Automatisierungstechnik nach internationaler Norm programmieren (10)

Autor: Dr. Ulrich Becker
Fachzentrum Automatisierungstechnik und vernetzte Systeme im BTZ Rohr-Kloster
Mail: Ulrich.Becker@BTZ-Rohr.de

Funktionsbausteine und ihre Parametrierung

In Folge 9 wurde die IEC-gerechte Programmierung von Funktionen und deren Parametrierung vorgestellt. Es erfolgte der Einstieg in die Programmierung mit Anweisungsliste.
Hier nun soll der Einsatz von parametrierbaren Funktionsbausteinen näher untersucht werden. Konsequent werden die Fertigkeiten der Programmierung mit Anweisungsliste erweitert.

Funktionsbausteine leisten mehr als Funktionen

Gegenüber Step7 unterscheidet IEC 61131-3 die Programmorganisationseinheiten Funktion (FC) und Funktionsbaustein (FB) in einem strengeren Sinne. Während die Funktion nur eine einzige Ausgangsvariable liefert und keine inneren Werte zur Verfügung stellt, kann man mit FB beliebig viele Ausgangs- und innere Variablen definieren. Diese inneren Variablen erhalten über die Aufrufe des FB hinaus ihren Wert. Landläufig spricht man hier von „Gedächtnis“. In Step7 wird dies durch Deklaration von statischen Variablen (Typ STAT) und deren Ablage im direkt zugeordneten Instanzdatenbaustein gewährleistet. Instanzdatenbausteine und statische Variablen kennt die Norm IEC 61131-3 in dieser Form nicht, obgleich die Zusammenhänge beim Bilden einer Instanz vergleichbar sind.

Durch ihre komplexen Möglichkeiten sind Funktionsbausteine die eigentlichen Träger der Programme nach IEC 61131-3. Folgerichtig haben wir diese für unsere bisherigen Programme konsequent eingesetzt. Wiederholt haben wir dabei gesehen, dass man FB instanzieren muss. Das erfolgt beim Aufruf des FB durch Vergabe des Instanznamens.

Dies soll nochmals an einem Beispiel verdeutlicht werden (**Bild 53**). Wenn wir einen FB mit dem Namen „*Beispiel*“ sowohl in seinem Deklarationsteil als auch im Anweisungsteil deklariert, d.h. nach Festlegung aller seiner Variablen programmiert und durch Abspeichern fertiggestellt haben, so ist seine Funktion durch Aufruf sicherzustellen. Der Aufruf erfolgte bisher stets in der POE PLC_PRG, ist aber auch in jeder anderen bereits aufgerufenen POE möglich.

In der Sprache FUP fügen wir deshalb den definierten Funktionsblock „*Beispiel*“ in die POE PLC_PRG ein und instanzieren den ersten Aufruf z.B. mit „*Einsatzfall_1*. Im Deklarationsteil des PLC_PRG erscheint nun dieser Instanzname als VAR des Typs Beispiel. Im Hintergrund wird damit die gesamte Datenstruktur des FB „*Beispiel*“ bereitgestellt.

Im Bild 53 ist weiter gezeigt, dass wir den FB „*Beispiel*“ ein zweites Mal und auch beliebig oft aufrufen können. Jedesmal wird durch Deklaration des dann zu vergebenden Instanznamens wiederum die Datenstruktur des FB komplett im System bereitgestellt.

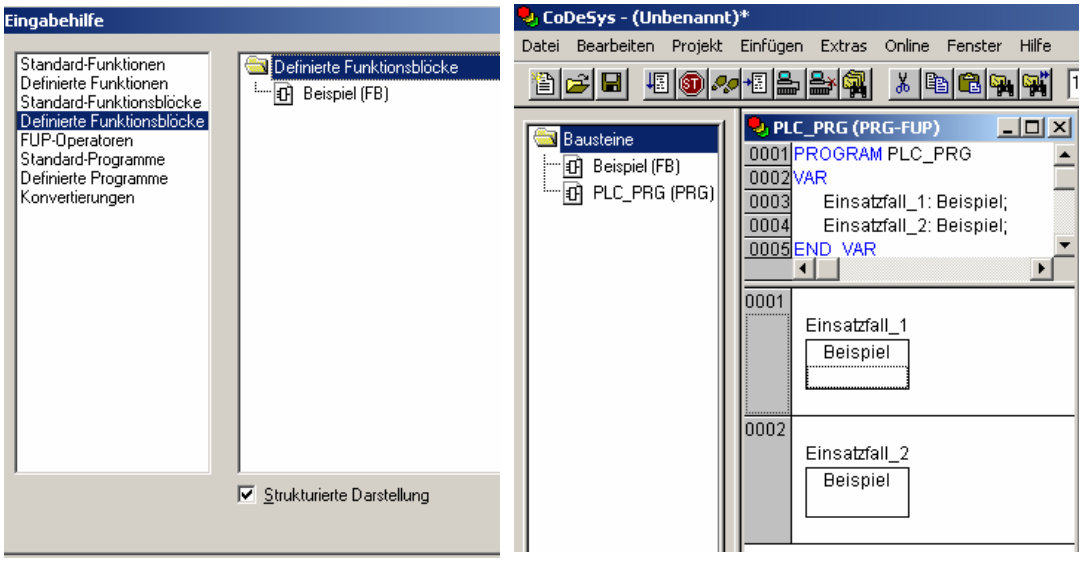


Bild 53: Grundsatz der Instanzierung von Funktionsbausteinen

Da wir schrittweise zur Programmierung in AWL übergehen wollen, soll nun der gleiche Vorgang in AWL programmiert werden. Der Funktionsbaustein „*Beispiel*“ sei fertiggestellt. Im PLC_PRG ist dann zu schreiben:

Deklarationsblock:

```
PROGRAM: PLC_PRG
VAR
Einsatzfall_1: Beispiel;        (* Erste Instanz des FB „Beispiel“ *)
Einsatzfall_2: Beispiel;        (* Zweite Instanz des FB „Beispiel“ *)
END_VAR
```

Anweisungsteil:

```
CAL Einsatzfall_1               (* Erster Aufruf des FB „Beispiel“ *)
CAL  Einsatzfall_2              (* Zweiter Aufruf des FB „Beispiel“ *)
```

Parametrierbare Funktionsbausteine

Das Wissen über parametrierbare Funktionen aus Folge 9 können wir vollständig auf Funktionsbausteine übertragen. Allerdings erkennt man die Komplexität von FB’s auch an der größeren Zahl zulässiger Bausteinparametern. So kennen FB’s neben dem lokalen (internen) Variablentyp VAR die Parameter VAR_INPUT, VAR_OUTPUT und VAR_IN_OUT. Diese größere Vielfalt der Bausteinparameter erlaubt vielfältige Gestaltungsmöglichkeiten von Programmen.

Mehr noch als bei Funktionen ist deshalb der Einsatz parametrierbarer FB für gleichartige, mehrfach anzuwendende Programmteile interessant. Der mehrfache Aufruf des FB wie in Bild 53 gezeigt wird sinnvoll, wenn bei jedem Aufruf unterschiedliche aktuelle Ein- und Ausgangsparameter übergeben werden. In Step7 beschreibt man dieses Vorgehen mit dem Übergeben von Aktualparametern an die Formalparameter. Die Formalparameter stellen die Schnittstelle des FB nach außen dar.

Aufgabenstellung 3: Parametrierbarer Meldebaustein

In einem Praxisbeispiel soll das Wissen über die Handhabung parametrierbarer FB gefestigt werden. Dazu kehren wir zurück zur Arbeit mit Trainingsrack und Verteilerband. Diese wurden in Folge1 Bild 1 (de 17/2005) vorgestellt. Das bisher erarbeitete Programm gemäß Aufgabenstellung 2 (Folge 6) soll mit folgenden Funktionen ergänzt werden (**Bild 54**):

Wenn beim Transport von Teilen vom Platz 1 aus in das Lager diese an den Initiatoren der dazwischen liegenden Arbeitsplätze 2 und 3 vorbeilaufen, soll auf der zugehörigen Meldeleuchte ein Dauersignal für maximal 2s ausgegeben werden. Gelangt ein fehlerhaftes Teil zurück zu einem der Plätze, so wird ein Blinksignal 1Hz ausgegeben. Dieses ist mit dem Platztaster zu quittieren. Wurde das Teil dann noch nicht entnommen, geht das Signal in Dauerlicht über. Bei der Entnahme des Teils verlischt es.

Ein solcher Baustein soll zuerst für Platz 3 angewendet werden. Dieser ist mit Busklemmen für Initiator INI3, Meldeleuchte H3 und Quittiertaster S3 bereits am Rack angeschlossen (hierzu Tabelle 1 Folge1). Aber auch für Platz 2 sollen die Funktionen zur Verfügung gestellt werden, da eine Nachrüstung geplant ist. Im Simulationsmodus könnten die Funktionen am Platz 2 bereits getestet werden.

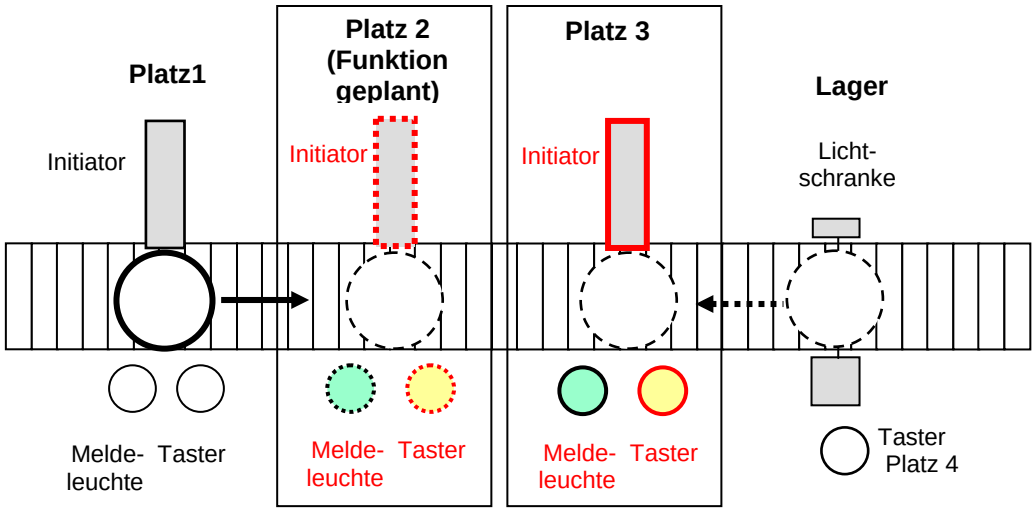


Bild 54 :Aufgabenstellung 3: Meldung mit parametrierbarem FB

Zur Lösung der Aufgabe schreiben wir zuerst einen parametrierbaren Meldebaustein „*Meldung*“ und rufen diesen dann für beide Plätze auf. Wir bemühen uns um Programmierung in AWL. Als Hilfe dient die Skizze eines Logikplans. Bei solch einem Entwurf sind auch nichtgenormte Darstellungen für den Steuerungstechniker erlaubt und hilfreich! Eine mögliche Lösung der Aufgabe zeigt **Bild 55**.

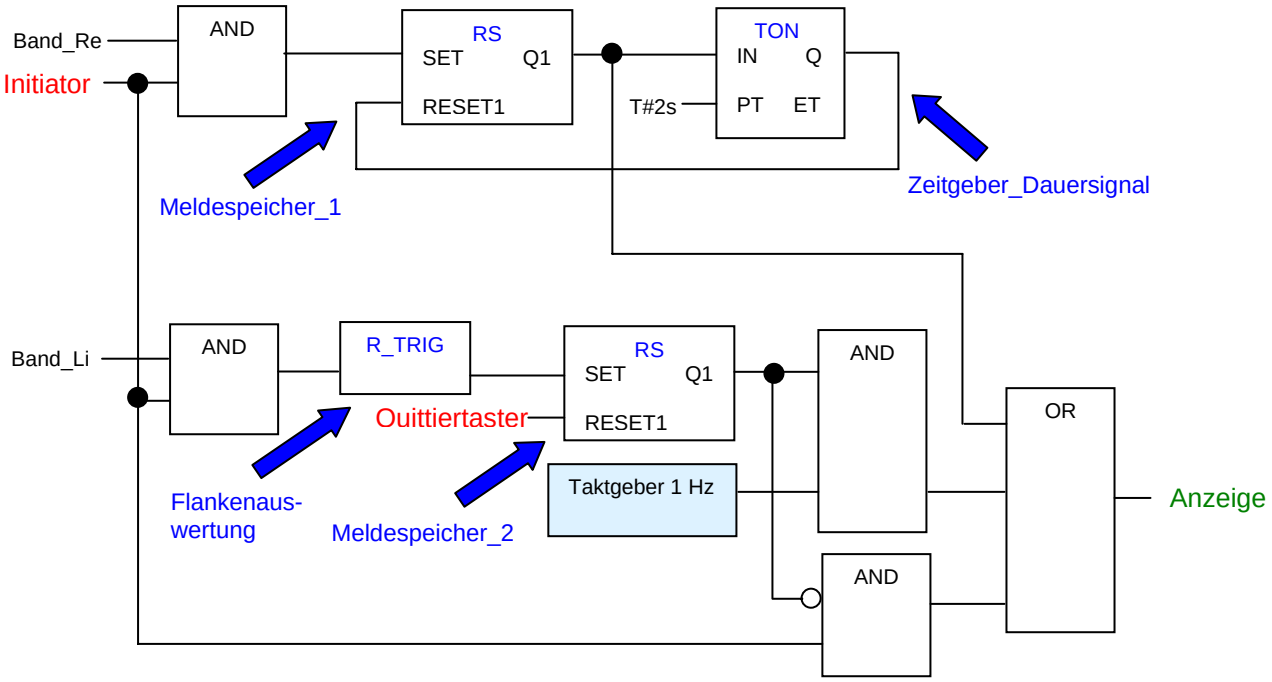


Bild 55: Entwurf eines Logikplans zur Lösung von Aufgabenstellung 3

Ein spezielles Problem stellt die Bereitstellung des geforderten Taktsignals 1 Hz dar. Selbstverständlich kann ein Takt durch Verschaltung zweier Timer erzeugt werden, jedoch stellen moderne Programmiersysteme hierzu fertige Programme in Form von Bibliotheken bereit. Simatic S7 löst das Problem hardwareseitig durch Bereitstellung eines Bytes Taktmerker unterschiedlicher Frequenz in den CPU.
Vor eigenen Programmierschritten lohnt stets die Information über verfügbare Bibliotheken. Bibliotheken können auch von den Homepage der Systemhäuser per Download ergänzt werden. **Bild 56** zeigt, wie die Bibliothek „*Util.lib*“ aus dem Order „*Library*“ des CoDeSys Programmiersystems in das Projekt eingefügt wurde. Dies erfolgt über die Menubefehle *>Fenster >Bibliotheksverwaltung* und *>Einfügen >Weitere Bibliothek*.

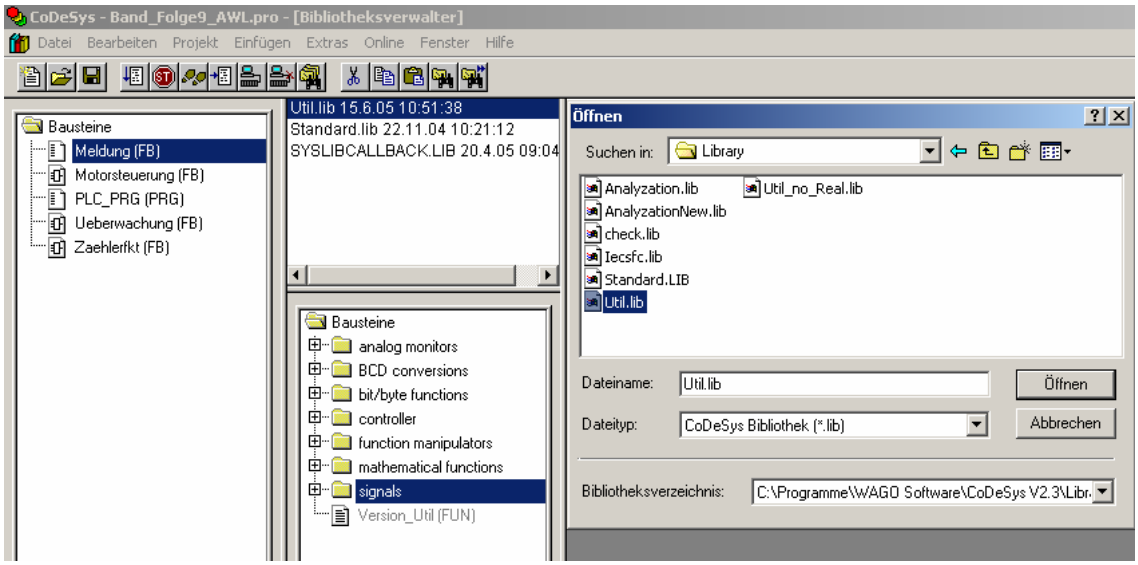


Bild 56: Einfügen der Bibliothek „*Util.lib*“ in das Projekt

Einfügen der Bibliothek „*Util.lib*“ bewirkt eine Ergänzung der bereits verfügbaren Standard Funktionsblöcke wie Trigger, Timer und Zähler. Im Ordner „*signals*“ ist nun der Baustein „*BLINK*“ verfügbar und kann wie jeder andere Standard Funktionsblock in das Programm eingefügt werden (**Bild 57**).

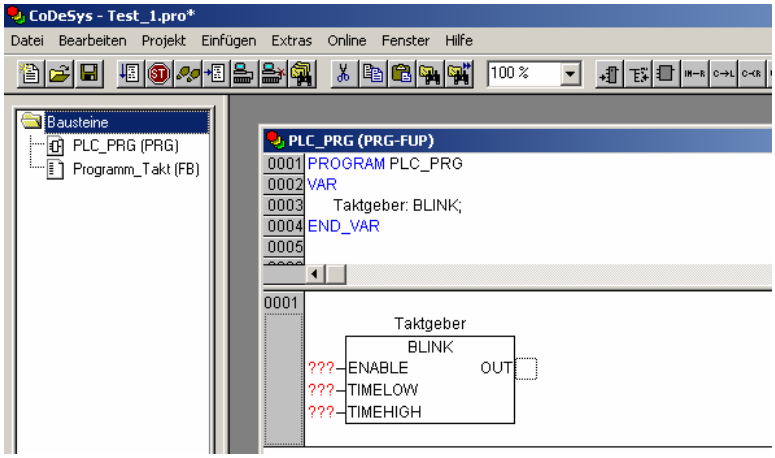
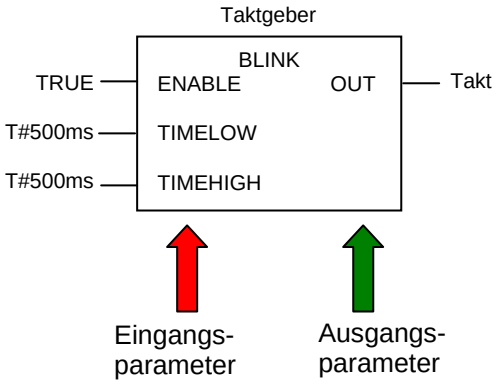


Bild 57: Einfügen und Instanzieren des Standard Funktionsblocks „*BLINK*“

Um diesen Taktgeber in das Programm zu integrieren, sind die gleichen Schritte erforderlich wie beispielsweise beim Einfügen eines SR-Flip-Flops: Instanzieren und Parameter übergeben. Nachfolgend wollen wir den bekannten Schritten in FUP die Programmierung in AWL gegenüberstellen (**Bild 58**).

FUP:

VAR
Taktgeber:BLINK;
Takt:BOOL;
END_VAR



AWL:

```
VAR
Taktgeber:BLINK;
Takt:BOOL;
END_VAR
```

```
CAL     Taktgeber (ENABLE:=TRUE, TIMELOW:=T#500ms,
        TIMEHIGH:=T#500ms)

LD      Taktgeber.OUT
ST      Takt
```

Variante:

```
LD      T#500ms
ST      Taktgeber.TIMELOW

CAL     Taktgeber (ENABLE:=TRUE,TIMEHIGH:=T#500ms)
```

Bild 58: Instanzierung und Parametrierung des FB „*BLINK*“ in FUP und AWL

Wir erkennen: Nach Aufruf des Instanznamen (hier „*Taktgeber*“*)* mit dem Befehl CAL sind in Klammer die aktuellen Werte an die Eingangparameter zu übergeben. In FUP stehen diese links im Block des FB. Die Wertzuweisung erfolgt mit dem Zeichen „ := “. Der Ausgangsparameter „*Taktgeber.OUT*“ (im Block rechtsstehend) kann an beliebiger Stelle des Programms

verarbeitet werden. Es ist auch möglich, an beliebiger Stelle vor dem Aufruf des FB einzelne Eingangsparameterwerte oder auch alle zu übergeben (siehe Variante Bild 58). Diese Parameter sind dann nicht weiter in der Klammer des CAL-Befehls aufzuführen.

Mit diesem Beispiel ist nachfolgende Definition des Funktionsbausteins nach IEC 61131 verständlich:

> Ein Funktionsbaustein (FB) ist eine in sich abgeschlossene Programmorganisationseinheit, der als Ergebnis der Ausführung ein oder mehrere Datenelemente liefert. FB besitzen auch interne Variable. Abhängig von deren Wert können Ausgangsvariable deshalb trotz gleicher Eingangsvariablen unterschiedliche Werte annehmen.
>
> Funktionsbausteine kennen Parameter vom Typ VAR_INPUT, VAR_OUTPUT und VAR_IN_OUT. Letztere sind interne Werte, die gelesen und geschrieben werden.
>
> Zwischen den Aufrufen des Bausteins können Daten gespeichert werden, diese Tatsache wird landläufig als „Gedächtnis“ des FB beschrieben.

Nach diesen Vorüberlegungen verfügen wir über die Kenntnisse zu Lösung von Aufgabenstellung 3 in AWL. Zu klären ist zuerst: Welche Parameter sind die spezifischen der einzelnen Arbeitsplätze und als solche zu deklarieren?
Als Eingansparameter werden wirksam das Initiator- und das Quittiersignal. Sie wurden in Bild 55 rot gekennzeichnet. Am Ausgang wird für jeden Platz ein Signal für die Meldeleuchte erzeugt (Darstellung grün). Wenn nun für das Programm bestimmte innere Speicher benötigt werden – z.B. für die Funktion von Flip-Flops, Flankenauswertungen oder Timer -, dann müssen diese für jeden Aufruf getrennt bereitgestellt werden. Das ist möglich durch Deklaration von Variablen des Typs VAR_IN_OUT. Solcher Speicherbedarf wurde in Bild 55 blau gekennzeichnet. Die schwarz dargestellten Signale für Rechts- und Linkslauf des Bandes wurden bereits in Aufgabenstellung 2 als globale Variable deklariert und sind damit dem FB bekannt.

Würde man programmtechnisch sicherstellen, dass zu jedem Zeitpunkt nur ein(!) Aufruf des FB „*Meldung*“ erfolgt, so könnten an Stelle der Parameter VAR_IN_OUT lokale Variablen VAR verwendet werden. Wenn aber beide Aufrufe gleichzeitig erfolgen, so müssen über diese Parameter getrennte Speicherbereiche bereitgestellt werden. Formal werden dazu den Variablen vom Typ VAR_IN_OUT aktuelle Werte in gleicher Weise wie den VAR_INPUT zugewiesen. In der graphischen Sprache FUP stehen auch die VAR_IN_OUT links im Block. Sie werden zusätzlich durch das Zeichen „ ▷ „ gekennzeichnet.

Im nachfolgenden **Deklarationsblock** des FB „Meldung“ wurde die farbliche Darstellung der Parameter zur besseren Übersicht beihalten.

```
FUNCTION_BLOCK Meldung          (*Beginn des Deklarationsteils*)

VAR_INPUT                       (* Gelesene Parameter *)
Initiator: BOOL;
Quittiertaster: BOOL;
END_VAR

VAR_OUTPUT                      (*Geschriebene Parameter *)
Anzeige: BOOL;
END_VAR

VAR_IN_OUT                      (* Gelesene und geschriebene Parameter *)
Meldespeicher_1:RS;
Meldespeicher_2: RS;
Zeitgeber_Dauersignal: TON;
Up_Trig4: R_TRIG;
END_VAR

VAR                             (* Lokale Variable *)
Taktgeber: BLINK;
Dauersignal: BOOL;
END_VAR                         (*Ende des Deklarationsteils*)
```

Der **Anweisungsteil** lautet mit Blick auf den Logikplan Bild 55:

```
CAL   Taktgeber(ENABLE:=TRUE,TIMEHIGH:=T#500ms,TIMELOW:=T#500ms)

LD    Band_Re
AND   Initiator
ST    Meldespeicher_1.SET
CAL   Meldespeicher_1(RESET1:=Zeitgeber_Dauersignal.Q)

LD    Meldespeicher_1.Q1
ST    Dauersignal
CAL   Zeitgeber_Dauersignal(IN:= Meldespeicher_1.Q1, PT:=T#2s)

LD    Band_Li
AND   Initiator
ST    Up_Trig4.CLK
CAL   Up_Trig4

LD    Up_Trig4.Q
ST    Meldespeicher_2.SET
CAL   Meldespeicher_2(RESET1:=Quittiertaster)

LD    Meldespeicher_2.Q1
AND   Taktgeber.OUT
OR    ( Initiator
ANDN  Meldespeicher_2.Q1
)
OR    Dauersignal
ST    Anzeige
```

Auffallend ist, dass für die Lösung keine über die Tabelle 10 (Folge 9) hinausgehenden Befehle erforderlich werden. Programmierung in AWL erfordert nur eine geringe, leicht zu lernende Zahl von Operatoren und Regeln.

Nun sind noch die Aufrufe des FB „Meldung“ in der POE PLC_PRG zu programmieren. Aufgeführt sind nur die Programmzeilen, welche für die Funktion der Meldungen erforderlich sind, nicht aber die gesamte Bandsteuerung. Im Anweisungsteil genutzte, aber im Deklarationsblock nicht aufgeführte Variable wie beispielsweise INI2 und INI3 wurden bereits global deklariert. Ebenfalls dort wurden die Variablen des Platzes 3 – für den Busklemmen am Rack verfügbar sind – auf Adressen gelegt. (Beispiel:INI3 AT % IX2.4:Bool;) Für Platz 2 könnte diese direkte Adressierung Busklemmen jederzeit nach dem Einbau weiterer Busklemmen nachgeholt werden.

Deklarationsblock zweier Aufrufe des FB „*Meldung*“:
(Die beiden Instanzen wurden hervorgehoben.)

```
PROGRAM: PLC_PRG
VAR
Meldung_Platz2: Meldung;        (* Erste Instanz des FB „Meldung“ *)
Meldung_Platz3; Meldung;        (* Zweite Instanz des FB „Meldung“ *)
Speicher2_1: RS;                (* betr. Meldespeicher_1 Platz 2 *)
Speicher2_2: RS;                (* betr. Meldespeicher_2 Platz 2 *)
Speicher3_1:RS;                 (* betr. Meldespeicher_1 Platz 3 *)
Speicher3_2:RS;                 (* betr. Meldespeicher_2 Platz 3 *)
Timer2:TON;                     (* betr. Zeitgeber Dauersignal Platz 2 *)
Timer3: TON;                    (* betr. Zeitgeber Dauersignal Platz 3 *)
Flanke2: R_TRIG;                (* betr. Flankenauswertung Platz 2 *)
Flanke3: R_TRIG;                (* betr. Flankenauswertung Platz 2 *)
```

Anweisungsteil zweier Aufrufe des FB „*Meldung*“:

```
CAL   Meldung_Platz2(Initiator := INI2, Quittiertaster := S2, Meldespeicher_1 :=
Speicher2_1, Meldespeicher_2 := Speicher2_2, Zeitgeber_Dauersignal := Timer2,
Up_Trig4 := Flanke2)

LD    Meldung_Platz2.Anzeige
ST    H2

CAL   Meldung_Platz3(Initiator := INI3, Quittiertaster := S3, Meldespeicher_1 :=
Speicher3_1, Meldespeicher_2 := Speicher3_2, Zeitgeber_Dauersignal := Timer3,
Up_Trig4 := Flanke3)

LD    Meldung_Platz3.Anzeige
ST    H3
```

Fazit:

In dieser Folge wurde der IEC-gerechte Einsatz von Funktionsbausteinen und deren Parametrierung vorgestellt. Funktionsbausteine sind komplexe POE und Hauptträger der Programme. Ihre Bausteinparameter und inneren Variablen bestimmen eine Datenstruktur, die mit jeder Instanzierung wiederholt zur Verfügung gestellt wird. Übungen zum Übergang

von grafischer Programmiersprache zur Anweisungsliste wurden fortgesetzt. In der nächsten Folge wird die Verarbeitung von Analogwerten vorgestellt.

Glossar: (Ergänzung zu Glossar Folge 3)

Deklarationsblock:	Bevor in einer POE Anweisungen mit Variablen geschrieben werden können, müssen diese im Deklarationsblock vereinbart (deklariert) werden. Eine POE weist immer Deklarations- und Anweisungsteil auf.
Instanz:	Nach IEC 61131 ist eine Instanz der strukturierte Datensatz eines FB's. Sie wird erzeugt durch Deklaration des FB's mit einem Bezeichner unter Angabe des Typs des FB.
Instanzierung:	Mit der Instanz wird eine Kopie der Datenstruktur und der Verarbeitungsvorschrift erzeugt, so dass diese mit anderen aktuellen Daten erneut verwendet werden können.