INT (s. S. 665), F328_DINT (s. S. 667), F333_FINT (s. S. 669), F334_FRINT (s. S. 671), F335_FSIGN (s. S. 673), F337_RAD (s. S. 675) und F338_DEG (s. S. 677).

Weitere Informationen und Beispiele zu den anderen Befehlen für Fließkommawerte finden Sie in der Online-Hilfe. Eine schnelle Referenz finden Sie in der nachstehenden Tabelle.

Name	Funktion	Äquivalente IEC-Funktion
F309_FMV	Konstanter Transfer von Fließkommawerten	E_MOVE
F310_FADD	Addition von Fließkommawerten	E_ADD
F311_FSUB	Subtraktion von Fließkommawerten	E_SUB
F312_FMUL	Multiplikation von Fließkommawerten	E_MUL
F313_FDIV	Division von Fließkommawerten	E_DIV
F314_FSIN	Sinus von Fließkommawerten	E_SIN
F315_FCOS	Cosinus von Fließkommawerten	E_COS
F316_FTAN	Tangens von Fließkommawerten	E_TAN
F317_ASIN	Arkussinus von Fließkommawerten	E_ASIN
F318_ACOS	Arkuscosinus von Fließkommawerten	E_ACOS
F319_ATAN	Arkustangens von Fließkommawerten	E_ATAN
F320_LN	Natürlicher Logarithmus von Fließkommawerten	E_LN
F321_EXP	Exponent von Fließkommawerten	E_EXP
F322_LOG	Logarithmus von Fließkommawerten	E_LOG
F323_PWR	Potenzieren von Fließkommawerten	E_EXPT
F324_FSQR	Quadratwurzel aus Fließkommawerten	E_SQRT
F325_FLT	16-Bit-Ganzzahl-Wert → Fließkommawert	E_INT_TO_REAL
F326_DFLT	32-Bit-Ganzzahl-Wert → Fließkommawert	E_DINT_TO_REAL
F329_FIX	Fließkommawert → 16-Bit-Ganzzahlwert Erste Nachkommastelle wird abgerundet	E_TRUNC_TO_INT
F330_DFIX	Fließkommawert → 32-Bit-Ganzzahlwert Erste Nachkommastelle wird abgerundet	E_TRUNC_TO_DINT
F331_ROFF	Fließkommawert → 16-Bit-Ganzzahlwert Erste Nachkommastelle wird abgerundet	E_REAL_TO_INT
F332_DROFF	Fließkommawert → 32-Bit-Ganzzahlwert Erste Nachkommastelle wird abgerundet	E_REAL_TO_DINT
F336_FABS	Absoluter Fließkommawert	E_ABS
F345_FCMP	Vergleich von Fließkommawerten	E_GE, E_GT, E_EQ, E_LE, E_LT, E_NE
F347_FLIMT	Begrenzer mit Fließkommawerten	E_LIMIT

39.3 Indexregister

Auch die Indexregister werden zum Lesen und Schreiben von 16-Bit-Daten verwendet. Es gibt sieben 16-Bit Register (IX, IY, IZ bis ID). In den Indexregistern können Speicherbereichsnummern indirekt angegeben werden. Wenn Sie eine Adresse mit einem Indexregisterwert verändern, heißt dies “Index-Modifizierung.”

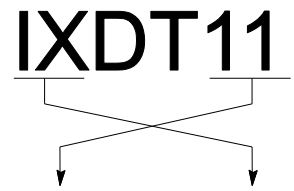
In FPWIN Pro kann der Benutzer nur auf IX und IY zugreifen. Die Indexregister IZ bis ID sind dem System für Arrayberechnungen, FB-Indizierung oder die Implementierung bestimmter FP-Befehle vorbehalten. (siehe Hinweis)

Bei der Index-Modifizierung ist zu beachten:

- Es können Konflikte durch gemeinsame Benutzung von Indexregistern durch den Anwender und das System auftreten, die nicht gemeldet werden. Generell sollte man also Indexregister vermeiden und stattdessen Array–Strukturen verwenden.
- Ein Indexregister kann nicht durch ein anderes Indexregister modifiziert werden.
- In FPWIN Pro kann ein Indexregister keinen konstanten Wert modifizieren.

Beispiel

Modifizierung einer Speicherbereichsadresse.
Adresse = Basisadresse + Wert in IX, IY, IZ bis ID
Modifizierung von DT11



Base address		IX value		Target address
11	+	0	=	DT11
11	+	10	=	DT21
11	+	-10	=	DT1

Wenn Sie das Indexregister benutzen, um eine Adresse zu modifizieren, stellen Sie sicher, daß die verschobene Adresse die letzte Adresse des Speicherbereichs nicht überschreitet. Wenn die verschobene Adresse ihre letzte Adresse überschreitet, tritt ein Betriebsfehler auf.

Wenn eine 32-Bit Konstante modifiziert wird, werden die spezifizierte Registernummer und die folgende Registernummer zusammen verwendet, um die Daten als 32-Bit Daten zu verarbeiten. Wenn Sie eine 32-Bit Konstante modifizieren, geben Sie die ID nicht an.

☞ **Es ist unbedingt zu empfehlen, für die Bearbeitung von Speicherbereichen Arrays anstelle von Indexregistern zu verwenden, andernfalls könnte es zu Konflikten kommen, wenn Benutzer und System auf dasselbe Indexregister zugreifen.**

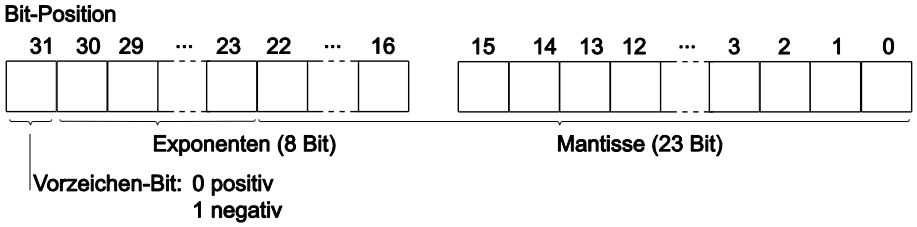
Weitere Informationen zur Verwendung von Indexregistern finden Sie im Handbuch "Programmable Controller FP10SH Programming" Manual (ACG-M0081-1).

39.4 Reelle Zahlen

Bei manchen Befehlen der FP10SH und FP2 Serien können Sie mit reellen Zahlen arbeiten. Zu den verfügbaren Arten von reellen Zahlen gehören: Fließkomma- und BCD-Konstanten.

39.4.1 Fließkomma-Konstante (f)

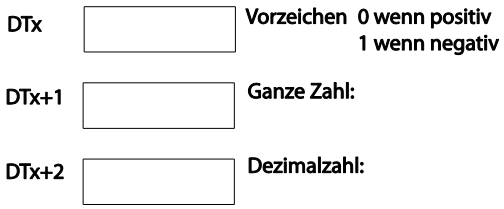
Fließkomma-Konstanten bestehen aus zwei Wörtern, die mit der einfachen Genauigkeit der Fließkomma-Logik verarbeitet werden. Es gibt sieben effektive Zahlen. Die Mantisse hat 23 Bit und der Exponent 8 Bit. (Basiert auf IEEE754)



Sie können die folgenden Zahlen verwenden: $\pm(1{,}175494 \times 10^{-38}$ bis $3{,}402823 \times 10^{38})$.

39.4.2 BCD-Konstanten

Die BCD Fließkomma-Konstanten werden, wie unten dargestellt, in Form von drei Worten verarbeitet.



Beispiel

Sie können die folgenden Zahlen verwenden: -9999,9999 bis 9999,9999

Die Hauptbefehle, die den Gebrauch von BCD-Konstanten erlauben, sind:

- **F300** BSIN Sinus-Funktion mit BCD-codierten Werten
- **F301** BCOS Cosinus-Funktion mit BCD-codierten Werten
- **F302** BTAN Tangens-Funktion mit BCD-codierten Werten
- **F303** BASIN Arcus-Sinus-Funktion mit BCD-codierten Werten
- **F304** BACOS Arcus-Cosinus-Funktion mit BCD-codierten Werten
- **F305** BACOS Arcus-Tangens-Funktion mit BCD-codierten Werten

In FPWIN Pro müssen Sie Werte in hexadezimaler Form (16#) eingeben, um eine BCD-Konstante zu spezifizieren. Dieser Datentyp wird als ARRAY [0..2] OF WORD implementiert.

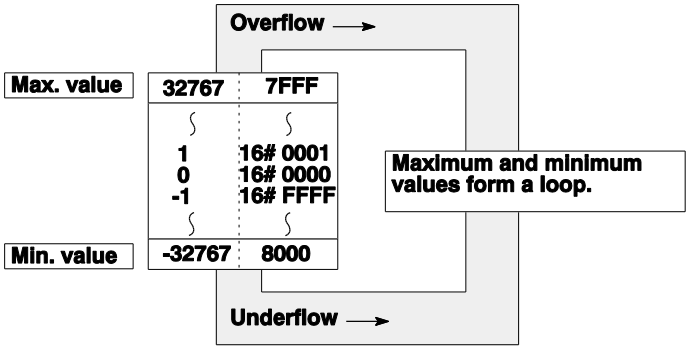
39.5 Über- und Unterlauf

Bei der Ausführung eines Befehls wird der zugelassene Wertbereich gelegentlich überschritten, bzw. unterschritten. Eine Überschreitung des maximalen Wertes wird als “Überlauf” bezeichnet. Wird der minimale Wert unterschritten, heißt dies “Unterlauf”. Wenn das Ergebnis zu einem Überlauf oder Unterlauf führt, schaltet sich R9009s Carry-Flag an.

39.5.1 Werte beim Überlauf/Unterlauf

Alle maximalen und minimalen-Werte, die die FP-SPS bearbeiten, bilden eine Schleife, wie nachstehend gezeigt wird.

Binäre 16-Bit Verarbeitung



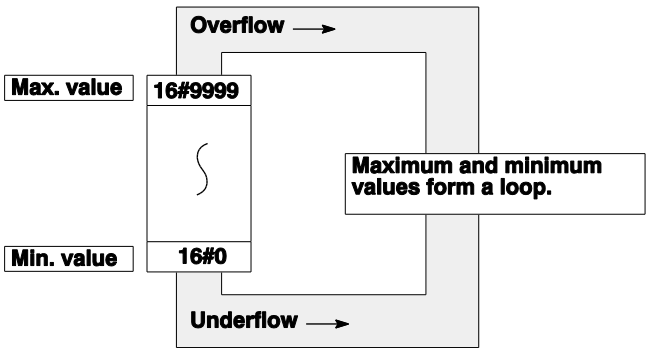
Beispiel 1: 32767 + 1 (Überlauf)

Das Ergebnis ist 32768 und das Carry-Flag schaltet sich an.

Beispiel 2: -32768 - 1 (Unterlauf)

Das Ergebnis ist 32767 und das Carry-Flag schaltet sich an.

BCD 4-Ganzzahlverarbeitung



Beispiel 1: 16#9999+ 16#1 (Überlauf)

Das Verarbeitungsergebnis ist 16#0, und das Carry-Flag schaltet sich an.

Beispiel 2: 16#0 - 16#1 (Unterlauf)

Das Verarbeitungsergebnis ist 16#9999, und das Carry-Flag schaltet sich an.

39.5.2 Umwandlungstabelle

Dezimalzahl	Binär	BCD (Binary Coded Decimal)	Gray Code
0	0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
1	0000 0000 0000 0001	0000 0000 0000 0001	0000 0000 0000 0001
2	0000 0000 0000 0010	0000 0000 0000 0010	0000 0000 0000 0011
3	0000 0000 0000 0011	0000 0000 0000 0011	0000 0000 0000 0010
4	0000 0000 0000 0100	0000 0000 0000 0100	0000 0000 0000 0110
5	0000 0000 0000 0101	0000 0000 0000 0101	0000 0000 0000 0111
6	0000 0000 0000 0110	0000 0000 0000 0110	0000 0000 0000 0101
7	0000 0000 0000 0111	0000 0000 0000 0111	0000 0000 0000 0100
8	0000 0000 0000 1000	0000 0000 0000 1000	0000 0000 0000 1100
9	0000 0000 0000 1001	0000 0000 0000 1001	0000 0000 0000 1101
10	0000 0000 0000 1010	0000 0000 0001 0000	0000 0000 0000 1111
11	0000 0000 0000 1011	0000 0000 0001 0001	0000 0000 0000 1110
12	0000 0000 0000 1100	0000 0000 0001 0010	0000 0000 0000 1010
13	0000 0000 0000 1101	0000 0000 0001 0011	0000 0000 0000 1011
14	0000 0000 0000 1110	0000 0000 0001 0100	0000 0000 0000 1001
15	0000 0000 0000 1111	0000 0000 0001 0101	0000 0000 0000 1000
16	0000 0000 0001 0000	0000 0000 0001 0110	0000 0000 0001 1000
17	0000 0000 0001 0001	0000 0000 0001 0111	0000 0000 0001 1001
18	0000 0000 0001 0010	0000 0000 0001 1000	0000 0000 0001 1011
19	0000 0000 0001 0011	0000 0000 0001 1001	0000 0000 0001 1010
20	0000 0000 0001 0100	0000 0000 0010 0000	0000 0000 0001 1110
21	0000 0000 0001 0101	0000 0000 0010 0001	0000 0000 0001 1111
22	0000 0000 0001 0110	0000 0000 0010 0010	0000 0000 0001 1101
23	0000 0000 0001 0111	0000 0000 0010 0011	0000 0000 0001 1100
24	0000 0000 0001 1000	0000 0000 0010 0100	0000 0000 0001 0100
25	0000 0000 0001 1001	0000 0000 0010 0101	0000 0000 0001 0101
26	0000 0000 0001 1010	0000 0000 0010 0110	0000 0000 0001 0111
27	0000 0000 0001 1011	0000 0000 0010 0111	0000 0000 0001 0110
28	0000 0000 0001 1100	0000 0000 0010 1000	0000 0000 0001 0010
29	0000 0000 0001 1101	0000 0000 0010 1001	0000 0000 0001 0011
30	0000 0000 0001 1110	0000 0000 0011 0000	0000 0000 0001 0001
31	0000 0000 0001 1111	0000 0000 0011 0001	0000 0000 0001 0000
32	0000 0000 0010 0000	0000 0000 0011 0010	0000 0000 0011 0000
...	...	...	...
63	0000 0000 0011 1111	0000 0000 0110 0011	0000 0000 0010 0000
64	0000 0000 0100 0000	0000 0000 0110 0100	0000 0000 0110 0000
...	...	...	...
255	0000 0000 1111 1111	0000 0010 0101 0101	0000 0000 1000 0000

39.6 Sonderdatenregister

Verwenden Sie die SPS-unabhängigen Systemvariablen für den Zugriff auf Sonderdatenregister und Sondermerker.

◆REFERENZ

Hinweise zur Verwendung von Systemvariablen finden Sie in der Online-Hilfe von FPWIN Pro.

39.7 Bitmerker und Speicherbereiche

39.7.1 Bitmerker und Speicherbereiche für FP0

Merker [Bits]

Typ	Speicher größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge	208	X0–X12F	%IX0.0–%IX12.15	Liefert den Zustand eines externen Eingangs.
Externe Ausgänge	208	Y0–Y12F	%QX0.0–%QX12.15	Ansteuerung eines externen Ausgangs.
Interne Merker 1)	1008	R0–R62F	%MX0.0.0–%MX0.62.15	Zum Speichern von Bitinformationen im SPS-Programm.
Zeitgeber 1) 2)	144	T0–T99/ C100–C143	%MX1.0–%MX1.99/ %MX2.100–%MX2.143	Wird nur intern verwendet. Ausgangskontakt eines TM-Befehls.
Zähler 1) 2)	144	C100–C143/ T0–T99	%MX2.100–%MX2.143/ %MX1.0–%MX1.99	Wird nur intern verwendet. Ausgangskontakt eines CT-Befehls.
Sondermerker	64	R9000–R903F	%MX0.900.0–%MX0.903.15	Zustand wechselt je nach Bedingung. Wird intern als Merker verwendet.

Speicherbereich [Worte]

Typ		Speicher größe	Verfügbarer Adressbereich		Beschreibung
			F/P	IEC	
Externe Eingänge		13	WX0–WX12	%IW0–%IW12	Liefert den Zustand von 16 externen Eingängen in einem Wort (16 Bit).
Externe Ausgänge		13	WY0–WY12	%QW0–%QW12	Liefert den Zustand von 16 externen Ausgängen in einem Wort (16 Bit).
Interne Merker 1)		63	WR0–WR62	%MW0.0–%MW0.62	Für den wortweisen Zugriff auf 16 interne Merker.
Daten-register 1)	**C10/C14/C16**	1660	DT0–DT1659	%MW5.0–%MW5.1659	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
	C32/SL1	6144	DT0–DT6143	%MW5.0–%MW5.6143	
	T32C	16384	DT0–DT16383	%MW5.0–%MW5.16383	
Sollwerte für Zeitgeber/Zähler 1) 2)		144	SV0–SV143	%MW3.0–%MW3.143	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 1) 2)		144	EV0–EV143	%MW4.0–%MW4.143	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
Sonderdatenregister		112	DT90000–DT90111	%MW5.90000–%MW5.90111	Datenspeicher für Einstellungen und Fehlercodes.
Indexregister		2	IX, IY	%MW6.0–%MW6.1	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

Speicherbereich [Doppelworte]

Typ		Speicher größe	Verfügbarer Adressbereich		Beschreibung
			F/P	IEC	
Externe Eingänge		6	DWX0–DWX11	%ID0– %ID11	Für den doppelwortweisen Zugriff auf 32 externe Eingänge.
Externe Ausgänge		6	DWY0–DWY11	%QD0– %QD11	Für den doppelwortweisen Zugriff auf 32 externe Ausgänge.
Interne Merker [1]		31	DWR0–DWR61	%MD0.0– %MD0.61	Für den doppelwortweisen Zugriff auf 32 interne Merker.
Daten-register [1]	**C10/C14/C 16**	830	DDT0– DDT1658	%MD5.0– %MD5.1658	Vom Programm verwendeter Datenspeicher. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
	C32/SL1	3072	DDT0– DDT6142	%MD5.0– %MD5.6142	
	T32C	8192	DDT0– DDT16382	%MD5.0– %MD5.16382	
Sollwerte für Zeitgeber/Zähler [1] [2]		72	DSV0–DSV142	%MD3.0– %MD3.142	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler [1] [2]		72	DEV0–DEV142	%MD4.0– %MD4.142	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
Sonderdatenregister		56	DDT90000– DDT90110	%MD5.90000– %MD5.90110	Datenspeicher für Einstellungen und Fehlercodes.
Indexregister		1	DI0	%MD6.0	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

1) Der Speicherinhalt kann entweder selbsthaltend oder nicht selbsthaltend sein. Ist er selbsthaltend, geht er beim Ausschalten des Geräts oder beim Umschalten vom RUN- in den PROG-Modus nicht verloren. Ist er nicht selbsthaltend, wird der Speicher gelöscht. FP0 T32C: Die Einstellung der selbsthaltenden und nicht selbsthaltenden Bereiche wird in den Systemregistern vorgenommen. Alle anderen CPU-Typen: Die Speicherbereiche sind fest in selbsthaltende und nicht selbsthaltende Bereiche unterteilt (siehe unten).

2) Die Anzahl der Zeitgeber- und Zählerkontakte kann in Systemregister 5 geändert werden. Die Zahlen in der Tabelle entsprechen den Standardeinstellungen.

Selbsthaltende und nicht selbsthaltende Speicherbereiche:

Speicherbereich		C10/C14/C16	C32
Zeitgeber		Nicht selbsthaltend: Alle	
Zähler	**Nicht selbsthaltend**	Ab dem angegebenen Wert bis C139	Ab dem angegebenen Wert bis C127
	Selbsthaltend	4 Kontakte (Istwerte) (C140–C143)	16 Kontakte (Istwerte) (C128–C143)
Interne Merker	**Nicht selbsthaltend**	976 Kontakte (R0–R60F) 61 Worte (WR0–WR60)	880 Kontakte (R0–R54F) 55 Worte (WR0–WR54)
	Selbsthaltend	32 Kontakte (R610–R62F) 2 Worte (WR61–WR62)	128 Kontakte (R550–R62F) 8 Worte (WR55–WR62)
Daten-register	**Nicht selbsthaltend**	1652 Worte (DT0–DT1651)	6112 Worte (DT0–DT6111)
	Selbsthaltend	8 Worte (DT1652–DT1659)	32 Worte (DT6112–DT6143)

39.7.2 Bitmerker und Speicherbereiche

Merker [Bits]

Typ	Speicher-größe	Verfügbarer Adressbereich		Funktion
		F/P	IEC	
Externe Eingänge 1)	1760	X0–X109F	%IX0.0– %IX109.15	Liefert den Zustand eines externen Eingangs.
Externe Ausgänge 1)	1760	Y0–Y109F	%QX0.0– %QX109.15	Ansteuerung eines externen Ausgangs.
Interne Merker 2)	4096	R0–R255F	%MX0.0.0– %MX0.255.15	Zum Speichern von Bitinformationen im SPS-Programm.
Koppelmerker 2)	2048	L0–L127F	%MX7.0.0– %MX7.127.15	Gemeinsam genutzter Merker bei SPS-Kopplung.
Zeitgeber 2) 3)	1024	T0–T1007/ C1008-C1023	%MX1.0– %MX1.1007/ %MX2.1008– %MX2.1023	Wird nur intern verwendet. Ausgangskontakt eines TM-Befehls.
Zähler 2) 3)	1024	C1008–C1023/ T0–T1007	%MX2.1008– %MX2.1023/ %MX1.0– %MX1.1007	Wird nur intern verwendet. Ausgangskontakt eines CT-Befehls.
Sondermerker	224	R9000–R913F	%MX0.900.0– %MX0.913.15	Zustand wechselt je nach Bedingung. Wird intern als Merker verwendet.

Speicherbereich [Worte]

Typ		Speicher-größe	Verfügbarer Adressbereich		Funktion
			F/P	IEC	
Externe Eingänge [1]		110	WX0–WX109	%IW0– %IW109	Liefert den Zustand von 16 externen Eingängen in einem Wort (16 Bit).
Externe Ausgänge [1]		110	WY0–WY109	%QW0– %QW109	Liefert den Zustand von 16 externen Ausgängen in einem Wort (16 Bit).
Interne Merker [2]		256	WR0–WR255	%MW0.0– %MW0.255	Für den wortweisen Zugriff auf 16 interne Merker.
Koppelmerker		128	WL0–WL127	%MW7.0– %MW7.127	Für den wortweisen Zugriff auf 16 Koppelmerker.
Daten-register [2]	**C10, C14, C16**	12315	DT0–DT12312	%MW5.0– %MW5.12312	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
	C32, T32, F32	32765	DT0–DT32762	%MW5.0– %MW5.32762	
Koppeldatenregister [2]		256	LD0–LD255	%MW8.0– %MW8.255	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden wortweise (16 Bit) verarbeitet..
Sollwerte für Zeitgeber/Zähler [2]		1024	SV0–SV1023	%MW3.0– %MW3.1023	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler [2]		1024	EV0–EV1023	%MW4.0– %MW4.1023	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
Sonderdatenregister		440	DT90000– DT90439	%MW5.90000– %MW5.90439	Datenspeicher für Einstellungen und Fehlercodes.

Speicherbereich [Doppelworte]

Typ		Speicher-größe	Verfügbarer Adressbereich		Funktion
			F/P	IEC	
Externe Eingänge [1]		55	DWX0–DWX108	%ID0– %ID108	Für den doppelwortweisen Zugriff auf 32 externe Eingänge.
Externe Ausgänge [1]		55	DWY0–DWY108	%QD0– %QD108	Für den doppelwortweisen Zugriff auf 32 externe Ausgänge.
Interne Merker [2]		128	DWR0– DWR254	%MD0.0– %MD0.254	Für den doppelwortweisen Zugriff auf 32 interne Merker.
Koppelmerker		64	DWL0–DWL126	%MD7.0– %MD7.126	Für den doppelwortweisen Zugriff auf 32 Koppelmerker.
Daten-register [2]	**C10, C14, C16**	6157	DDT0– DDT12311	%MD5.0– %MD5.12311	Vom Programm verwendeter Datenspeicher. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
	C32, T32, F32	16382	DDT0– DDT32761	%MD5.0– %MD5.32761	
Koppeldatenregister [2]		128	DLD0–DLD126	%MD8.0– %MD8.126	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Sollwerte für Zeitgeber/Zähler [2]		512	DSV0–DSV1022	%MD3.0– %MD3.1022	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.

Typ	Speicher-größe	Verfügbarer Adressbereich		Funktion
		F/P	IEC	
Istwerte für Zeitgeber/Zähler 2)	512	DEV0–DEV1022	%MD4.0– %MD4.1022	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
Sonderdatenregister	220	DDT90000– DDT90438	%MD5.90000– %MD5.90438	Datenspeicher für Einstellungen und Fehlercodes.

1) Die angegebene Zahl der Eingangskontakte ist die für den internen Speicher reservierte Zahl. Die tatsächliche Anzahl wird durch die Hardware-Konfiguration bestimmt.

2) Es gibt selbsthaltende und nicht selbsthaltende Speicherbereiche. Selbsthaltende Bereiche werden im Gegensatz zu den nicht selbsthaltenden Bereichen beim Abschalten der Steuerung oder beim Umschalten vom RUN- in den PROG-Modus gespeichert.

C10/C14/C16/C32:
Die selbsthaltenden und nicht selbsthaltenden Bereiche sind nicht veränderbar. Angaben zur Größe der einzelnen Bereiche finden Sie in den Leistungsdaten.

T32/F32:
Die selbsthaltenden und nicht selbsthaltenden Bereiche können in den Systemregistern verändert werden.

T32:
Die Funktion der Selbsthaltebereiche ist nicht mehr garantiert, wenn zusätzliche Bereiche definiert wurden, die Batterie jedoch entladen ist. Die Daten können undefinierte Werte annehmen. Sie werden auf 0 zurück gesetzt, wenn die Steuerung das nächste Mal eingeschaltet wird. Siehe.

3) Die Anzahl der Zeitgeber- und Zählerkontakte kann in Systemregister 5 geändert werden. Die Zahlen in der Tabelle entsprechen den Standardeinstellungen.

39.7.3 Bitmerker und Speicherbereiche

Merker [Bits]

Typ	Speicher-größe	Verfügbarer Adressbereich		Funktion
		FP	IEC	
Externe Eingänge 1)	1184	X0–X73F	%IX0.0– %IX73.15	Liefert den Zustand eines externen Eingangs.
Externe Ausgänge 1)	1184	Y0–Y73F	%QX0.0– %QX73.15	Ansteuerung eines externen Ausgangs.
Interne Merker 2)	4096	R0–R255F	%MX0.0– %MX0.255.15	Zum Speichern von Bitinformationen im SPS-Programm.
Koppelmerker 2)	2048	L0–L127F	%MX7.0.0– %MX7.63.15	Gemeinsam genutzter Merker bei SPS-Kopplung.
Zeitgeber 2) 3)	1024	T0–T1007/ C1008–C1023	%MX1.0– %MX1.1007/ %MX2.1008– %MX2.1023	Wird nur intern verwendet. Ausgangskontakt eines TM-Befehls.
Zähler 2) 3)	1024	C1008–C1023/ T0–T1007	%MX2.1008– %MX2.1023/ %MX1.0– %MX1.1007	Wird nur intern verwendet. Ausgangskontakt eines CT-Befehls.
Sondermerker	176	R9000–R910F	%MX0.900.0– %MX0.910.15	Zustand wechselt je nach Bedingung. Wird intern als Merker verwendet.

Speicherbereich [Worte]

Typ	Speicher-größe	Verfügbarer Adressbereich		Funktion
		FP	IEC	
Externe Eingänge 1)	74	WX0–WX73	%IW0– %IW73	Liefert den Zustand von 16 externen Eingängen in einem Wort (16 Bit).
Externe Ausgänge 1)	74	WY0–WY73	%QW0– %QW73	Liefert den Zustand von 16 externen Ausgängen in einem Wort (16 Bit).
Interne Merker 2)	256	WR0–WR255	%MW0.0– %MW0.255	Für den wortweisen Zugriff auf 16 interne Merker.
Koppelmerker	128	WL0–WL127	%MW7.0– %MW7.127	Für den wortweisen Zugriff auf 16 Koppelmerker.
Datenregister 2)	32763	DT0–DT32762	%MW5.0– %MW5.32762	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
Koppeldatenregister 2)	256	LD0–LD255	%MW8.0– %MW8.255	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden wortweise (16 Bit) verarbeitet.
Sollwerte für Zeitgeber/Zähler 2)	1024	SV0–SV1023	%MW3.0– %MW3.1023	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 2)	1024	EV0–EV1023	%MW4.0– %MW4.1023	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
Sonderdatenregister	260	DT90000– DT90259	%MW5.90000– %MW5.90259	Datenspeicher für Einstellungen und Fehlercodes.

Speicherbereich [Doppelworte]

Typ	Speicher-größe	Verfügbarer Adressbereich		Funktion
		FP	IEC	
Externe Eingänge 1)	37	DWX0–DWX72	%ID0– %ID72	Für den doppelwortweisen Zugriff auf 32 externe Eingänge.
Externe Ausgänge 1)	37	DWY0–DWY72	%QD0– %QD72	Für den doppelwortweisen Zugriff auf 32 externe Ausgänge.
Interne Merker 2)	128	DWR0–DWR254	%MD0.0– %MD0.254	Für den doppelwortweisen Zugriff auf 32 interne Merker.
Koppelmerker	64	DWL0–DWL126	%MD7.0– %MD7.126	Für den doppelwortweisen Zugriff auf 32 Koppelmerker.
Datenregister 2)	16382	DDT0– DDT32763	%MD5.0– %MD5.32763	Vom Programm verwendeter Datenspeicher. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Koppeldatenregister 2)	128	DLD0–DLD254	%MD8.0– %MD8.254	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Sollwerte für Zeitgeber/Zähler 2)	512	DSV0–DSV1022	%MD3.0– %MD3.1022	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 2)	512	DEV0–DEV1022	%MD4.0– %MD4.1022	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
Sonderdatenregister	130	DDT90000– DDT90258	%MD5.90000– %MD5.90258	Datenspeicher für Einstellungen und Fehlercodes.

1) Die angegebene Zahl der Eingangskontakte ist die für den internen Speicher reservierte Zahl. Die tatsächliche Anzahl wird durch die Hardware-Konfiguration bestimmt.

2) Ohne die optionale Pufferbatterie wird nur der feste Bereich gespeichert:
Zähler: 16 (C1008–C1023)
Interne Merker: 128 (R900–R97F)
Datenregister: DT32710–DT32764.
Mit Batterie ist eine Datensicherung der selbsthaltenden und nicht selbsthaltenden Bereiche in den Systemregistern möglich.
Die Funktion der Selbsthaltebereiche ist nicht mehr garantiert, wenn zusätzliche Bereiche definiert wurden, die Batterie jedoch entladen ist. Die Daten können undefinierte Werte annehmen. Sie werden nicht auf 0 zurück gesetzt, wenn die Steuerung das nächste Mal eingeschaltet wird. Überwachen Sie unbedingt den Ladezustand der Batterie. Wenn keine Batterie verwendet wird, müssen Sie die Selbsthaltebereiche auf die Werkseinstellungen zurücksetzen. Siehe.

3) Die Anzahl der Zeitgeber- und Zählerkontakte kann in Systemregister 5 geändert werden. Die Zahlen in der Tabelle entsprechen den Standardeinstellungen.

39.7.4 Bitmerker und Speicherbereiche für FP-X

Merker [Bits]

Typ	Speicher-größe	Verfügbarer Adressbereich		Funktion
		F/P	IEC	
Externe Eingänge 1)	1760	X0–X109F	%IX0.0– %IX109.15	Liefert den Zustand eines externen Eingangs.
Externe Ausgänge 1)	1760	Y0–Y109F	%QX0.0– %QX109.15	Ansteuerung eines externen Ausgangs.
Interne Merker 2)	4096	R0–R255F	%MX0.0.0– %MX0.255.15	Zum Speichern von Bitinformationen im SPS-Programm.
Koppelmerker 2)	2048	L0–L127F	%MX7.0.0– %MX7.127.15	Gemeinsam genutzter Merker bei SPS-Kopplung.
Zeitgeber 2) 3)	1024	T0–T1007/ C1008–C1023	%MX1.0– %MX1.1007/ %MX2.1008– %MX2.1023	Wird nur intern verwendet. Ausgangskontakt eines TM-Befehls.
Zähler 2) 3)	1024	C1008–C1023/ T0–T1007	%MX2.1008– %MX2.1023/ %MX1.0– %MX1.1007	Wird nur intern verwendet. Ausgangskontakt eines CT-Befehls.
Sondermerker	192	R9000–R911F	%MX0.900.0– %MX0.911.15	Zustand wechselt je nach Bedingung. Wird intern als Merker verwendet.

Speicherbereich [Worte]

Typ		Speicher-größe	Verfügbarer Adressbereich		Funktion
			F/P	IEC	
Externe Eingänge 1)		110	WX0–WX109	%IW0– %IW109	Liefert den Zustand von 16 externen Eingängen in einem Wort (16 Bit).
Externe Ausgänge 1)		110	WY0–WY109	%QW0– %QW109	Liefert den Zustand von 16 externen Ausgängen in einem Wort (16 Bit).
Interne Merker 2)		256	WR0–WR255	%MW0.0– %MW0.255	Für den wortweisen Zugriff auf 16 interne Merker.
Koppelmerker		128	WL0–WL127	%MW7.0– %MW7.127	Für den wortweisen Zugriff auf 16 Koppelmerker.
Daten-register 2)	**C14**	12285	DT0–DT12284	%MW5.0– %MW5.12284	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
	C30, C60	32765	DT0–DT32764	%MW5.0– %MW5.32764	

Typ	Speicher-größe	Verfügbarer Adressbereich		Funktion
		F/P	IEC	
Koppeldatenregister 2)	256	LD0–LD255	%MW8.0– %MW8.255	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden wortweise (16 Bit) verarbeitet.
Sollwerte für Zeitgeber/Zähler 2)	1024	SV0–SV1023	%MW3.0– %MW3.1023	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 2)	1024	EV0–EV1023	%MW4.0– %MW4.1023	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
Sonderdatenregister	374	DT90000– DT90373	%MW5.90000– %MW5.90373	Datenspeicher für Einstellungen und Fehlercodes.
Indexregister	14	I0–ID	%MW6.0– %MW6.13	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

Speicherbereich [Doppelworte]

Typ		Speicher größe	Verfügbarer Adressbereich		Funktion
			F/P	IEC	
Externe Eingänge 1)		55	DWX0–DWX108	%ID0– %ID108	Für den doppelwortweisen Zugriff auf 32 externe Eingänge.
Externe Ausgänge 1)		55	DWY0–DWY108	%QD0– %QD108	Für den doppelwortweisen Zugriff auf 32 externe Ausgänge.
Interne Merker 2)		128	DWR0–DWR254	%MD0.0– %MD0.254	Für den doppelwortweisen Zugriff auf 32 interne Merker.
Koppelmerker		64	DWL0–DWL126	%MD7.0– %MD7.126	Für den doppelwortweisen Zugriff auf 32 Koppelmerker.
Datenregister 2)	**C14**	6142	DDT0– DDT12283	%MD5.0– %MD5.12283	Vom Programm verwendeter Datenspeicher. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
	C30, C60	16382	DDT0– DDT32763	%MD5.0– %MD5.32763	
Koppeldatenregister 2)		128	DLD0–DLD126	%MD8.0– %MD8.126	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Sollwerte für Zeitgeber/Zähler 2)		512	DSV0–DSV1022	%MD3.0– %MD3.1022	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 2)		512	DEV0–DEV1022	%MD4.0– %MD4.1022	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
Sonderdatenregister		187	DDT90000– DDT90438	%MD5.90000– %MD5.90438	Datenspeicher für Einstellungen und Fehlercodes.
Indexregister		7	DI0–DIC	%MD6.0– %MD6.12	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

1) Die angegebene Zahl der Eingangskontakte ist die für den internen Speicher reservierte Zahl. Die tatsächliche Anzahl wird durch die Hardware-Konfiguration bestimmt.

2) Ohne die optionale Pufferbatterie wird nur der feste Bereich gespeichert:

Mit Batterie ist eine Datensicherung der selbsthaltenden und nicht selbsthaltenden Bereiche in den Systemregistern möglich.

Die Funktion der Selbsthaltebereiche ist nicht mehr garantiert, wenn zusätzliche Bereiche definiert wurden, die Batterie jedoch entladen ist. Die Daten können undefinierte Werte annehmen. Sie werden nicht auf 0 zurück gesetzt, wenn die Steuerung das nächste Mal eingeschaltet wird. Überwachen Sie unbedingt den Ladezustand der Batterie. Wenn keine Batterie verwendet wird, müssen Sie die Selbsthaltebereiche auf die Werkseinstellungen zurücksetzen. Siehe.

3) Die Anzahl der Zeitgeber- und Zählerkontakte kann in Systemregister 5 geändert werden. Die Zahlen in der Tabelle entsprechen den Standardeinstellungen.

39.7.5 Bitmerker und Speicherbereiche für FP-e

Merker [Bits]

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge 1)	208	X0–X12F	%IX0.0– %IX12.15	Liefert den Zustand eines externen Eingangs.
Externe Ausgänge 1)	208	Y0–Y12F	%QX0.0– %QX12.15	Ansteuerung eines externen Ausgangs.
Interne Merker 2)	1008	R0–R62F	%MX0.0.0– %MX0.62.15	Zum Speichern von Bitinformationen im SPS-Programm.
Zeitgeber 2) 3)	100	T0–T99/ C100–C143	%MX1.0– %MX1.99/ %MX2.100– %MX2.143	Wird nur intern verwendet. Ausgangskontakt eines TM-Befehls.
Zähler 2) 3)	44	C100–C143/ T0–T99	%MX2.100– %MX2.143/ %MX1.0– %MX1.99	Wird nur intern verwendet. Ausgangskontakt eines CT-Befehls.
Sondermerker	64	R9000–R903F	%MX0.900.0– %MX0.903.15	Zustand wechselt je nach Bedingung. Wird intern als Merker verwendet.

Speicherbereich [Worte]

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge 1)	13	WX0–WX12	%IW0– %IW12	Liefert den Zustand von 16 externen Eingängen in einem Wort (16 Bit).
Externe Ausgänge 1)	13	WY0–WY12	%QW0– %QW12	Liefert den Zustand von 16 externen Ausgängen in einem Wort (16 Bit).
Interne Merker 2)	63	WR0–WR62	%MW0.0– %MW0.62	Für den wortweisen Zugriff auf 16 interne Merker.
Datenregister 2)	1660	DT0–DT1659	%MW5.0– %MW5.1659	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
Sollwerte für Zeitgeber/Zähler 2) 3)	144	SV0–SV143	%MW3.0– %MW3.143	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 2) 3)	144	EV0–EV143	%MW4.0– %MW4.143	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
Sonderdatenregister	112	DT90000– DT90111	%MW5.90000– %MW5.90111	Datenspeicher für Einstellungen und Fehlercodes.
Indexregister	2	IX, IY	%MW6.0– %MW6.1	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

Speicherbereich [Doppelworte]

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge 1)	6	DWX0–DWX11	%ID0– %ID11	Für den doppelwortweisen Zugriff auf 32 externe Eingänge.
Externe Ausgänge 1)	6	DWY0–DWY11	%QD0– %QD11	Für den doppelwortweisen Zugriff auf 32 externe Ausgänge.
Interne Merker 2)	31	DWR0–DWR61	%MD0.0– %MD0.61	Für den doppelwortweisen Zugriff auf 32 interne Merker.
Datenregister 2)	830	DDT0– DDT1658	%MD5.0– %MD5.1658	Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Sollwerte für Zeitgeber/Zähler 2) 3)	72	DSV0–DSV142	%MD3.0– %MD3.142	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 2) 3)	72	DEV0–DEV142	%MD4.0– %MD4.142	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
Sonderdatenregister	56	DDT90000– DDT90110	%MD5.90000– %MD5.90110	Datenspeicher für Einstellungen und Fehlercodes.
Indexregister	1	DI0	%MD6.0	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

1) Die angegebene Zahl der Eingangskontakte ist die für den internen Speicher reservierte Zahl. Die tatsächliche Anzahl wird durch die Hardware-Konfiguration bestimmt.

2) Der Speicherinhalt kann entweder selbsthaltend oder nicht selbsthaltend sein. Ist er selbsthaltend, geht er beim Ausschalten des Geräts oder beim Umschalten vom RUN- in den PROG-Modus nicht verloren. Ist er nicht selbsthaltend, wird der Speicher gelöscht. Standard-CPU: Die Speicherbereiche sind fest in selbsthaltende und nicht selbsthaltende Bereiche unterteilt (siehe unten). CPU mit Uhr-/Kalenderfunktion: Die Einstellung der selbsthaltenden und nicht selbsthaltenden Bereiche wird in den Systemregistern vorgenommen.

3) Die Anzahl der Zeitgeber- und Zählerkontakte kann in Systemregister 5 geändert werden. Die Zahlen in der Tabelle entsprechen den Standardeinstellungen.

Selbsthaltende und nicht selbsthaltende Speicherbereiche:

Speicherbereich		Standard-CPU	CPU mit Uhr-/Kalenderfunktion
Zeitgeber		Nicht selbsthaltend	
Zähler	**Nicht selbsthaltend**	Ab dem angegebenen Wert bis C139	
	Selbsthaltend	Nicht selbsthaltend	4 Kontakte (Istwerte) (C140–C143) 1)
Interne Merker	**Nicht selbsthaltend**	976 Kontakte (R0–R60F) 61 Worte (WR0–WR60)	
	Selbsthaltend	32 Kontakte (R610–R62F) 2 Worte (WR61–WR62)	
Daten-register	**Nicht selbsthaltend**	1652 Worte (DT0–DT1651)	
	Selbsthaltend	8 Worte (DT1652–DT1659)	

1) Es muss eine Batterie installiert sein.

39.7.6 Bitmerker und Speicherbereiche für FP2

Merker [Bits]

Typ	Speicher größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge	2048	X0–X127F	%IX0.0– %IX127.15	Liefert den Zustand eines externen Eingangs.
Externe Ausgänge	2048	Y0–Y127F	%QX0.0– %QX127.15	Ansteuerung eines externen Ausgangs.
Interne Merker 1)	4048	R0–R252F	%MX0.0– %MX0.252.15	Zum Speichern von Bitinformationen im SPS-Programm.
Koppelmerker 1)	2048	L0–L127F	%MX7.0.0– %MX7.127.15	Gemeinsam genutzter Merker bei SPS-Kopplung.
Zeitgeber 1) 2)	1024	T0–T999/ C1000–C1023	%MX1.0– %MX1.999/ %MX2.1000– %MX2.1023	Wird nur intern verwendet. Ausgangskontakt eines TM-Befehls.
Zähler 1) 2)	1024	C1000–C1023/ T0–T999	%MX2.1000– %MX2.1023/ %MX1.0– %MX1.999	Wird nur intern verwendet. Ausgangskontakt eines CT-Befehls.
Pulsmerker	1024	P0–P63F	%MX11.0.0– %MX11.63.15	Wird nur für einen Zyklus eingeschaltet. Wird intern vom SPS-Programm verwendet.
Sondermerker	176	R9000–R910F	%MX0.900.0– %MX0.910.15	Zustand wechselt je nach Bedingung. Wird intern als Merker verwendet.

Speicherbereich [Worte]

Typ		Speicher-größe	Verfügbarer Adressbereich		Beschreibung
			F/P	IEC	
Externe Eingänge		128	WX0–WX127	%IW0– %IW127	Liefert den Zustand von 16 externen Eingängen in einem Wort (16 Bit).
Externe Ausgänge		128	WY0–WY127	%QW0– %QW127	Liefert den Zustand von 16 externen Ausgängen in einem Wort (16 Bit).
Interne Merker		253	WR0–WR252	%MW0.0– %MW0.252	Für den wortweisen Zugriff auf 16 interne Merker.
Koppelmerker		128	WL0–WL127	%MW7.0– %MW7.127	Für den wortweisen Zugriff auf 16 Koppelmerker.
Datenregister 1)		6000	DT0–DT5999	%MW5.0– %MW5.5999	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
Koppeldatenregister 1)		256	LD0–LD255	%MW8.0– %MW8.255	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden wortweise (16 Bit) verarbeitet..
Sollwerte für Zeitgeber/Zähler 1)		1024	SV0–SV1023	%MW3.0– %MW3.1023	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 1)		1024	EV0–EV1023	%MW4.0– %MW4.1023	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
File-Register 1) 3)	**12k-Typ**	14333	FL0–FL14332	%MW9.0– %MW9.14332	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
	32k-Typ (erweitert)	30717	FL0–FL30716	%MW9.0– %MW9.30716	
Sonderdatenregister		256	DT90000– DT90255	%MW5.90000– %MW5.90255	Datenspeicher für Einstellungen und Fehlercodes.
Indexregister		14	I0–ID	%MW6.0– %MW6.13	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

Speicherbereich [Doppelworte]

Typ		Speicher-größe	Verfügbarer Adressbereich		Beschreibung
			F/P	IEC	
Externe Eingänge 1)		64	DWX0–DWX72	%ID0– %ID72	Für den doppelwortweisen Zugriff auf 32 externe Eingänge.
Externe Ausgänge 1)		64	DWY0–DWY72	%QD0– %QD72	Für den doppelwortweisen Zugriff auf 32 externe Ausgänge.
Interne Merker 2)		126	DWR0– DWR254	%MD0.0– %MD0.254	Für den doppelwortweisen Zugriff auf 32 interne Merker.
Koppelmerker		64	DWL0–DWL126	%MD7.0– %MD7.126	Für den doppelwortweisen Zugriff auf 32 Koppelmerker.
Datenregister 2)		3000	DDT0– DDT32763	%MD5.0– %MD5.32763	Vom Programm verwendeter Datenspeicher. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Koppeldatenregister 2)		128	DLD0–DLD254	%MD8.0– %MD8.254	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Sollwerte für Zeitgeber/Zähler 2)		512	DSV0–DSV1022	%MD3.0– %MD3.1022	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 2)		512	DEV0–DEV1022	%MD4.0– %MD4.1022	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
File-Register 1) 3)	**12k-Typ**	7166	DFL0– DFL14331	%MD9.0– %MD9.14331	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
	32k-Typ (erweitert)	15358	DFL0– DFL30715	%MD9.0– %MD9.30715	
Sonderdatenregister		128	DDT90000– DDT90254	%MD5.90000– %MD5.90254	Datenspeicher für Einstellungen und Fehlercodes.
Indexregister		7	DI0–DID	%MD6.0– %MD6.13	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

1) Der Speicherinhalt kann entweder selbsthaltend oder nicht selbsthaltend sein. Ist er selbsthaltend, geht er beim Ausschalten des Geräts oder beim Umschalten vom RUN- in den PROG-Modus nicht verloren. Ist er nicht selbsthaltend, wird der Speicher gelöscht. Die Einstellung der selbsthaltenden und nicht selbsthaltenden Bereiche wird in den Systemregistern vorgenommen.

2) Die Anzahl der Zeitgeber- und Zählerkontakte kann in Systemregister 5 geändert werden. Die Zahlen in der Tabelle entsprechen den Standardeinstellungen.

3) Die Größe der File-Register richtet sich nach den Einstellungen der Systemregister 0, 1 und 2.

39.7.7 Bitmerker und Speicherbereiche für FP2SH

Merker [Bits]

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge	8192	X0–X511F	%IX0.0– %IX511.15	Liefert den Zustand eines externen Eingangs.
Externe Ausgänge	8192	Y0–Y511F	%QX0.0– %QX511.15	Ansteuerung eines externen Ausgangs.
Interne Merker 1)	14192	R0–R886F	%MX0.0.0– %MX0.886.15	Zum Speichern von Bitinformationen im SPS-Programm.
Koppelmerker 1)	10240	L0–L639F	%MX7.0.0– %MX7.639.15	Gemeinsam genutzter Merker bei SPS-Kopplung.
Zeitgeber 1) 2)	3072	T0–T2999/ C3000–C3071	%MX1.0– %MX1.2999/ %MX2.3000– %MX2.3071	Wird nur intern verwendet. Ausgangskontakt eines TM-Befehls.
Zähler 1) 2)	3072	C3000–C3071/ T0–T2999	%MX2.3000– %MX2.3071/ %MX1.0– %MX1.2999	Wird nur intern verwendet. Ausgangskontakt eines CT-Befehls.
Pulsmerker	2048	P0–P127F	%MX11.0.0– %MX11.127.15	Wird nur für einen Zyklus eingeschaltet. Wird intern vom SPS-Programm verwendet.
Fehleralarm-Merker	2048	E0–E127F	%MX10.0.0– %MX10.127.15	Wird bei einem Fehler eingeschaltet. Die Fehlerhistorie wird in einem speziellen Datenregister gespeichert. Prüfen
Sondermerker	176	R9000–R910F	%MX0.900.0– %MX0.910.15	Zustand wechselt je nach Bedingung. Wird intern als Merker verwendet.

Speicherbereich [Worte]

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge	512	WX0–WX127	%IW0– %IW127	Liefert den Zustand von 16 externen Eingängen in einem Wort (16 Bit).
Externe Ausgänge	512	WY0–WY127	%QW0– %QW127	Liefert den Zustand von 16 externen Ausgängen in einem Wort (16 Bit).
Interne Merker	887	WR0–WR252	%MW0.0– %MW0.252	Für den wortweisen Zugriff auf 16 interne Merker.
Koppelmerker	640	WL0–WL127	%MW7.0– %MW7.127	Für den wortweisen Zugriff auf 16 Koppelmerker.
Datenregister 1)	10240	DT0–DT5999	%MW5.0– %MW5.5999	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
Koppeldatenregister 1)	8448	LD0–LD255	%MW8.0– %MW8.255	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden wortweise (16 Bit) verarbeitet..
Sollwerte für Zeitgeber/Zähler 1)	3072	SV0–SV1023	%MW3.0– %MW3.1023	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 1)	3072	EV0–EV1023	%MW4.0– %MW4.1023	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
File-Register 1)	98295 (32765 ´ 3 Speicher-bänke)	FL0–FL32764	%MW9.0– %MW9.32764	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
Sonderdatenregister	512	DT90000– DT90511	%MW5.90000– %MW5.90511	Datenspeicher für Einstellungen und Fehlercodes.

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Indexregister	14	I0–ID	%MW6.0– %MW6.13	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

Speicherbereich [Doppelworte]

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge 1)	256	DWX0–DWX510	%ID0– %ID510	Für den doppelwortweisen Zugriff auf 32 externe Eingänge.
Externe Ausgänge 1)	256	DWY0–DWY510	%QD0– %QD510	Für den doppelwortweisen Zugriff auf 32 externe Ausgänge.
Interne Merker 2)	443	DWR0– DWR885	%MD0.0– %MD0.885	Für den doppelwortweisen Zugriff auf 32 interne Merker.
Koppelmerker	320	DWL0–DWL638	%MD7.0– %MD7.638	Für den doppelwortweisen Zugriff auf 32 Koppelmerker.
Datenregister 2)	5120	DDT0– DDT10238	%MD5.0– %MD5.10238	Vom Programm verwendeter Datenspeicher. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Koppeldatenregister 2)	4224	DLD0–DLD8446	%MD8.0– %MD8.8446	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Sollwerte für Zeitgeber/Zähler 2)	1513	DSV0–DSV3070	%MD3.0– %MD3.3070	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 2)	1513	DEV0–DEV3070	%MD4.0– %MD4.3070	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
File-Register	16382	DFL0– DFL32763	%MD9.0– %MD9.32763	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
Sonderdatenregister	265	DDT90000– DDT90510	%MD5.90000– %MD5.90510	Datenspeicher für Einstellungen und Fehlercodes.
Indexregister	7	DI0–DID	%MD6.0– %MD6.13	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

1) Der Speicherinhalt kann entweder selbsthaltend oder nicht selbsthaltend sein. Ist er selbsthaltend, geht er beim Ausschalten des Geräts oder beim Umschalten vom RUN- in den PROG-Modus nicht verloren. Ist er nicht selbsthaltend, wird der Speicher gelöscht. Die Einstellung der selbsthaltenden und nicht selbsthaltenden Bereiche wird in den Systemregistern vorgenommen.

2) Die Anzahl der Zeitgeber- und Zählerkontakte kann in Systemregister 5 geändert werden. Die Zahlen in der Tabelle entsprechen den Standardeinstellungen.

39.7.8 Bitmerker und Speicherbereiche für FP10SH

Merker [Bits]

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge	8192	X0–X511F	%IX0.0– %IX511.15	Liefert den Zustand eines externen Eingangs.
Externe Ausgänge	8192	Y0–Y511F	%QX0.0– %QX511.15	Ansteuerung eines externen Ausgangs.
Interne Merker 1)	14192	R0–R886F	%MX0.0.0– %MX0.886.15	Zum Speichern von Bitinformationen im SPS-Programm.
Koppelmerker 1)	10240	L0–L639F	%MX7.0.0– %MX7.639.15	Gemeinsam genutzter Merker bei SPS-Kopplung.
Zeitgeber 1) 2)	3072	T0–T2999/ C3000–C3071	%MX1.0– %MX1.2999/ %MX2.3000– %MX2.3071	Wird nur intern verwendet. Ausgangskontakt eines TM-Befehls.
Zähler 1) 2)	3072	C3000–C3071/ T0–T2999	%MX2.3000– %MX2.3071/ %MX1.0– %MX1.2999	Wird nur intern verwendet. Ausgangskontakt eines CT-Befehls.
Pulsmerker	2048	P0–P127F	%MX11.0.0– %MX11.127.15	Wird nur für einen Zyklus eingeschaltet. Wird intern vom SPS-Programm verwendet.
Fehleralarm-Merker	2048	E0–E127F	%MX10.0.0– %MX10.127.15	Wird bei einem Fehler eingeschaltet. Die Fehlerhistorie wird in einem speziellen Datenregister gespeichert. Prüfen
Sondermerker	176	R9000–R910F	%MX0.900.0– %MX0.910.15	Zustand wechselt je nach Bedingung. Wird intern als Merker verwendet.

Speicherbereich [Worte]

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge	512	WX0–WX127	%IW0– %IW127	Liefert den Zustand von 16 externen Eingängen in einem Wort (16 Bit).
Externe Ausgänge	512	WY0–WY127	%QW0– %QW127	Liefert den Zustand von 16 externen Ausgängen in einem Wort (16 Bit).
Interne Merker	887	WR0–WR252	%MW0.0– %MW0.252	Für den wortweisen Zugriff auf 16 interne Merker.
Koppelmerker	640	WL0–WL127	%MW7.0– %MW7.127	Für den wortweisen Zugriff auf 16 Koppelmerker.
Datenregister 1)	10240	DT0–DT5999	%MW5.0– %MW5.5999	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
Koppeldatenregister 1)	8448	LD0–LD255	%MW8.0– %MW8.255	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden wortweise (16 Bit) verarbeitet..
Sollwerte für Zeitgeber/Zähler 1)	3072	SV0–SV1023	%MW3.0– %MW3.1023	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler 1)	3072	EV0–EV1023	%MW4.0– %MW4.1023	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
File-Register 1)	32765	FL0–FL32764	%MW9.0– %MW9.32764	Die Daten werden wortweise (16 Bit) verarbeitet. Vom Programm verwendeter Datenspeicher.
Sonderdatenregister	512	DT90000– DT90511	%MW5.90000– %MW5.90511	Datenspeicher für Einstellungen und Fehlercodes.

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Indexregister	14	I0–ID	%MW6.0– %MW6.13	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

Speicherbereich [Doppelworte]

Typ	Speicher-größe	Verfügbarer Adressbereich		Beschreibung
		F/P	IEC	
Externe Eingänge [1]	256	DWX0–DWX510	%ID0– %ID510	Für den doppelwortweisen Zugriff auf 32 externe Eingänge.
Externe Ausgänge [1]	256	DWY0–DWY510	%QD0– %QD510	Für den doppelwortweisen Zugriff auf 32 externe Ausgänge.
Interne Merker [2]	443	DWR0– DWR885	%MD0.0– %MD0.885	Für den doppelwortweisen Zugriff auf 32 interne Merker.
Koppelmerker	320	DWL0–DWL638	%MD7.0– %MD7.638	Für den doppelwortweisen Zugriff auf 32 Koppelmerker.
Datenregister [2]	5120	DDT0– DDT10238	%MD5.0– %MD5.10238	Vom Programm verwendeter Datenspeicher. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Koppeldatenregister [2]	4224	DLD0–DLD8446	%MD8.0– %MD8.8446	Datenspeicher, der von mehreren Steuerungen genutzt wird, die über SPS-Kopplung vernetzt sind. Die Daten werden als Doppelworte (32 Bit) verarbeitet.
Sollwerte für Zeitgeber/Zähler [2]	1513	DSV0–DSV3070	%MD3.0– %MD3.3070	Datenspeicher für Zeitgeber- und Zählersollwerte. Die Werte werden mit der Zeitgeber-/Zählernummer gespeichert.
Istwerte für Zeitgeber/Zähler [2]	1513	DEV0–DEV3070	%MD4.0– %MD4.3070	Datenspeicher für Zeitgeber- und Zähleristwerte. Die Werte werden anhand der Zeitgeber-/Zählernummer gespeichert.
File-Register	16382	DFL0– DFL32763	%MD9.0– %MD9.32763	Vom Programm verwendeter Datenspeicher. Die Daten werden wortweise (16 Bit) verarbeitet.
Sonderdatenregister	265	DDT90000– DDT90510	%MD5.90000– %MD5.90510	Datenspeicher für Einstellungen und Fehlercodes.
Indexregister	7	DI0–DID	%MD6.0– %MD6.13	Datenspeicher für Änderung der Konstanten und Speicherbereichsadressen.

1) Der Speicherinhalt kann entweder selbsthaltend oder nicht selbsthaltend sein. Ist er selbsthaltend, geht er beim Ausschalten des Geräts oder beim Umschalten vom RUN- in den PROG-Modus nicht verloren. Ist er nicht selbsthaltend, wird der Speicher gelöscht. Die Einstellung der selbsthaltenden und nicht selbsthaltenden Bereiche wird in den Systemregistern vorgenommen.

2) Die Anzahl der Zeitgeber- und Zählerkontakte kann in Systemregister 5 geändert werden. Die Zahlen in der Tabelle entsprechen den Standardeinstellungen.

39.8 Fehlercodes

39.8.1 Fehlercodes E1 bis E8

Fehlercode	Fehlername	SPS-Betrieb	Beschreibung und Fehlerbehebung
E1 (siehe Hinweis)	Syntaxfehler	Stoppt	Das Programm enthält einen Syntaxfehler. Wechseln Sie in den PROG-Modus und beheben Sie den Fehler.
E2 (siehe Hinweis)	Ausgänge mehrfach belegt	Stoppt	Einem Ausgang wurde mehr als einmal im Programm ein Verknüpfungsergebnis zugewiesen. (Dieser Fehler tritt auch auf, wenn dieselbe Zeitgeber-/Zählernummer verwendet wird.) Wechseln Sie in den PROG-Modus und beheben Sie den Fehler. Dieser Fehler wird auch beim Online-Editieren erkannt. Der Betrieb wird ohne Änderung des Programms fortgesetzt.
E3	Befehlspaar unvollständig	Stoppt	Bei einem Befehlspaar, z. B. JP und LBL, fehlt ein Befehlsteil oder die Reihenfolge ist vertauscht. Wechseln Sie in den PROG-Modus und beheben Sie den Fehler.
E4 (siehe Hinweis)	Systemregister-Parameter fehlerhaft	Stoppt	Der im Befehl verwendete Operand liegt nicht in dem Bereich, der im Systemregister definiert worden ist. Beispiel: Die angegebene Zahl für Zeitgeber/Zähler liegt außerhalb des festgelegten Bereichs. Wechseln Sie in den PROG-Modus und beheben Sie den Fehler.
E5 (siehe Hinweis)	Befehlsposition fehlerhaft	Stoppt	Ein Befehl befindet sich nicht an der erwarteten Position (im Haupt- oder im Unterprogramm). Wechseln Sie in den PROG-Modus und beheben Sie den Fehler. Dieser Fehler wird auch beim Online-Editieren erkannt. Der Betrieb wird ohne Änderung des Programms fortgesetzt.
E6 (siehe Hinweis)	Programmspeicherüberlauf	Stoppt	Das in der SPS gespeicherte Programm ist zu groß für den Programmspeicher des Compilers. Wechseln Sie in den PROG-Modus und beheben Sie den Fehler.
E7 (siehe Hinweis)	Gemischte Befehls-Trigger	Stoppt	Im Programm sind pegel- und flankengetriggerte erweiterte Befehle an ein und dasselbe Verknüpfungsergebnis angebunden. Wechseln Sie in den PROG-Modus und programmieren Sie pegel- und flankengetriggerte erweiterte Befehle so, dass nur jeweils homogene Befehlsgruppen von ein und demselben Verknüpfungsergebnis abhängen.
E8	Falscher Operand	Stoppt	Ein Operand in einem Befehl, der Operanden gleichen Typs erfordert, ist ungültig. Wechseln Sie in den PROG-Modus und beheben Sie den Fehler.

◆**HINWEIS**

In FPWIN Pro werden dieses Fehler vom Compiler abgefangen. Sie sind daher unkritisch.

39.8.2 Selbstdiagnosefehler

Fehler code	Fehlername	SPS-Betrieb	Beschreibung und Fehlerbehebung
E26	Fehler im ROM-Zusatzspeicher	Stoppt	Vermutlich ein Hardware-Fehler. Bitte wenden Sie sich an Ihren Händler.
E27	Zu viele Module gesteckt	Stoppt	Es sind zu viele Module gesteckt. Schalten Sie die Steuerung ab und prüfen Sie die maximal zulässige Zahl der Module.
E28	Systemregisterfehler	Stoppt	Systemregister vermutlich fehlerhaft. Überprüfen Sie die Systemregistereinstellungen.
E30	Interrupt-Fehler 0	Stoppt	Vermutlich ein Hardware-Fehler. Bitte wenden Sie sich an Ihren Händler.
E31	Interrupt-Fehler 1	Stoppt	Eine Unterbrechung wurde ohne Unterbrechungsanforderung ausgeführt. Vermutlich liegt ein Hardware-Fehler oder eine Beeinträchtigung durch Störstrahlung vor. Schalten Sie die Steuerung ab und prüfen Sie, ob Störstrahlungen auftreten.

Fehler code	Fehlername		SPS-Betrieb	Beschreibung und Fehlerbehebung
E32	Interrupt-Fehler 2		Stoppt	Eine Unterbrechung wurde ohne Unterbrechungsanforderung ausgeführt. Vermutlich liegt ein Hardware-Fehler oder eine Beeinträchtigung durch Störstrahlung vor. Schalten Sie die Steuerung ab und prüfen Sie, ob Störstrahlungen auftreten.
				Die Unterbrechung wurde nicht von einem Interrupt-Programm verursacht. Prüfen Sie in der Task-Liste, ob für das betreffende Interrupt ein Programm eingetragen wurde.
E34	Modul-Fehler		Stoppt	Ein Modul ist fehlerhaft. Tauschen Sie das Modul aus.
E42	Position eines E/A-Moduls hat sich geändert oder E/A-Modul ist defekt		Einstellbar	Die Position eines E/A-Moduls wurde nach dem Einschalten geändert. Überprüfen Sie mit sys_wVerifyErrorUnit_0_15, um welches Modul es sich handelt. In Systemregister 23 können Sie einstellen, ob der Betrieb bei diesem Fehler angehalten oder fortgesetzt werden soll.
E45	Operationsfehler		Einstellbar	Bei der Ausführung eines erweiterten Befehls trat ein Fehler auf. Ein Operationsfehler kann je nach Befehl unterschiedliche Ursachen haben. In Systemregister 23 können Sie einstellen, ob der Betrieb bei diesem Fehler angehalten oder fortgesetzt werden soll.
E100–E299	Selbstdiagnose-Fehler, der durch den Befehl F148_ERR ausgelöst wird	E100–E199	Stoppt	Der im Befehl F148_ERR (s. S. 1023) gesetzte Selbstdiagnosefehler ist aufgetreten. Prüfen Sie in FPWIN Pro im Fenster "SPS-Status", um welchen Fehler es sich handelt.
		E200–E299	Läuft weiter	

39.8.3 MEWTOCOL-COM-Fehlercodes

Fehlercode	Name	Beschreibung
!21	NACK-Fehler	Netzwerkfehler
!22	WACK-Fehler	
!23	Teilnehmeradresse doppelt	
!24	Fehler im Übertragungsformat	
!25	Hardware-Fehler	
!26	Teilnehmeradresse fehlerhaft	
!27	Befehl wird nicht unterstützt	
!28	Keine Antwort	
!29	Puffer geschlossen	
!30	Zeitüberschreitung	
!32	Übertragung nicht möglich	
!33	Kommunikation unterbrochen	
!36	Keine Zieladresse	
!38	Anderer Kommunikationsfehler	
!40	BCC-Fehler	Übertragungsfehler in empfangenen Daten
!41	Formatfehler	Formatfehler in empfangenem Befehl
!42	Befehl wird nicht unterstützt	Der empfangene Befehl wird nicht unterstützt
!43	Multi-Frame-Verarbeitungsfehler	Während der Multi-Frame-Verarbeitung wurde ein neuer Befehl empfangen
!50	Koppelprozessornummer fehlerhaft	Die angegebene Route-Nummer existiert nicht.
!51	Übertragungsfehler	Die Datenübertragung ist nicht möglich, da der Sendepuffer voll ist.
!52	Übertragungsfehler	Die Datenübertragung ist nicht möglich; Fehler unbekannt.
!53	Übertragung nicht möglich	Empfangener Befehl kann nicht verarbeitet werden wegen Multi-Frame-Verarbeitung oder weil Vorgängerbefehl noch nicht verarbeitet wurde.
!60	Parameterfehler	Die Parameterangabe ist fehlerhaft.

Fehlercode	Name	Beschreibung
!61	Datenfehler	Operandenadresse, Speicherbereich oder Speicherformat fehlerhaft.
!62	Registrierungsüberlauf	Die Zahl der Registrierungen wurde überschritten oder es wurden keine Daten registriert.
!63	SPS-Modusfehler	Der Befehl kann nicht verarbeitet werden, da SPS im RUN-Modus.
!64	Externer Speicherfehler	Beim Laden des RAM-Inhalts in den ROM-Speicher/auf die IC-Karte ist ein Fehler aufgetreten. ▪ Der installierte ROM/die IC-Karte ist möglicherweise defekt. ▪ Die Kapazität des Speichers wurde überschritten. Ein Schreibfehler ist aufgetreten. ▪ ROM oder IC-Speicherkarte ist nicht installiert. ROM oder IC-Speicherkarte entspricht nicht den Spezifikationen.
!65	Schutzverletzung	Schreibversuch in Programm oder Systemregister mit Schreibschutz (Passwort, DIP-Schaltereinstellung usw.) oder im ROM-Betrieb.
!66	Adressfehler	Fehler im Adressformat oder in der Bereichsangabe.
!67	Programm oder Daten nicht vorhanden	Daten können nicht gelesen werden, da im Programmbereich kein Programm oder Fehler im Speicher. Oder zu lesende Daten sind nicht registriert.
!68	Übertragen des Programms im RUN-Modus nicht möglich	Die Befehle ED, SUB, RET, INT, IRET, SSTP und STPE dürfen nicht im RUN-Modus auf die SPS übertragen werden.
!70	Programmspeicherüberlauf	Beim Einfügen eines Programmteils wurde die maximale Anzahl der Programmschritte überschritten.
!71	Fehler bei exklusivem Zugriff	Befehl kann nicht ausgeführt werden, weil Vorgängerbefehl noch nicht verarbeitet ist.

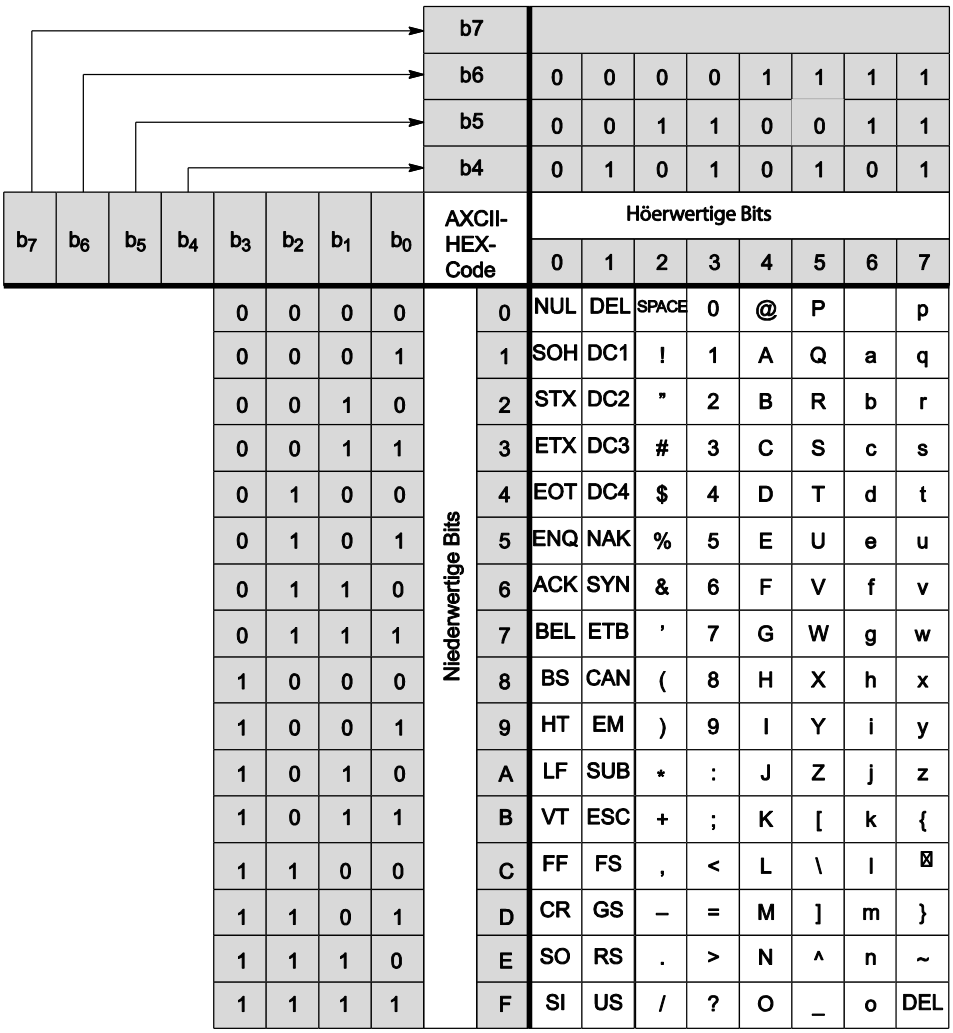
39.9 MEWTOCOL-COM-Befehle

Befehlsname	Code	Beschreibung
Read contact area	RC (RCS) (RCP) (RCC)	TRUE-/FALSE-Zustand des Kontaktbereichs lesen ▪ Einen Bitoperanden lesen ▪ Mehere Bitoperanden lesen ▪ Wortoperanden lesen
Write contact area	WC (WCS) (WCP) (WCC)	TRUE/FALSE-Zustand des Kontaktbereichs ändern ▪ Einen Bitoperanden ändern ▪ Mehere Bitoperanden ändern ▪ Wortoperanden ändern
Read data area	RD	Wortoperanden im Datenbereich lesen
Write data area	WD	Wortoperanden im Datenbereich ändern
Read timer/counter set value area	RS	Sollwert für Zeitgeber/Zähler lesen
Write timer/counter set value area	WS	Sollwert für Zeitgeber/Zähler ändern
Read timer/counter elapsed value area	RK	Istwert für Zeitgeber/Zähler lesen
Write timer/counter elapsed value area	WK	Istwert für Zeitgeber/Zähler ändern
Register or Reset contacts monitored	MC	Bitoperandennummer (Kontaktnr.) für Kontaktmonitor setzen und zurücksetzen
Register or Reset data monitored	MD	Wortoperandennummer (Kontaktnr.) für Datenmonitor setzen und zurücksetzen
Monitoring start	MG	Monitor starten
Preset contact area (Kopierbefehl)	SC	Wortoperanden (Kontakte) im Kontaktbereich mit einem 16-Bit-Muster setzen
Preset data area (Kopierbefehl)	SD	Gleiches Wort in jedes Register des angegebenen Datenbereichs schreiben
Read system register	RR	Systemregister lesen
Write system register	WR	Systemregistereinstellungen ändern
Read the status of PLC	RT	SPS-Zustand und ggf. Fehlercode lesen
Remote control	RM	SPS-Modus (RUN-/PROG-Modus) umschalten
Abort	AB	Kommunikation abbrechen

39.10 Binär-, Hex- und BCD-Code

Dezimal	Hexadezimal	Binärwerte	BCD (Binary Coded Decimal)
0	0000	0000 0000 0000 0000	0000 0000 0000 0000
1	0001	0000 0000 0000 0001	0000 0000 0000 0001
2	0002	0000 0000 0000 0010	0000 0000 0000 0010
3	0003	0000 0000 0000 0011	0000 0000 0000 0011
4	0004	0000 0000 0000 0100	0000 0000 0000 0100
5	0005	0000 0000 0000 0101	0000 0000 0000 0101
6	0006	0000 0000 0000 0110	0000 0000 0000 0110
7	0007	0000 0000 0000 0111	0000 0000 0000 0111
8	0008	0000 0000 0000 1000	0000 0000 0000 1000
9	0009	0000 0000 0000 1001	0000 0000 0000 1001
10	000A	0000 0000 0000 1010	0000 0000 0001 0000
11	000B	0000 0000 0000 1011	0000 0000 0001 0001
12	000C	0000 0000 0000 1100	0000 0000 0001 0010
13	000D	0000 0000 0000 1101	0000 0000 0001 0011
14	000E	0000 0000 0000 1110	0000 0000 0001 0100
15	000F	0000 0000 0000 1111	0000 0000 0001 0101
16	0010	0000 0000 0001 0000	0000 0000 0001 0110
17	0011	0000 0000 0001 0001	0000 0000 0001 0111
18	0012	0000 0000 0001 0010	0000 0000 0001 1000
19	0013	0000 0000 0001 0011	0000 0000 0001 1001
20	0014	0000 0000 0001 0100	0000 0000 0010 0000
21	0015	0000 0000 0001 0101	0000 0000 0010 0001
22	0016	0000 0000 0001 0110	0000 0000 0010 0010
23	0017	0000 0000 0001 0111	0000 0000 0010 0011
24	0018	0000 0000 0001 1000	0000 0000 0010 0100
25	0019	0000 0000 0001 1001	0000 0000 0010 0101
26	001A	0000 0000 0001 1010	0000 0000 0010 0110
27	001B	0000 0000 0001 1011	0000 0000 0010 0111
28	001C	0000 0000 0001 1100	0000 0000 0010 1000
29	001D	0000 0000 0001 1101	0000 0000 0010 1001
30	001E	0000 0000 0001 1110	0000 0000 0011 0000
31	001F	0000 0000 0001 1111	0000 0000 0011 0001
.	.	.	.
.	.	.	.
.	.	.	.
63	**003F**	0000 0000 0011 1111	0000 0000 0110 0011
.	.	.	.
.	.	.	.
.	.	.	.
255	**00FF**	0000 0000 1111 1111	0000 0010 0101 0101
.	.	.	.
.	.	.	.
.	.	.	.
9999	**270F**	0010 0111 0000 1111	1001 1001 1001 1001

39.11 ASCII-Codes



b7															
b6								0	0	0	0	1	1	1	1
b5								0	0	1	1	0	0	1	1
b4								0	1	0	1	0	1	0	1
b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	AXCII-HEX-Code		Höerwertige Bits					
										0	1	2	3	4	5	6	7
				0	0	0	0	Niederwertige Bits	0	NUL	DEL	SPACE	0	@	P		p
				0	0	0	1		1	SOH	DC1	!	1	A	Q	a	q
				0	0	1	0		2	STX	DC2	"	2	B	R	b	r
				0	0	1	1		3	ETX	DC3	#	3	C	S	c	s
				0	1	0	0		4	EOT	DC4	$	4	D	T	d	t
				0	1	0	1		5	ENQ	NAK	%	5	E	U	e	u
				0	1	1	0		6	ACK	SYN	&	6	F	V	f	v
				0	1	1	1		7	BEL	ETB	'	7	G	W	g	w
				1	0	0	0		8	BS	CAN	(	8	H	X	h	x
				1	0	0	1		9	HT	EM	)	9	I	Y	i	y
				1	0	1	0		A	LF	SUB	*	:	J	Z	j	z
				1	0	1	1		B	VT	ESC	+	;	K	[	k	{
				1	1	0	0		C	FF	FS	,	<	L	\	l	⊠
				1	1	0	1		D	CR	GS	–	=	M	]	m	}
				1	1	1	0		E	SO	RS	.	>	N	^	n	~
				1	1	1	1		F	SI	US	/	?	O	_	o	DEL

39.12 Verfügbarkeit aller Befehle für alle SPS-Typen

Befehl ● verfügbar ❍ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
ABS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	66
ACOS	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	72
ActivateStepsOfStoppedSfc	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
ADD_DT_TIME	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	276
ADD_TIME	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	324
ADD_TOD_TIME	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	277
Adr_Of_Var	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
Adr_Of_VarOffs	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
AdrDT_Of_Offs	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
AdrFL_Of_Offs	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
AdrLast_Of_Var	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
AllSfcsStopped	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
ALT			●	●	●	●	●	●	●	●			●	●	●	●	●	850
AreaOffs_ToVar	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
ASIN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	70
ATAN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	74
ATAN2_YX	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	75
BOOL_TO_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	168
BOOL_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	132
BOOL_TO_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	145
BOOL_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	210
BOOL_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	180
BOOL_TO_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	157
BOOL_TO_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	120
BOOL16_TO_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	121
BOOL32_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	133
BOOLS_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	134
BOOLS_TO_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	122
BRK												●	●	●	●	●	●	1036
ClearReceiveBuffer	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	719
CONCAT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	265
CONCAT_DATE_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	278
CONCAT_DATE_TOD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	279
CONCAT_DT_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	280
CONCAT_TIME_INT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	325
CONCAT_TOD_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	282
ControlSfc	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
ControlSfcAndData	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
COS	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	71
CRC16	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	81

Befehl ● verfügbar ❍ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
CT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	854
CT_FB	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	852
CTD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	312
CTU	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	310
CTUD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	314
DATE_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	216
DATE_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	190
DAY_OF_WEEK1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
DELETE	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	267
DF	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	846
DFI			●	●			●	●	●	●	●		●	●	●	●	●	848
DFN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	847
DINT_TO_BCD_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	242
DINT_TO_BOOL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	117
DINT_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	138
DINT_TO_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	149
DINT_TO_REAL	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	195
DINT_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	221
DINT_TO_STRING_LEADING_ZEROS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	223
DINT_TO_TIME	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	203
DINT_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	186
DINT_TO_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	162
DINT_TO_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	126
DIV_TIME_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	328
DIV_TIME_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	327
DIV_TIME_REAL	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	329
DT_TO_DATE	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	206
DT_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	217
DT_TO_TOD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	208
DT_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	191
DWORD_BCD_TO_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	170
DWORD_BCD_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	183
DWORD_TO_BOOL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	115
DWORD_TO_BOOL32	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	236
DWORD_TO_BOOLS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	239
DWORD_TO_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	171
DWORD_TO_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	148
DWORD_TO_REAL	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	193
DWORD_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	214
DWORD_TO_TIME	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	201
DWORD_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	182
DWORD_TO_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	160
DWORD_TO_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	124

Befehl ● verfügbar ◑ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
Elem_OfArray1D	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
Elem_OfArray2D	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
Elem_OfArray3D	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
ETLANADDR_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	232
ETLANADDR_TO_STRING_NO_LEADING_ZEROS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	233
EXP	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	79
ExpansionUnitNumberToIOWordOffset_FP0	●	●	●	●	●	●												1054
ExpansionUnitNumberToIOWordOffset_FPX_FP0							●	●	●	●								1055
EXPT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	80
F_TRIG	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	307
F0_MV	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	762
F1_DMV	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	764
F10_BKMV	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	776
F10_BKMV_NUMBER	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	778
F10_BKMV_NUMBER_OFFSET	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	781
F10_BKMV_OFFSET	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	780
F100_SHR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	540
F101_SHL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	542
F102_DSHR			●	●	●	●	●	●	●	●			●	●	●	●	●	544
F103_DSHL			●	●	●	●	●	●	●	●			●	●	●	●	●	546
F105_BSR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	548
F106_BSL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	550
F108_BITR			●	●	●	●	●	●	●	●			●	●	●	●	●	552
F109_BITL			●	●	●	●	●	●	●	●			●	●	●	●	●	554
F11_COPY	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	783
F110_WSHR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	556
F111_WSHL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	558
F112_WBSR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	560
F113_WBSL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	562
F115_FIFT			●	●	●	●	●	●	●	●		●	●	●	●	●	●	474
F116_FIFR			●	●	●	●	●	●	●	●		●	●	●	●	●	●	477
F117_FIFW			●	●	●	●	●	●	●	●		●	●	●	●	●	●	480
F118_UDC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	857
F119_LRSR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	564
F12_EPRD	●	●	●	●	●	●	●	●	●	●	●		●	●				785
F12_ICRD															●	●	●	787
F120_ROR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	567
F121_ROL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	569
F122_RCR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	571
F123_RCL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	573
F125_DROR			●	●	●	●	●	●	●	●			●	●	●	●	●	575

Befehl ● verfügbar ❍ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
F126_DROL			●	●	●	●	●	●	●	●			●	●	●	●	●	577
F127_DRCR			●	●	●	●	●	●	●	●			●	●	●	●	●	579
F128_DRCL			●	●	●	●	●	●	●	●			●	●	●	●	●	581
F13_ICWT															●	●	●	789
F130_BTS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	522
F131_BTR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	523
F132_BTI	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	524
F133_BTT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	525
F135_BCU	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	527
F136_DBCU	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	528
F137_STMR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	973
F138_TIMEBCD_TO_SECBCD		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	830
F139_SECBCD_TO_TIMEBCD		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	831
F14_PGRD															●	●	●	791
F140_STC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1020
F141_CLC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1021
F142_WDT															●	●	●	1022
F143_IORF	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	801
F145_SEND			●	●		●	●			●		●	●	●	●	●	●	745
F145_WRITE_DATA			●	●		●	●			●								724
F145_WRITE_DATA_TYPE_OFFS			●	●		●	●			●								726
F145F146_MODBUS_COMMAND			●	●		●	●			●								734
F145F146_MODBUS_MASTER			●	●		●	●			●								737
F146_READ_DATA			●	●		●	●			●								729
F146_READ_DATA_TYPE_OFFS			●	●		●	●			●								731
F146_RECV			●	●		●	●			●		●	●	●	●	●	●	747
F147_PR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	803
F148_ERR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1023
F149_MSG	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1025
F15_XCH	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	794
F150_READ		●	●	●	●	●	●					●	●	●	●	●	●	806
F151_WRT		●	●	●	●	●	●					●	●	●	●	●	●	809
F152_RMRD												●	●	●	●	●	●	750
F153_RMWT												●	●	●	●	●	●	753
F155_SMPL			●	●		●	●			●		●		●	●	●	●	1026
F156_STRG			●	●		●	●			●		●		●	●	●	●	1027
F157_ADD_DTBCD_TIMEBCD		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	832
F158_SUB_DTBCD_TIMEBCD		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	834
F159_MTRN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	720
F159_MWRT_PARA													●	●	●	●		688
F16_DXCH	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	795
F160_DSQR			●	●	●	●	●	●	●	●		●	●	●	●	●	●	409
F161_MRCV	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	717

Befehl ● verfügbar ◑ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
F161_MRD_PARA													●	●	●	●		693
F161_MRD_STATUS													●	●	●	●		695
F165_HighSpeedCounter_Cam			●	●														871
F166_HighSpeedCounter_Set	●	●	●	●	●	●	●	●	●	●	●							878
F166_PulseOutput_Set			●	●														882
F167_HighSpeedCounter_Reset	●	●	●	●	●	●	●	●	●	●	●							885
F167_PulseOutput_Reset			●	●														889
F168_PulseOutput_Home	●	●									●							895
F168_PulseOutput_Trapezoidal	●	●									●							892
F169_PulseOutput_Jog	●	●									●							899
F17_SWAP	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	797
F170_PulseOutput_PWM	●	●									●							902
F171_PulseOutput_Home					●	●	●											911
F171_PulseOutput_Jog_Positioning			●	●														915
F171_PulseOutput_Trapezoidal Except FP-X16k L14|FP-X32k L30,L60			●	●	●	●	●	◑	●	●								905
F172_PulseOutput_Jog Except FP-X16k L14|FP-X32k L30,L60			●	●	●	●	●	◑	●	●								920
F173_PulseOutput_PWM Except FP-X16k L14|FP-X32k L30,L60			●	●	●	●	●	◑	●	●								926
F174_PulseOutput_DataTable Except FP-X16k L14|FP-X32k L30,L60			●	●	●	●	●	◑	●	●								929
F175_PulseOutput_Linear			●	●	●	●	●			●								933
F176_PulseOutput_Center					●	●												938
F176_PulseOutput_Pass					●	●												942
F177_PulseOutput_Home Except FP-X16k L14|FP-X32k L30,L60			●	●				◑	●	●								946
F178_HighSpeedCounter_Measure			●	●						●								950
F18_BXCH			●	●	●	●	●	●	●	●			●	●	●	●	●	799
F180_SCR_DUT											●							995
F181_DSP											●							999
F182_FILTER			●	●		●	●	●	●	●	●							534
F183_DSTM	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	974
F19_SJP												●	●	●	●	●	●	1033
F190_MV3			●	●	●	●	●	●	●	●			●	●	●	●	●	812
F191_DMV3			●	●	●	●	●	●	●	●			●	●	●	●	●	814
F2_MVN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	766

Befehl ● verfügbar ❍ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
F20_ADD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	336
F21_DADD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	338
F215_DAND			●	●	●	●	●	●	●	●			●	●	●	●	●	512
F216_DOR			●	●	●	●	●	●	●	●			●	●	●	●	●	514
F217_DXOR			●	●	●	●	●	●	●	●			●	●	●	●	●	516
F218_DXNR			●	●	●	●	●	●	●	●			●	●	●	●	●	518
F219_DUNI			●	●	●	●	●	●	●	●			●	●	●	●	●	520
F22_ADD2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	340
F23_DADD2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	342
F230_DTBCD_TO_SEC			●	●		●	●	●	●	●			●	●	●	●		836
F231_SEC_TO_DTBCD			●	●		●	●	●	●	●			●	●	●	●		837
F235_GRY			●	●	●	●	●	●	●	●			●	●	●	●	●	644
F236_DGRY			●	●	●	●	●	●	●	●			●	●	●	●	●	645
F237_GBIN			●	●	●	●	●	●	●	●			●	●	●	●	●	646
F238_DGBIN			●	●	●	●	●	●	●	●			●	●	●	●	●	647
F240_COLM			●	●	●	●	●	●	●	●			●	●	●	●	●	648
F241_LINE			●	●	●	●	●	●	●	●			●	●	●	●	●	650
F25_SUB	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	360
F250_BTOA			●	●		●	●	●	●	●								652
F251_ATOB			●	●		●	●	●	●	●								657
F252_ACHK			●	●		●	●	●	●	●								661
F26_DSUB	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	362
F27_SUB2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	364
F270_MAX			●	●	●	●	●	●	●	●	●		●	●	●	●	●	441
F271_DMAX			●	●	●	●	●	●	●	●	●		●	●	●	●	●	443
F272_MIN			●	●	●	●	●	●	●	●	●		●	●	●	●	●	447
F273_DMIN			●	●	●	●	●	●	●	●	●		●	●	●	●	●	449
F275_MEAN			●	●	●	●	●	●	●	●	●		●	●	●	●	●	453
F276_DMEAN			●	●	●	●	●	●	●	●	●		●	●	●	●	●	455
F277_SORT			●	●	●	●	●	●	●	●	●		●	●	●	●	●	488
F278_DSORT			●	●	●	●	●	●	●	●	●		●	●	●	●	●	490
F28_DSUB2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	366
F282_SCAL			●	●	●	●	●	●	●	●	●		●	●	●	●	●	459
F283_DSCAL			●	●	●	●	●	●	●	●	●		●	●	●	●	●	462
F284_RAMP			●	●		●	●	●	●	●								465
F285_LIMT			●	●	●	●	●	●	●	●			●	●	●	●	●	824
F286_DLIMT			●	●	●	●	●	●	●	●			●	●	●	●	●	826
F287_BAND			●	●	●	●	●	●	●	●	●		●	●	●	●	●	427
F288_DBAND			●	●	●	●	●	●	●	●	●		●	●	●	●	●	429
F289_ZONE			●	●	●	●	●	●	●	●	●		●	●	●	●	●	433
F290_DZONE			●	●	●	●	●	●	●	●	●		●	●	●	●	●	435
F3_DMVN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	768
F30_MUL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	384

Befehl ● verfügbar ◑ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
F300_BSIN													●	●	●	●	●	411
F301_BCOS													●	●	●	●	●	413
F302_BTAN													●	●	●	●	●	415
F303_BASIN													●	●	●	●	●	417
F304_BACOS													●	●	●	●	●	419
F305_BATAN													●	●	●	●	●	421
F309_FMV	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	816
F31_DMUL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	386
F310_FADD	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F311_FSUB	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F312_FMUL	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F313_FDIV	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	404
F314_SIN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F315_COS	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F316_TAN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F317_ASIN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F318_ACOS	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F319_ATAN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F32_DIV	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	396
F320_LN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F321_EXP	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F322_LOG	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F323_PWR	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F324_FSQR	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F325_FLT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	663
F326_DFLT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	664
F327_INT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	666
F328_DINT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	668
F329_FIX	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F33_DDIV	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	398
F330_DFIX	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F331_ROFF	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F332_DROFF	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F333_FINT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	670
F334_FRINT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	672
F335_FSIGN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	674
F336_FABS	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	1156
F337_RAD	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	676
F338_DEG	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	678
F34_MULW			●	●	●	●	●	●	●	●			●	●	●	●	●	388
F345_FCMP			●	●	●	●	●	●	●	●			●	●	●	●	●	1156
F346_FWIN			●	●	●	●	●	●	●	●			●	●	●	●	●	594
F347_FLIMT			●	●	●	●	●	●	●	●			●	●	●	●	●	1156

Befehl ● verfügbar ❍ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
F348_FBAND			●	●	●	●	●	●	●	●			●	●	●	●	●	431
F349_FZONE			●	●	●	●	●	●	●	●			●	●	●	●	●	437
F35_INC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	352
F350_FMAX													●	●	●	●	●	445
F351_FMIN													●	●	●	●	●	451
F352_FMEAN													●	●	●	●	●	457
F353_FSORT													●	●	●	●	●	492
F354_FSCAL			●	●		●	●	●	●	●			●	●	●	●	●	467
F355_PID_DUT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	980
F356_PID_PWM Except FP-X16k L14|FP-X32k L30,L60			●	●		●	●	❍	●	●								983
F36_DINC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	354
F37_DEC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	376
F373_DTR			●	●	●	●	●	●	●	●			●	●	●	●	●	596
F374_DDTR			●	●	●	●	●	●	●	●			●	●	●	●	●	598
F38_DDEC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	378
F39_DMULD			●	●	●	●	●	●	●	●			●	●	●	●	●	390
F4_GETS													●	●	●	●		770
F40_BADD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	344
F41_DBADD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	346
F42_BADD2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	348
F43_DBADD2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	350
F45_BSUB	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	368
F46_DBSUB	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	370
F47_BSUB2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	372
F48_DBSUB2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	374
F5_BTM	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	496
F50_BMUL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	392
F51_DBMUL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	394
F52_BDIV	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	400
F53_DBDIV	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	402
F55_BINC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	356
F56_DBINC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	358
F57_BDEC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	380
F58_DBDEC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	382
F6_DGT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	498
F60_CMP	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	584
F61_DCMP	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	586
F62_WIN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	588
F63_DWIN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	590
F64_BCMP	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	592
F65_WAN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	502
F66_WOR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	504
F67_XOR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	506

Befehl ● verfügbar ❍ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
F68_XNR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	508
F69_WUNI			●	●	●	●	●	●	●	●			●	●	●	●	●	510
F7_MV2			●	●	●	●	●	●	●	●			●	●	●	●	●	772
F70_BCC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	406
F71_HEX2A	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	602
F72_A2HEX	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	605
F73_BCD2A	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	608
F74_A2BCD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	611
F75_BIN2A	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	614
F76_A2BIN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	617
F77_DBIN2A	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	620
F78_DA2BIN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	623
F8_DMV2			●	●	●	●	●	●	●	●			●	●	●	●	●	774
F80_BCD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	626
F81_BIN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	628
F82_DBCD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	630
F83_DBIN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	632
F84_INV	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	529
F85_NEG	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	439
F86_DNEG	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	440
F87_ABS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	423
F88_DABS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	425
F89_EXT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	634
F90_DECO	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	636
F91_SEGT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	638
F92_ENCO	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	639
F93_UNIT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	530
F94_DIST	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	532
F95_ASC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	641
F96_SRC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	469
F97_DSRC			●	●	●	●	●	●	●	●			●	●	●	●	●	471
F98_CMPR			●	●	●	●	●	●	●	●		●	●	●	●	●	●	483
F99_CMPW			●	●	●	●	●	●	●	●		●	●	●	●	●	●	486
FIND	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	269
FNS_InitConfigDataTable					●	●							●	●	●	●		757
FNS_InitConfigNameTable					●	●							●	●	●	●		759
GET_RTC_DT		●		●	●	●	●			●	●	●	●	●	●	●	●	283
GET_RTC_DTBCD		●		●	●	●	●			●	●	●	●	●	●	●	●	838
GET_RTC_INT		●		●	●	●	●			●	●	●	●	●	●	●	●	1154
GetPointer	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
GT_ActivateScreen	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1058
GT_ChangeBacklightBrightness	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1060
Hsc_TargetValueMatch_Reset	●	●	●	●	●	●	●	●	●	●	●							1086

Befehl ● verfügbar ❍ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
Hsc_TargetValueMatch_Set	●	●	●	●	●	●	●	●	●	●	●							1088
HscControl_CountingDisable	●	●	●	●	●	●	●	●	●	●	●							1063
HscControl_CountingEnable	●	●	●	●	●	●	●	●	●	●	●							1064
HscControl_ElapsedValueContinue	●	●	●	●	●	●	●	●	●	●	●							1065
HscControl_ElapsedValueReset	●	●	●	●	●	●	●	●	●	●	●							1067
HscControl_HscInstructionClear	●	●	●	●	●	●	●	●	●	●	●							1069
HscControl_ResetInputDisable	●	●	●	●	●	●	●	●	●	●	●							1070
HscControl_ResetInputEnable	●	●	●	●	●	●	●	●	●	●	●							1071
HscControl_SetDefaults	●	●	●	●	●	●	●	●	●	●	●							1072
HscControl_WriteElapsedValue	●	●	●	●	●	●	●	●	●	●	●							1073
HscInfo_GetControlCode	●	●	●	●	●	●	●	●	●	●	●							1076
HscInfo_GetCurrentSpeed	●	●	●	●	●	●	●	●	●	●	●							1077
HscInfo_IsActive	●	●	●	●	●	●	●	●	●	●	●							1078
HscInfo_IsChannelEnabled	●	●	●	●	●	●	●	●	●	●	●							1079
HscInfo_IsCountingDisabled	●	●	●	●	●	●	●	●	●	●	●							1080
HscInfo_IsElapsedValueReset	●	●	●	●	●	●	●	●	●	●	●							1081
HscInfo_IsResetInputDisabled	●	●	●	●	●	●	●	●	●	●	●							1082
HscInfo_ReadElapsedValue	●	●	●	●	●	●	●	●	●	●	●							1083
HscInfo_ReadTargetValue	●	●	●	●	●	●	●	●	●	●	●							1084
ICTL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1037
INSERT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	271
INT_TO_BCD_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	241
INT_TO_BOOL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	116
INT_TO_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	172
INT_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	137
INT_TO_REAL	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	194
INT_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	218
INT_TO_STRING_LEADING_ZEROS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	220
INT_TO_TIME	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	202
INT_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	184
INT_TO_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	161
INT_TO_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	125
IPADDR_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	230
IPADDR_TO_STRING_NO_LEADING_ZEROS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	231
Is_AreaDT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
Is_AreaFL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
IS_VALID_DATE_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	284
IS_VALID_DT_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	285
IS_VALID_TOD_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	287
IsAnyHscChannelEnabled	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
IsAnyPulseChannelEnabled	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
IsClockCalendarSupported	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
IsCommunicationError	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	722

Befehl ● verfügbar ◑ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
IsDataUnitTypeSupported	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
IsF145F146Error			●	●		●	●			●		●	●	●	●	●	●	741
IsF145F146NotActive			●	●		●	●			●		●	●	●	●	●	●	740
IsFileRegisterAreaSupported	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
IsHscChannelEnabled	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
IsInstructionSupported	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
IsPlcLink	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	692
IsProgramControlled	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	691
IsPulseChannelEnabled	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
IsReceptionDone	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	712
IsReceptionDoneByTimeOut	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	**Fehler! Textmarke nicht definiert.**
IsSystemVariableSupported	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1196
IsTransmissionDone	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	707
JP	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1032
KEEP	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	842
LBL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1035
LEFT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	259
LEN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	258
LIMIT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	83
LN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	77
LOG	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	78
LOOP	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1034
LSR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	538
MAX	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	250
MC	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1030
MCE	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1031
MID	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	263
MIN	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	251
MOD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	67
MOVE	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	60
MUL_TIME_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	331
MUL_TIME_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	330
MUL_TIME_REAL	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	332
MUX	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	252
NOT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	89
P13_EPWT	●	●	●	●	●	●	●	●	●	●	●		●	●				792
PID_FB	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	988
PID_FB_DUT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	990

Befehl ● verfügbar ❍ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
Pulse_TargetValueMatch_Reset			●	●														1149
Pulse_TargetValueMatch_Set			●	●														1151
PulseControl_CountingDisable	●	●	●	●	●	●	●	●	●	●	●							1196
PulseControl_CountingEnable	●	●	●	●	●	●	●	●	●	●	●							1116
PulseControl_DeceleratedStop			●	●														1117
PulseControl_ElapsedValueContinue	●	●	●	●	●	●	●	●	●	●	●							1118
PulseControl_ElapsedValueReset	●	●	●	●	●	●	●	●	●	●	●							1120
PulseControl_JogPositionControl			●	●														1122
PulseControl_NearHome	●	●	●	●	●	●	●	●	●	●	●							1123
PulseControl_PulseOutputContinue	●	●	●	●	●	●	●	●	●	●	●							1125
PulseControl_PulseOutputStop	●	●	●	●	●	●	●	●	●	●	●							1126
PulseControl_SetDefaults	●	●	●	●	●	●	●	●	●	●	●							1127
PulseControl_WriteElapsedValue	●	●	●	●	●	●	●	●	●	●	●							1128
PulseInfo_GetControlCode	●	●	●	●	●	●	●	●	●	●	●							1132
PulseInfo_GetCurrentSpeed	●	●	●	●	●	●	●	●	●	●	●							1133
PulseInfo_IsActive	●	●	●	●	●	●	●	●	●	●	●							1135
PulseInfo_IsChannelEnabled	●	●	●	●	●	●	●	●	●	●	●							1136
PulseInfo_IsCountingDisabled	●	●	●	●	●	●	●	●	●	●	●							1137
PulseInfo_IsElapsedValueReset	●	●	●	●	●	●	●	●	●	●	●							1138
PulseInfo_IsHomeInputTrue	●	●	●	●	●	●	●	●	●	●	●							1139
PulseInfo_IsPulseOutputStopped	●	●	●	●	●	●	●	●	●	●	●							1140
PulseInfo_IsTargetValueMatchActive			●	●														1141
PulseInfo_ReadAccelerationForbiddenAreaStartingPosition			●	●														1142
PulseInfo_ReadCorrectedFinalSpeed			●	●					●	●								1143
PulseInfo_ReadCorrectedInitialSpeed			●	●					●	●								1144
PulseInfo_ReadElapsedValue	●	●	●	●	●	●	●	●	●	●	●							1145
PulseInfo_ReadTargetValue	●	●	●	●	●	●	●	●	●	●	●							1146
PulseInfo_ReadTargetValueMatchValue			●	●														1147
PulseOutput_Center_FB					●	●												1093
PulseOutput_Home_FB	●	●	●	●	●	●	●	●	●	●	●							1095
PulseOutput_Jog_FB	●	●	●	●	●	●	●	●	●	●	●							1098
PulseOutput_Jog_Positioning0_FB			●	●														1100
PulseOutput_Jog_Positioning1_FB			●	●														1102
PulseOutput_Jog_TargetValue_FB Except FP-X16k L14|FP-X32k L30,L60			●	●	●	●	●	❍	●	●								1104
PulseOutput_Linear_FB			●	●	●	●	●			●								1106
PulseOutput_Pass_FB					●	●												1109
PulseOutput_Trapezoidal_FB	●	●	●	●	●	●	●	●	●	●	●							1111
R_TRIG	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	306
ReadDataFromFileRegisterBank																●		819
REAL_TO_DINT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	175
REAL_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	141

Befehl ● verfügbar ◑ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
REAL_TO_INT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	152
REAL_TO_STRING	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	226
REAL_TO_TIME	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	204
REAL_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	187
REAL_TO_UINT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	164
ReceiveCharacters	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	716
ReceiveData	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	715
REPLACE	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	273
RIGHT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	261
ROL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	98
ROR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	96
RS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	302
RST	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	843
SCALE_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SCALE_INT_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SCALE_REAL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SCALE_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SCALE_UINT_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SCLR_CLEAR_MULTIPLE_STEPS			●	●	●	●	●	●	●	●			●	●	●	●	●	1196
SEL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	254
SendCharacters	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	699
SendCharactersAndClearString	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	701
SET	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	843
SET_RTC_DT		●		●	●	●	●			●	●	●	●	●	●	●	●	288
SET_RTC_DTBCD		●		●	●	●	●			●	●	●	●	●	●	●	●	839
SET_RTC_INT		●		●	●	●	●			●	●	●	●	●	●	●	●	1154
SetCommunicationMode	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	684
SfcOutputsReset	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SfcRunning	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SfcStopped	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SfcTransitionsInhibited	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SHL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	94
SHR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	92
SIN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	69
Size_Of_Var	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SmoothSignal_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SmoothSignal_REAL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SmoothSignal_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
SPLIT_DATE_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	289
SPLIT_DT_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	290
SPLIT_TIME_INT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	333
SPLIT_TOD_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	292
SQRT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	68

Befehl ● verfügbar ❍ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
SR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	300
StartStopAllSfcs	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
StartStopAllSfcsAndInitData	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
StartStopSfc	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
StartStopSfcAndInitData	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
STRING_TO_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	178
STRING_TO_DINT_STEPSAVER	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
STRING_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	143
STRING_TO_DWORD_STEPSAVER	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	144
STRING_TO_ETLANADDR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	247
STRING_TO_ETLANADDR_STEPSAVER	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	248
STRING_TO_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	155
STRING_TO_INT_STEPSAVER	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	156
STRING_TO_IPADDR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	245
STRING_TO_IPADDR_STEPSAVER	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	246
STRING_TO_REAL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	199
STRING_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	189
STRING_TO_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	166
STRING_TO_UINT_STEPSAVER	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	167
STRING_TO_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	130
STRING_TO_WORD_STEPSAVER	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	131
SUB_DATE_DATE	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	293
SUB_DT_DT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	294
SUB_DT_TIME	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	295
SUB_TIME	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	334
SUB_TOD_TIME	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	296
SUB_TOD_TOD	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	297
SYS1			●	●	●	●	●	●	●	●								1014
SYS2			●	●	●	●	●			●								1017
TAN	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	73
TIME_TO_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	177
TIME_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	142
TIME_TO_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	154
TIME_TO_REAL	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	198
TIME_TO_STRING			●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	228
TIME_TO_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	129
TM_100ms	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	969
TM_100ms_FB	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	959
TM_10ms	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	967
TM_10ms_FB	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	956
TM_1ms	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	965
TM_1ms_FB	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	954
TM_1s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	971

Befehl ● verfügbar ◑ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
TM_1s_FB	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	962
TOD_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	234
TOD_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	192
TOF	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	318
TON	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	320
TP	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	322
TRUNC_TO_DINT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	176
TRUNC_TO_INT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	153
TRUNC_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	188
TRUNC_TO_UINT	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	165
UDINT_TO_BCD_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	244
UDINT_TO_BOOL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	119
UDINT_TO_DATE	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	207
UDINT_TO_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	174
UDINT_TO_DT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	205
UDINT_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	140
UDINT_TO_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	151
UDINT_TO_REAL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	197
UDINT_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
UDINT_TO_STRING_LEADING_ZEROS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
UDINT_TO_TOD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	209
UDINT_TO_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	163
UDINT_TO_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	128
UINT_TO_BCD_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	243
UINT_TO_BOOL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	118
UINT_TO_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	173
UINT_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	139
UINT_TO_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	150
UINT_TO_REAL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	196
UINT_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	224
UINT_TO_STRING_LEADING_ZEROS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	225
UINT_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	185
UINT_TO_WORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	127
Unit_AnalogInOut_FP0_A21	●	●	●	●	●	●	●	●	●	●								1040
Unit_AnalogInOut_FPG_A44					●	●												1049
Unit_AnalogInput_FP0_A80	●	●	●	●	●	●	●	●	●	●								1041
Unit_AnalogInput_FP0_RTD_INT	●	●	●	●	●	●	●	●	●	●								1043
Unit_AnalogInput_FP0_RTD_REAL	●	●	●	●	●	●	●	●	●	●								1045
Unit_AnalogInput_FP0_TC4_TC8	●	●	●	●	●	●	●	●	●	●								1047
Unit_AnalogOutput_FP0_A04I	●	●	●	●	●	●	●	●	●	●								1051
Unit_AnalogOutput_FP0_A04V	●	●	●	●	●	●	●	●	●	●								1053
Var_ToAreaOffs	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154

Befehl ● verfügbar ❍ teilweise verfügbar	FP0 2,7k/5k	FP0 10k T32	FP0R 16k/32k	FP0R 32k T32	FPΣ 12k	FPΣ 32k	FP-X 16k/32k R-Typen	FP-X 16k/32k T-Typen	FP-X0 2,5k L14/L30	FP-X0 8k L40/L60	FP-e 2,7k	FP3, FP-C	FP2 16K	FP2 32k	FP2SH 32k	FP2SH 60k/120k	FP10SH	Seite
WITHIN_LIMITS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	1154
WORD_BCD_TO_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	147
WORD_BCD_TO_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	159
WORD_TO_BOOL	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	114
WORD_TO_BOOL16	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	235
WORD_TO_BOOLS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	237
WORD_TO_DINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	169
WORD_TO_DWORD	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	136
WORD_TO_INT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	146
WORD_TO_STRING	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	212
WORD_TO_TIME	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	200
WORD_TO_UDINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	181
WORD_TO_UINT	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	158
WriteDataToFileRegisterBank																●		821

Index

R

S

T

U

W

X

Änderungsverzeichnis

Handbuchnummer	Datum	Änderungen
ACGM0313V3DE	Dez. 2012	Erste Ausgabe für FPWIN Pro 6.410

North America | **Europe** | **Asia Pacific** | **China** | **Japan**

Panasonic Electric Works Global Sales Companies

Europe

▶ Headquarters	Panasonic Electric Works Europe AG	Rudolf-Diesel-Ring 2, 83607 Holzkirchen, Tel. +49 (0) 8024 648-0, Fax +49 (0) 8024 648-111, www.panasonic-electric-works.com
▶ Austria	Panasonic Electric Works Austria GmbH	Josef Madersperger Str. 2, 2362 Biedermannsdorf, Tel. +43 (0) 2236-26846, Fax +43 (0) 2236-46133, www.panasonic-electric-works.at
	Panasonic Industrial Devices Materials Europe GmbH	Ennshafenstraße 30, 4470 Enns, Tel. +43 (0) 7223 883, Fax +43 (0) 7223 88333, www.panasonic-electronic-materials.com
▶ Benelux	Panasonic Electric Works Sales Western Europe B.V.	De Rijn 4, (Postbus 211), 5684 PJ Best, (5680 AE Best), Netherlands, Tel. +31 (0) 499 372727, Fax +31 (0) 499 372185, www.panasonic-electric-works.nl
▶ Czech Republic	Panasonic Electric Works Czech s.r.o.	Administrative centre PLATINIUM, Veveri 111, 616 00 Brno, Tel. (+420)541 217 001, Fax (+420)541 217 101, www.panasonic-electric-works.cz
▶ France	Panasonic Electric Works Sales Western Europe B.V.	Succursale française, 10, rue des petits ruisseaux, 91371 Verrières le Buisson, Tél. +33 (0) 1 6013 5757, Fax +33 (0) 1 6013 5758, www.panasonic-electric-works.fr
▶ Germany	Panasonic Electric Works Europe AG	Rudolf-Diesel-Ring 2, 83607 Holzkirchen, Tel. +49 (0) 8024 648-0, Fax +49 (0) 8024 648-111 www.panasonic-electric-works.de
▶ Hungary	Panasonic Electric Works Europe AG	Magyarországi Közvetlen Kereskedelmi Képviselet, 1117 Budapest, Neumann János u. 1., Tel. +36(0)1482 9258, Fax +36 (0) 1482 9259, www.panasonic-electric-works.hu
▶ Ireland	Panasonic Electric Works UK Ltd.	Dublin, Tel. +353 (0) 14600969, Fax +353 (0) 14601131, www.panasonic-electric-works.co.uk
▶ Italy	Panasonic Electric Works Italia s.r.l.	Via del Commercio 3-5 (Z.I. Ferlina), 37012 Bussolengo (VR), Tel. +39 (0) 456752711, Fax +39 (0) 456700444, www.panasonic-electric-works.it
▶ Nordic Countries	Panasonic Electric Works Nordic AB	Filial Nordic, Knarrarnäsgatan 15, 16440 Kista, Sweden, Tel. +46 859476680, Fax +46 859476690, www.panasonic-electric-works.se
	Panasonic Eco Solutions Nordic AB	Jungmansgatan 12, 21119 Malmö, Tel. +46 40697-7000, Fax +46 40697-7099, www.panasonic-fire-security.com
▶ Poland	Panasonic Electric Works Polska sp. z o.o.	Al. Krakowska 4/6, 02-284 Warszawa, Tel. +48 (0) 22 338-11-33, Fax +48 (0) 22 338-12-00, www.panasonic-electric-works.pl
▶ Portugal	Panasonic Electric Works España S.A.	Portuguese Branch Office, Avda Adelino Amaro da Costa 728 R/C J, 2750-277 Cascais, Tel. +351 214812520, Fax +351 214812529
▶ Spain	Panasonic Electric Works España S.A.	Barajas Park, San Severo 20, 28042 Madrid, Tel. +34 913293875, Fax +34 913292976, www.panasonic-electric-works.es
▶ Switzerland	Panasonic Electric Works Schweiz AG	Grundstrasse 8, 6343 Rotkreuz, Tel. +41 (0) 417997050, Fax +41 (0) 417997055, www.panasonic-electric-works.ch
▶ United Kingdom	Panasonic Electric Works UK Ltd.	Sunrise Parkway, Linford Wood, Milton Keynes, MK14 6LF, Tel. +44(0) 1908 231555, +44(0) 1908 231599, www.panasonic-electric-works.co.uk

North & South America

▶ USA	Panasonic Industrial Devices Sales Company of America	629 Central Avenue, New Providence, N.J. 07974, Tel. +1-908-464-3550, Fax +1-908-464-8513, www.pewa.panasonic.com

Asia Pacific / China / Japan

▶ China	Panasonic Electric Works (China) Co., Ltd.	Level 2, Tower W3, The Tower Oriental Plaza, No. 2, East Chang An Ave., Dong Cheng District, Beijing 100738, Tel. +86-10-5925-5988, Fax +86-10-5925-5973
▶ Hong Kong	Panasonic Industrial Devices Automation Controls Sales (Hong Kong) Co., Ltd.	RM1205-9, 12/F, Tower 2, The Gateway, 25 Canton Road, Tsimshatsui, Kowloon, Hong Kong, Tel. +852-2956-3118, Fax +852-2956-0398
▶ Japan	Panasonic Corporation	1048 Kadoma, Kadoma-shi, Osaka 571-8686, Japan, Tel. +81-6-6908-1050, Fax +81-6-6908-5781, www.panasonic.net
▶ Singapore	Panasonic Industrial Devices Automation Controls Sales Asia Pacific Pte. Ltd.	300 Beach Road, #16-01 The Concourse, Singapore 199555, Tel. +65-6390-3811, Fax +65-6390-3810

ACGM0313V3DE 12/2012