9. Handhabung von Funktionsbausteinen (Funktionsblock, FB)

-> hierzu auch Folge 9

9.1 Übersicht und Regeln

Funktionsbausteine (FB) werden auch als Funktionsblöcke bezeichnet. Es sind komplexe Programmorganisationseinheiten, die alle Freiheiten der Programmierung nach IEC 61131-3 erlauben. Das Schlüsselwort der POE ist FUNCTION_BLOCK.

Funktionsbausteine (Funktionsblocks) können mehrere Ausgangsvariablen besitzen und für gleiche Eingangswerte durchaus unterschiedliche Ausgangswerte zulassen. Das ist möglich, weil sie auch interne Variablen besitzen, deren Werte über den Aufruf der POE hinaus erhalten bleiben..

Funktionsbausteine sind parametrierbar, wobei Eingangs-, Ausgangs- und innere Variablen benutzt werden können.

Funktionsbausteinen können von Programmen und anderen FB's aus aufgerufen werden. Der Aufruf erfolgt durch ihre **Instanzierung**. Die Instanz ist vergleichbar mit einer Kopie des deklarierten FB für einen speziellen Anwendungsfall. In der Deklaration wird die Instanz mit Ihrem Typ vereinbart. Damit wird für jede Instanz der notwendige Speicherbereich zur Verfügung gestellt.

Zwischen den Aufrufen des Bausteins werden die Daten gespeichert. Diese Tatsache wird landläufig als „Gedächtnis" des FB beschrieben.

Typische Funktionsbausteine wie Zähler, Zeitgeber, Bistabile Elemente oder Trigger stehen als **Standard-FB** in den Bibliotheken der Programmiersystemen zur Verfügung. Häufig benötigte Programmteile kann der Anwender selbst als parametrierbaren Funktionsbaustein definieren.

9.2 Standard-Funktionsbausteine

Eine Vielzahl häufig benötigter Lösungen der Automatisierungstechnik stehen als Standard-Funktionen und Standard-Funktionsbausteine in Bibliotheken bereit oder aber sind bereits in die Programmiersysteme eingearbeitet.
Die Entscheidung, ob ein Teilproblem als Funktion oder als Funktionsbaustein hinterlegt werden muss,
hängt übereinstimmend mit den Grundeigenschaften der POE von den benötigten inneren Variablen ab.

Im System CoDeSys sind in der Standardbibliothek (standard.lib) folgende Standard-Funktionsbausteine hinterlegt:

- Bistabile Multivibratoren: SR, RS
- Timer : TON, TOF, TP
- Zähler: CTU, CTD, CTUD
- Trigger: F_TRIG, R_TRIG
- Diagnostic

Eine Vielzahl weiterer FB stehen in anderen Bibliotheken bereit. Voraussetzung ihrer Nutzung ist stets, dass über die Bibliotheksverwaltung die **Bibliotheken in das Projekt eingebunden** wurden, d.h. bekannt sind.

In den Abschnitten 3 und 4 wurden solche Bausteine bereits eingesetzt. Hier sollen nun die Kenntnisse über FB erweitert werden, damit auch **selbstdefinierte Funktionsbausteine** erstellt werden können.
Bild 9-1 zeigt rechts unten die Auswahlliste der Standard-Funktionsbausteine in der Sprache FUP. Im oberen Teil ist ersichtlich, wie die Anzahl verfügbarer FB wächst, wenn weitere Bibliotheken in ein Projekt eingefügt werden. Im Beispiel ist dies die Bibliothek util.lib.

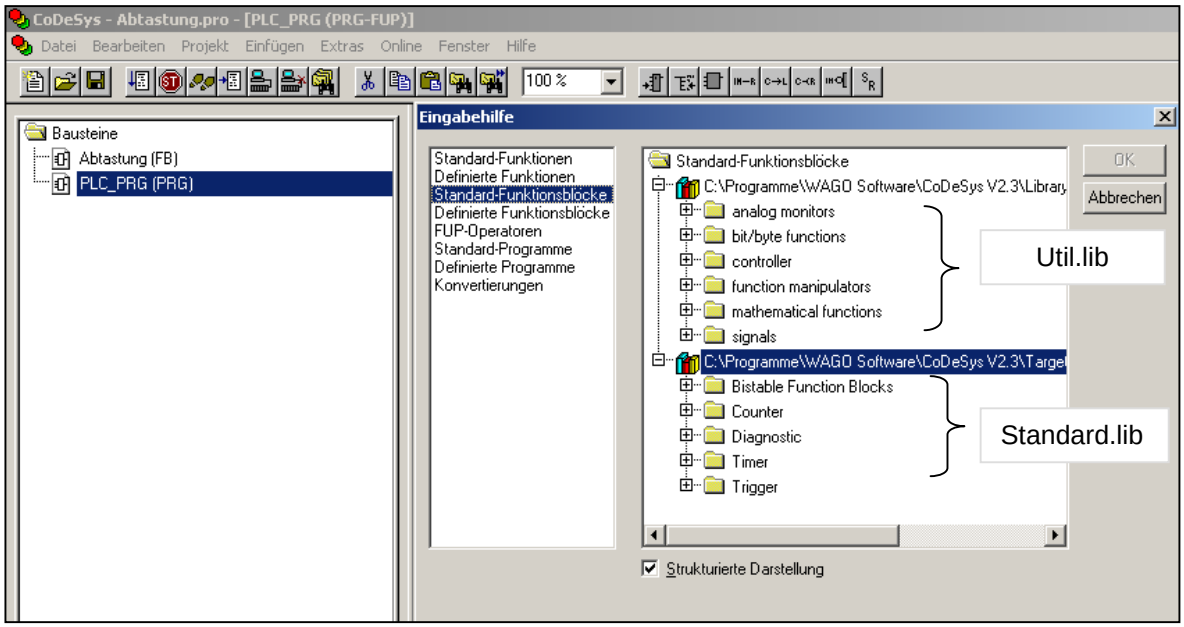


Bild 9-1: Verfügbare Standardfunktionsbausteine im System CoDeSys, oben aus Bibliothek util.lib, unten aus Standardbibliothek standard.lib

- **Nutzung von Standard-Funktionsbausteinen im FUP**

Die Einbindung der Bausteine in ein Programm erfolgt im graphischen FUP durch Anklicken der gewünschten FB im Verzeichnis nach Markierung eines bereits eingetragenen allgemeinen Blocks. Hilfreich ist die Eingabehilfe Taste F2.

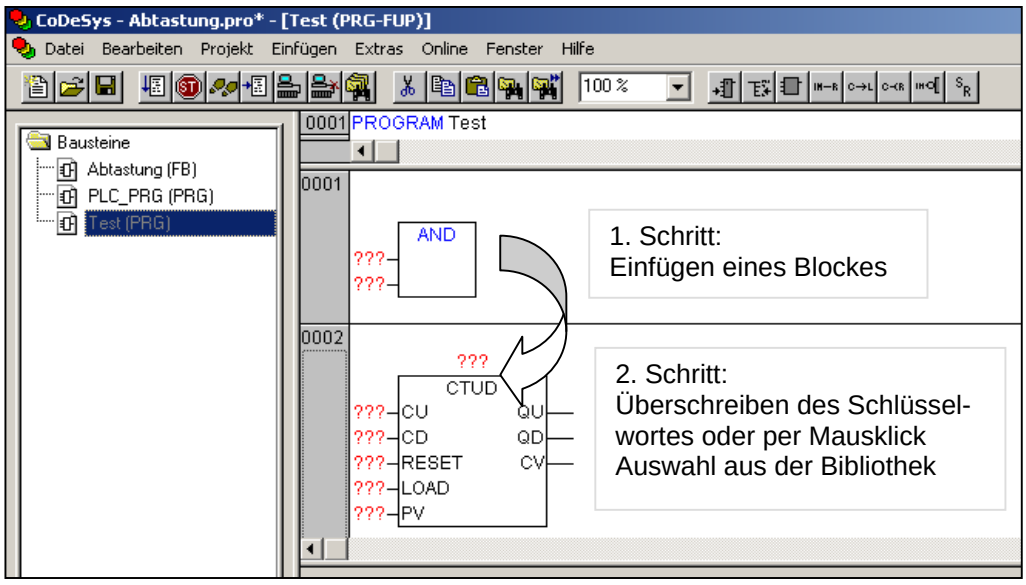


Bild 9-2: Aufruf eines Standard-FB vom Typ Vor-Rückwärtszähler CTUD in FUP

In FUP erkennt man die Standard-FB deutlich als **parametrierbare Bausteine**. Nach deren Regeln erscheinen links Variablen vom Typ INPUT und IN_OUT und rechts Variablen vom Typ OUTPUT. Die Fragezeichen über dem Block verlangen den Eintrag eines Instanznamens.

Nach dem Aufruf müssen
1. die FB instanziert und
2. die aktuellen Variablen für den konkreten Anwendungsfall angetragen werden.

Bild 9-3 zeigt die Ausführung beider Schritte bei unterschiedlichen Zählern.

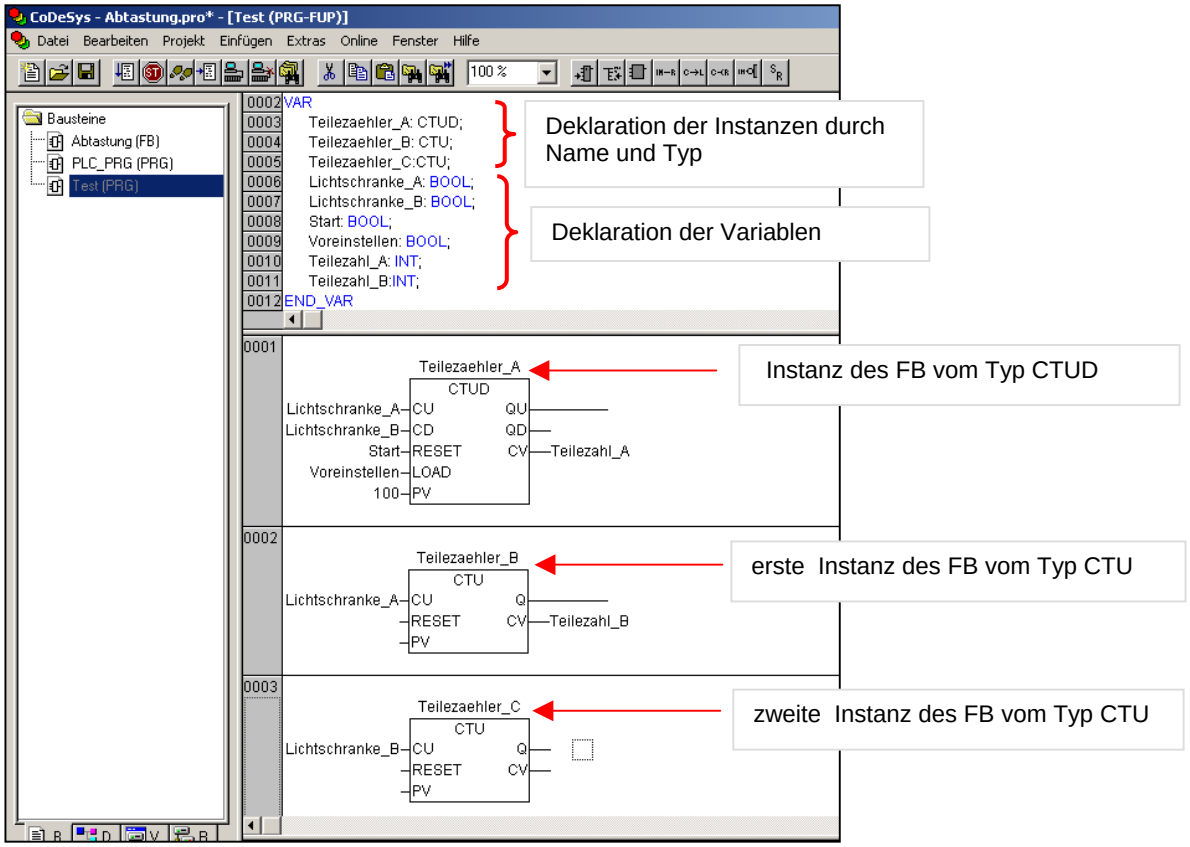


Bild 9-3: Beispiel für Instanzierung und Deklaration erforderlicher Variablen von FB

Bei der **Instanzierung** wird de facto eine Kopie des Standard-Funktionsbausteins angefertigt und der dafür erforderliche Raum im Datenspeicher reserviert. Diese Kopie wird dann mit ihrem Namen angesprochen. In der Deklaration wird diesem Namen der Typ des Standard-FB zugeordnet.

In einem Programm können **beliebig viele** Instanzen eines FB benutzt werden, alle unterschieden durch ihren Instanznamen.

Bei der **Variablendeklaration** fällt auf, dass **nicht alle** Variablen des parametrierbaren Blockes benutzt werden müssen. Es werden nur die deklariert, die im konkreten Anwendungsfall von Interesse sind.

- **Nutzung von Standard-Funktionsbausteinen in AWL**

In AWL ruft man FB's mit dem Befehl CAL und dem Namen ihrer **Instanz** auf. Voraussetzung ist selbstverständlich auch hier, dass die entsprechenden Bibliotheken in das Projekt eingebunden wurden. Im Kopf der POE ist dem Namen der Instanz der Typ des Standard-FB zuzuordnen.

Bei der Anschaltung der aktuellen Variablen in die Parameter vom Typ INPUT oder IN_OUT gibt es zwei Möglichkeiten:

1. Vor dem Aufruf der Instanz werden die Variablen mit dem Befehl STORE (ST) in die Parameter eingetragen (zugewiesen, de facto gespeichert)

2. Beim Aufruf der Instanz werden die Parameter mit den Variablen initialisiert (Zeichen :=).

Beide Verfahren können gemischt werden. Die Parameter vom Typ OUTPUT werden mit dem Befehl STORE (ST) mit den entsprechenden Variablen verbunden, oder sie werden mit ihrem Namen in späteren Anweisungen benutzt.

Grundsätzlich können alle Variablen des FB mit dem Namen der Instanz und dem Namen des Parameter, getrennt durch einen Punkt, angesprochen werden.
Beispiele (siehe Bild 9-3):
Teilezahler_A.RESET
Teilezähler_A.Start
Teilezahler_A.CV
Teilezahler_B.CU
Teilezahler_B.CV

Bei gleicher Deklaration der Variablen lautet die Nutzung des Standard-FB's CTUD in AWL in einem Programm:

```
PROGRAM Test
VAR
Teilezaehler_A:CTUD;
Lichtschranke_A:BOOL;
Lichtschranke_B:BOOL;
Start:BOOL;
Voreinstellen:BOOL;
Teilezahl_A:INT;
END_VAR

Methode 1:  LD   Lichtsschranke_A
            ST   Teilezaehler_A.CU
            LD   Lichtschranke_B
            ST   Teilezaehler_A.CD
            LD   Start
            ST   Teilezaehler_A.RESET
            LD   Voreinstellen
            ST   Teilezaehler_A.LOAD
            LD   100
            ST   Teilezaehler_A.PV
            CAL  Teilezaehler_A
            LD   Teilezaehler_A.CV
            ST   Teilezahl_A

Methode 2:  CAL  Teilezaehler_A (CU:=Lichtschranke_A, CD:=Lichtschranke_B,
                 RESET:=Start, LOAD:=Voreinstellen,PV:=100)
            LD   Teilezaehler_A.CV
            ST   Teilezahl_A

Beide Methoden können gemischt angewendet werden.
```

9.3 Selbstdefinierte Funktionsbausteine

Nachfolgende Aufgabe soll beispielhaft mit einem parametrierbaren Funktionsbaustein gelöst werden. Das Programm könnte alternativ auch in eine POE vom Typ PROGRAM geschrieben werden.

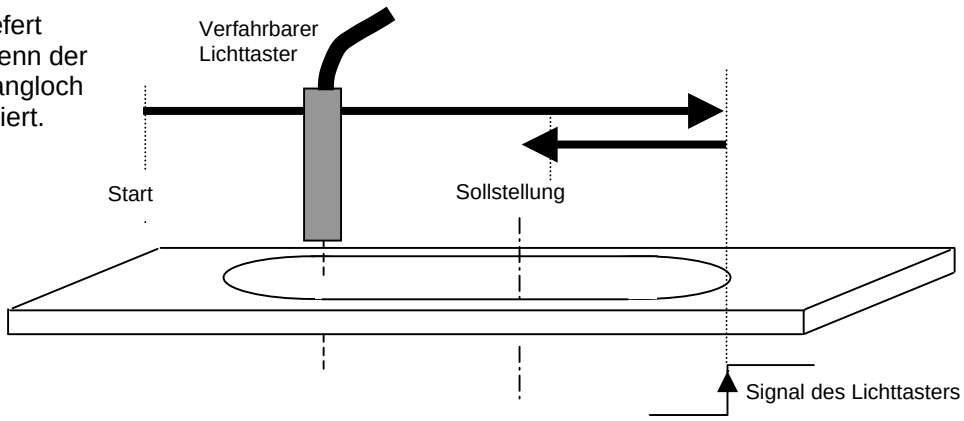
Warum **parametrierbarer** Funktionsbaustein?

Ein parametrierbarer FB ist dann von Vorteil, wenn das vorliegende Detail einer Steuerung mehrfach Verwendung findet. Es wird wie ein Bibliotheksbaustein ohne Hardwarebezug geschrieben. Der Bezug zu den konkreten Ein- und Ausgabebaugruppen wird erst bei der Instanzierung des FB hergestellt.

Aufgabe:

Ein Lichttaster soll im Rechtslauf ein von Werkstück zu Werkstück veränderliches Langloch abtasten und nach einer Zwangs-Umschaltpause im Linkslauf in der Mitte desselben positioniert werden. Pro Längeneinheit des Verfahrweges werden eine bestimmte Anzahl Inkremente erzeugt.

Der Lichttaster liefert FALSE-Signal, wenn der Lichtstrahl das Langloch ungehindert passiert.



Eingangsparameter:	Ausgangsparameter:
Startsignal für Verfahren nach rechts	Signal Verfahren nach rechts
Signal des Lichttasters	Signal Verfahren nach links
Signal der Inkremente	

Lösungsweg:

Derartige Aufgaben werden heute mit hochauflösenden Encodern und schnellen lagegeregelten Achsantrieben gelöst. Dann muss man sich beispielsweise auch mit der zulässigen Impulsfrequenz der Encoder befassen. Hier soll dagegen eine einfachste Lösung gefunden werden, die auch am PC simuliert werden kann.

Nach Start verfährt der Lichttaster nach rechts. Erreicht er das Langloch, werden Inkremente aufgezählt. Entscheidend für die Steuerung ist der positiver Flankenwechsel des Lichttaster-Signals. Hier schaltet der Antrieb ab und nach einer Pause Linkslauf ein. Die Zahl der Inkremente am Umschaltpunkt wird halbiert, und um diesen halben Wert wird der Lichttaster zurückgefahren.

In diesem Beispiel wird nicht berücksichtigt, wie der Lichttaster nach Halt wieder zurück in die Startposition gelangt.

Für Simulation und Test des Programms wird ein Impulsgeber 1:1 in das Programm integriert.

- **Variablendeklaration:**

Der selbstdefinierte FB erhält den Namen „Abtastung“ Es werden in ihn eine Reihe von Standard-FB’s eingebaut, deren Instanzen selbstverständlich als interne Variablen zu deklarieren sind.
Die Zahl der anderweitigen internen Variablen (hier „Impulszahl“ und „Soll_Impulszahl“), die als Zwischenergebnisse fungieren, hängt auch von der Art der Programmierung und von den Möglichkeiten der graphischen Editoren ab.

Die Variablendeklaration für den FB lautet:

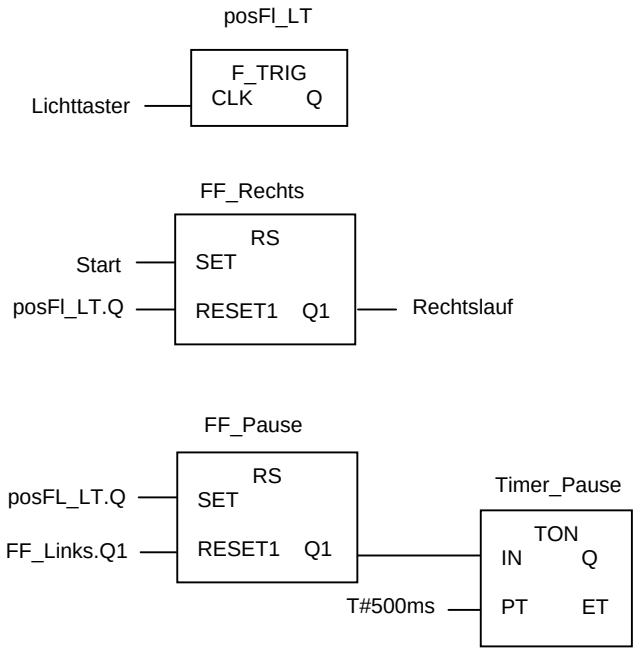
```
FUNCTION_BLOCK Abtastung

VAR_INPUT
Start:BOOL;                 (*Startsignal*)
Lichttaster:BOOL;           (*Signal des Lichttaster, im Langloch Wert FALSE*)
Impulsgeber:BOOL;           (*Signal des Impulsgebers, gekoppelt mit dem Antrieb*)
END_VAR

VAR_OUTPUT
Rechtslauf:BOOL;            (*Steuersignal für Antrieb Rechtslauf*)
Linkslauf:BOOL;             (*Steuersignal für Antrieb Linkslauf*)
END_VAR

VAR
FF_Rechts: RS;
FF_Links: RS;
FF_Pause: RS;               (*Benötigte Instanzen von Standard-Funktionsbausteinen*)
Timer_Pause: TON;
posFl_LT: F_TRIG;
Impulszaehler: CTUD;
Impulszahl: INT;
Soll_Impulszahl: INT;
END_VAR
```

- **Programmcode in FUP:**

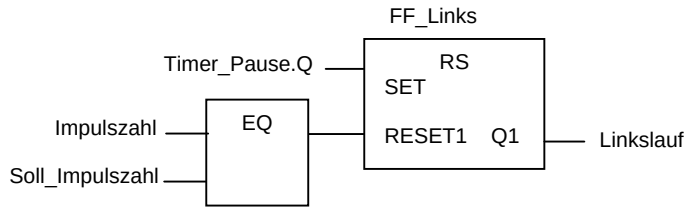


Kommentar:

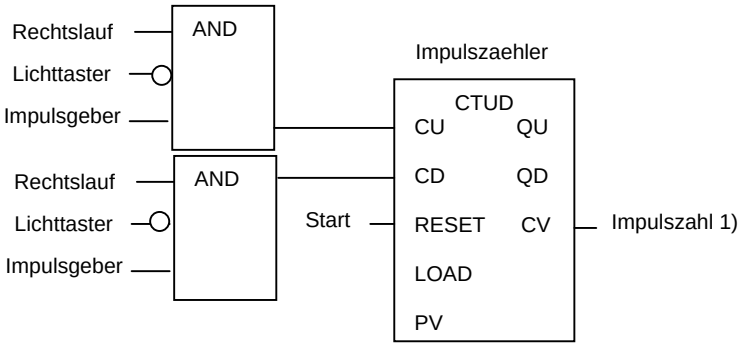
Positive Flankenauswertung des Signals des Lichttasters.
Der 1-Impuls bei positiver Flanke steht mit posFl_LT.Q im weiteren Programm zur Verfügung.

Steuerung Antrieb Rechtslauf nach Start. Da das Startsignal kein Dauersignal ist, muss ein S-R-Flip-Flop eingesetzt werden. Bei Flankenwechsel des Lichttaster-Signals wird gestoppt.

Mit dem Flankenwechsel wird ein weiteres Flip-Flop gesetzt, welches das Start-Signal einer Einschaltverzögerung liefert.
Dieses Flip-Flop wird bei Einschaltung des Linkslaufes zurückgesetzt.

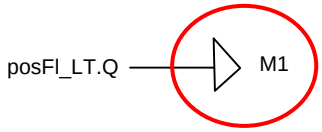


Nach Ablauf der Umschaltpause (Signal Timer_Pause.Q = TRUE) wird das Flip-Flop für die Steuerung des Linkslaufs des Antriebs gesetzt. Linkslauf wird zurückgesetzt, wenn die Hälfte der Inkremente = Sollimpulszahl verfahren wurde.

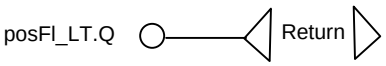


Vor-Rückwärts-Zähler für die Impulse (Inkremente):

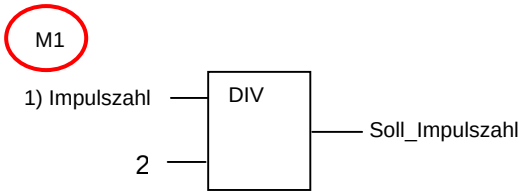
Jeweils bei Start erfolgt ein Rücksetzen des Zählers auf Null. Impulse werden nur bei freiem Lichttaster gezählt.
Bei Rechtslauf werden die Impulse vorwärts aufgezählt, bei Linkslauf wird rückwärts gezählt.



Bei positivem Flankenwechsel des Lichttaster-Signal (es entsteht ein 1-Impuls von Zykluszeit) erfolgt ein Sprung zur Marke M1. Dort wird die halbe Impulszahl als Soll_Impulszahl für Linkslauf bereitgestellt.



In allen anderen Fällen wird das Programm beendet und in einem neuen Zyklus bearbeitet.



Berechnung der Soll-Impulszahl als halber Wert der bei Rechtslauf abgezählten Impulse

1) Auf die Variable Impulszahl:INT kann verzichtet werden, wenn stattdessen der Ausgang des Impulszaehlers mit Impulszaehler.CV angesprochen wird.

- **Programmcode in AWL**

```
CAL     posFl_LT(CLK:=Lichttaster)                  (*positive Flankenauswertung  des Lichtaster-Signals*)

CAL     FF_Rechts(SET:=Start,RESET1:=posFl_LT.Q)    (*FlipFlop Rechtslauf*)
LD      FF_Rechts.Q1
ST      Rechtslauf

CAL     FF_Pause(SET:=posFl_LT.Q, RESET1:=FF-Links.Q1)   (*FlipFlop Pausenzeit*)

CAL     Timer_Pause(IN:=FF_Pause.Q1,PT=T#500ms)     (*Einschaltverzögerung TON*)

LD      Impulszahl
EQ      Soll_Impulszahl
ST      FF_Links.RESET1
CAL     FF_Links(SET:=Timer_Pause.Q)                (*FlipFlop Linkslauf*)
LD      FF_Links
ST      Linkslauf

LD      Rechtslauf
AND     Impulsgeber
ANDN    Lichttaster
ST      ST Impulszaehler.CU
LD      Linkslauf                                   (*Impulszaehler*)
AND     Impulsgeber
ANDN    Lichttaster
ST      ST Impulszaehler.CU
CAL     Impulszaehler(RESET:=Start)

LD      posFl_LT.Q                                  (*Mit 1-Impuls bei positiverFlanke des Lichttaster-Signals Sprung zu Marke M1, sonst Absprung aus der POE *)
JMP     M1
LDN     posFl_LT.Q
RETC

M1 :                                                (*Bei Marke M1 Berechnung der Soll-Impulszahl fürLinkslauf. Indem hier der Ausgang des Impulszaehlers mit Impulszaehler.CV angesprochen wird, wurde auf die Variable Impulszahl verzichtet! *)
LD      Impulszaehler.CV
DIV     2
ST      Soll_Impulszahl
RET
```

- **Aufruf des selbstdefinierten FB in der POE PLC_PRG**

Damit der parametrierbare Funktionsblock „Abtastung“ zyklisch bearbeitet wird, muss er selbstverständlich in der POE PLC_PRG aufgerufen werden. Dies soll nachfolgend für zwei derartige Applikationen Achse_A und Achse_B erfolgen. Das bedeutet, dass zwei Instanzen des FB zu bilden sind. Diese Instanzen sind mit Namen und Typ „Abtastung“ zu deklarieren.

Beim graphischen Aufruf des selbstdefinierten FB erscheinen den Regeln gemäß Variablen des FB vom Typ INPUT links und vom Typ OUTPUT rechts im Blocksymbol der POE. Hier werden die aktuellen Variablen der konkreten Applikation angetragen.

Weiter wird in der POE PLC_PRG die Simulation des Impulsgebers programmiert.

- **Variablendeklaration:**

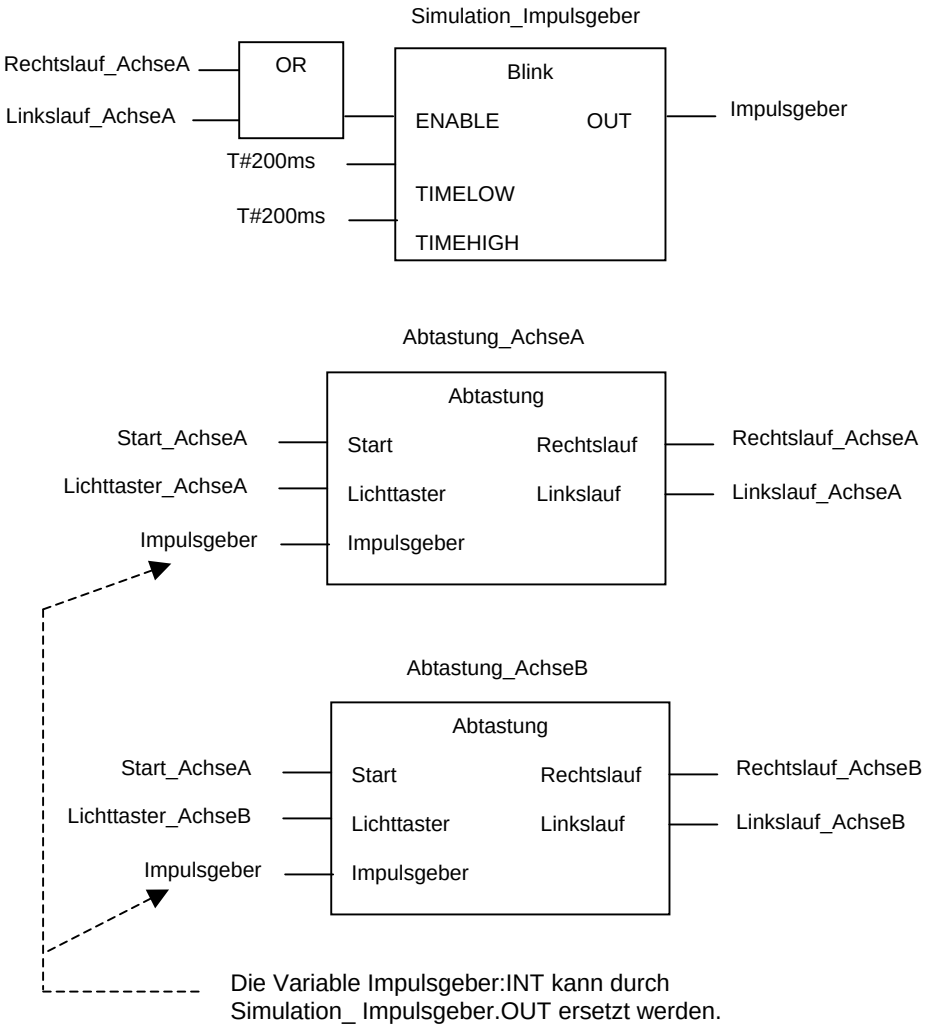
```
PROGRAM PLC_PRG
VAR
Abtastung_AchseA: Abtastung;
Abtastung_AchseB:Abtastung;          (*Zwei Instanzen des selbstdefinierten FB vom Typ Abtastung*)

Simulation_Impulsgeber:BLINK;        Instanz des Standard-FB vomTyp BLINK

Start_AchseA: BOOL;
Lichttaster_AchseA: BOOL;
Impulsgeber: BOOL;                   (*Aktuelle Variablen der Applikation AchseA*)
Rechtslauf_AchseA: BOOL;
Linkslauf_AchseA: BOOL;

Start_AchseB: BOOL;
Lichttaster_AchseB: BOOL;
Impulsgeber: BOOL;                   (*Aktuelle Variablen der Applikation AchseB*)
Rechtslauf_AchseB: BOOL;
Linkslauf_AchseB: BOOL;
END_VAR
```

- **Programmcode in FUP:**



Simulation Impulsgeber

Sowohl bei Rechts- als auch bei Linkslauf liefert der Baustein ein Taktsignal mit Impuls- und Pausenzeit von je 200ms.

Erste Instanz des FB Abtastung: Anwendung auf Antrieb Achse A

Zweite Instanz des FB Abtastung: Anwendung auf Antrieb Achse B

- **Programmcode in AWL:**

```
LD      Rechtslauf_AchseA                                   (*Simulation Impulsgeber*)
OR      Linkslauf_AchseA
ST      Simulation_Impulsgeber:ENABLE
CAL     Simulation_Impulsgeber (TIMELOW:=T#200ms,TIMEHIGH:=200ms)
LD      Simulation_Impulsgeber.OUT
ST      Impulsgeber

CAL     Abtastung_AchseA(Start:=Start_AchseA,Lichttaster:=Lichttaster_AchseA;
        Impulsgeber:=Impulsgeber)
LD      Abtastung_AchseA.Rechtslauf                         (*erste Instanz des FB Abtastung*)
ST      Rechtslauf_AchseA
LD      Abtastung_AchseA.Linkslauf
ST      Linkslauf_AchseA

CAL     Abtastung_AchseB(Start:=Start_AchseB,Lichttaster:=Lichttaster_AchseB;
        Impulsgeber:=Impulsgeber)
LD      Abtastung_AchseB.Rechtslauf                         (*zweite Instanz des FB Abtastung*)
ST      Linkslauf_AchseB
LD      Abtastung_AchseB.Linkslauf
ST      Linkslauf_AchseB
```

9.4 Zum Vergleich: Einsatz von Funktionsbausteinen im System Simatic / Step7

Funktionsbausteine kann man in Step7 mit dem gesamten Befehlsvorrat und mit beliebig vielen Ein- und Ausgängen programmieren.

Für das Parametrieren von FB stehen die lokalen Variablen vom Typ IN (analog VAR_INPUT), OUT (analog VAR_OUTPUT) und IN_OUT (analog VAR_IN_OUT) zur Verfügung.

Beide genannten Eigenschaften stimmen bei Step7 mit denen von Funktionen überein. Der Unterschied zu Funktionen liegt in den Instanzdatenbausteinen der FB und den damit möglichen statischen Variablen STAT.

- **Gibt es Standard-Funktionsbausteine?**

Anders als im System CoDeSys werden Bistabile Kippstufen, Trigger, S5-Timer und S5-Zähler nicht als Standard-Funktionsbausteine behandelt, sondern stehen als Operationen direkt zur Verfügung (**Bild 9-4**).Die erforderlichen Speicher für Bistabile Kippstufen und Trigger werden vom Anwender im Merkerbereich oder im Bereich von globalen Datenbausteinen vergeben oder aber als lokale Variablen vom Typ STAT. Bei Timern und Zählern wird mit Wahl der entsprechenden Nr. wie z.B. T12 oder Z20 ein spezielles Timer- bzw.Zählerwort reserviert. Wie viel derartige Timer oder Zähler verwendet werden können, hängt von der Leistung der CPU ab.

- **Werden Funktionsbausteine instanziert?**

Die Vorgehensweise ist vergleichbar, jedoch wird anstelle einer Instanzbildung der FB mit einem Instanzdatenbaustein aufgerufen (**Bild 9-5**).
Dieser wird vom System mit dem Anlegen von Variablen selbsttätig generiert (**Bild 9-6**).Im Instanzdatenbaustein werden die Parameter IN, OUT und IN_OUT und statische Variable STAT verwaltet.

Daneben stehen im Baustein temporäre Variable zur Verfügung, die aber nicht in den Instanzdatenbaustein eingetragen werden und deshalb weder im nächsten Bausteinaufruf noch im nächsten Zyklus zur Verfügung stehen.

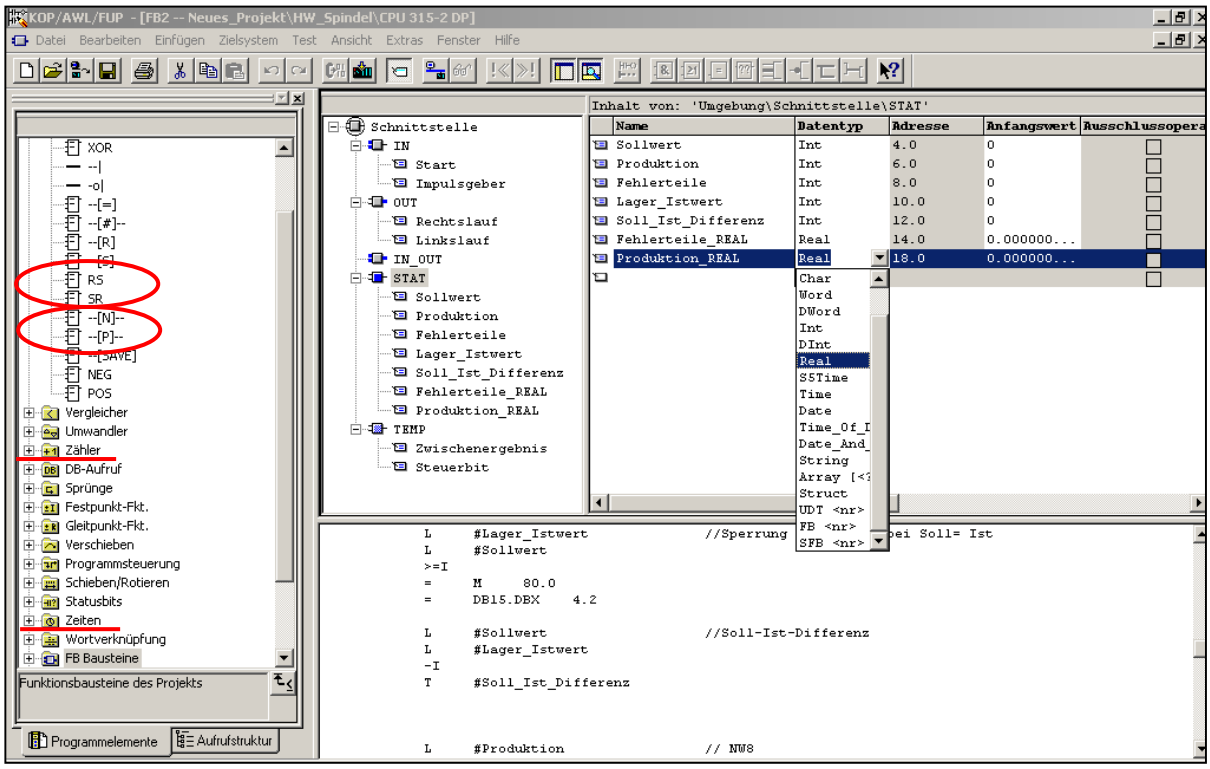


Bild 9-4: Ausschnitt der Step7-Programmieroberfläche für das Erstellen eines Funktionsbausteins

links: R_S_Speicher, Trigger, Timer und Zähler stehen als Operationen direkt zur Verfügung und müssen nicht wie Standard-Funktionsbausteine behandelt werden.

oben: Beispiel für die Deklaration von Parametern IN und OUT sowie temporären und statischen Variablen. Gezeigt ist weiter auch die Auswahlliste der Datentypen für die statische Variable mit Namen Produktion_REAL.

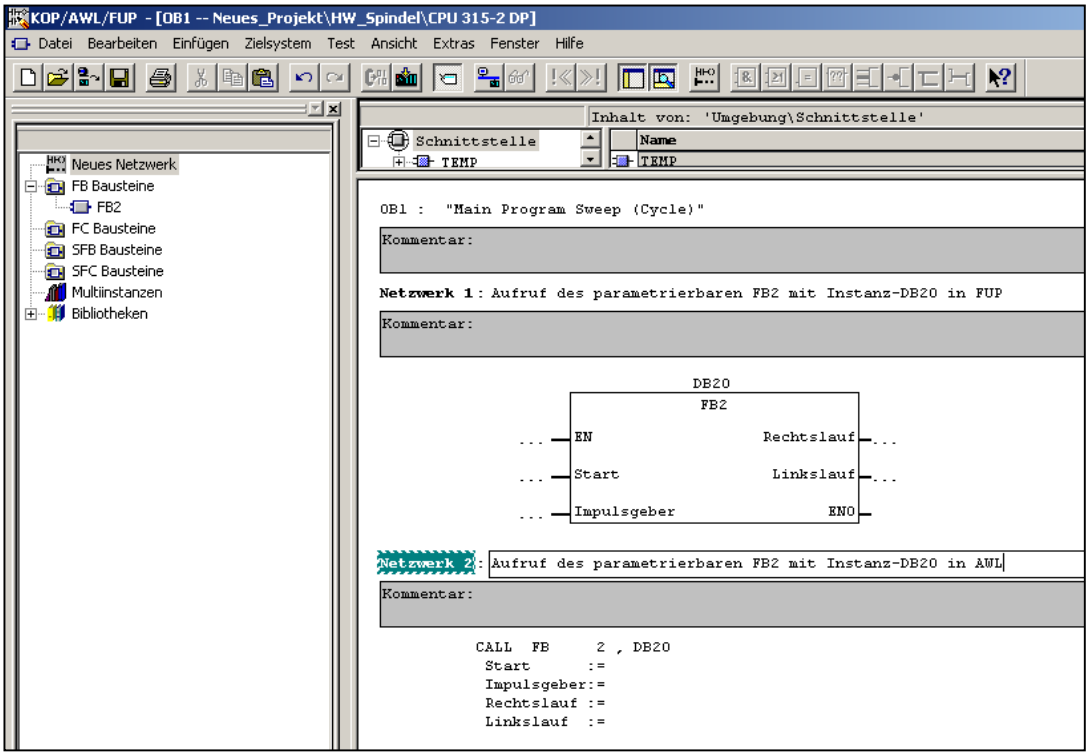
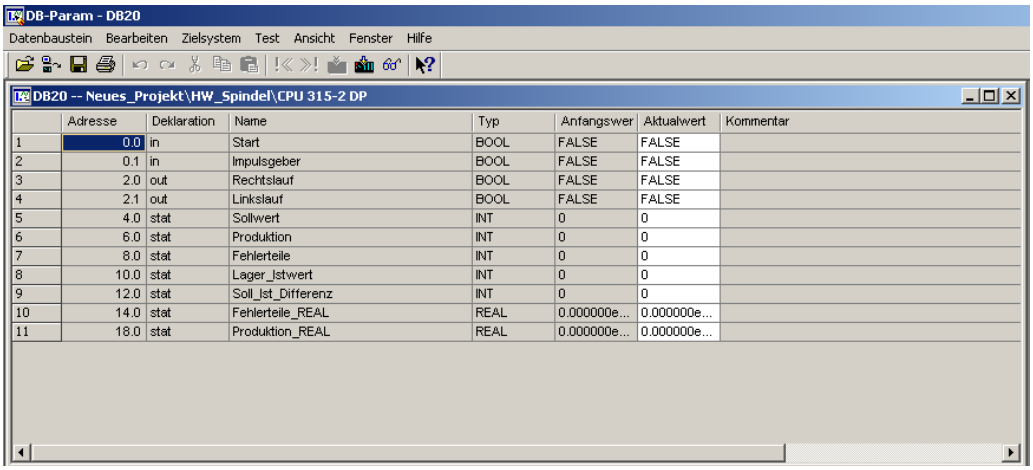


Bild 9-5: Aufruf des o.a. parametrierbaren Funktionsbausteins FB2 im OB1 mit Instanzdatenbaustein DB20 im OB1, oben in FUP, unten in AWL.



	Adresse	Deklaration	Name	Typ	Anfangswer	Aktualwert	Kommentar
1	0.0	in	Start	BOOL	FALSE	FALSE	
2	0.1	in	Impulsgeber	BOOL	FALSE	FALSE	
3	2.0	out	Rechtslauf	BOOL	FALSE	FALSE	
4	2.1	out	Linkslauf	BOOL	FALSE	FALSE	
5	4.0	stat	Sollwert	INT	0	0	
6	6.0	stat	Produktion	INT	0	0	
7	8.0	stat	Fehlerteile	INT	0	0	
8	10.0	stat	Lager_Istwert	INT	0	0	
9	12.0	stat	Soll_Ist_Differenz	INT	0	0	
10	14.0	stat	Fehlerteile_REAL	REAL	0.000000e...	0.000000e...	
11	18.0	stat	Produktion_REAL	REAL	0.000000e...	0.000000e...	

Bild 9-6: Instanzdatenbaustein DB20 des o.a. Funktionsbausteins FB2 in der Datensicht

Zu erkennen sind die deklarierten Parameter Typ IN und Typ OUT sowie die statischen Variablen. Die temporären Variablen TEMP werden nicht im Instanzdatenbaustein verwaltet!