8. Einsatz von Funktionen

-> hierzu Folge 9 der Reihe Praktische Einführung in CoDeSys

8.1 Übersicht und Regeln

Das Wirken von Funktionen kann am besten mit der Wirkung von **„Durchgangslogik“** oder „**Sofortlogik**“ beschrieben werden, d.h. Funktionen werden für die direkte logische Verarbeitung von Variablen eingesetzt. Innere Werte können nicht gespeichert werden, und mehrere Ausgangswerte sind nicht möglich (wohl aber mehrelementige Variable wie z.B. ARRAY’s) ! Dadurch unterscheiden sich der Gebrauch von Funktionen nach IEC 61131 erheblich von dem im System Simatic S7 / Step7.

Bei definierten Eingangsvariablen liegt das einzige Ausgangssignal der Funktion als **Rückgabewert (RET_VAL)** eindeutig fest. Der Rückgabewert hat den gleichen Namen wie die Funktion selbst. Das bedeutet, dass der Funktionsname wie eine Ausgangsvariable benutzt wird.

Die IEC Operatoren wie zum Beispiel ADD, ROL, MAX etc. sind vom Aufbau her Funktionen, ohne dass man sich bei der praktischen Arbeit dessen bewusst ist und diese als Funktionen deklariert. IEC-Operatoren sind im ganzen Projekt bekannt. Im Gegensatz dazu müssen **Standardfunktionen** und andre - von Programmiersystemen zusätzlich bereitgestellte - Funktionen wie z. B. Adress-operatoren oder Arithmetische Operatoren über Bibliotheken bereitgestellt werden.

Schließlich kann man selbst **Funktionen deklarieren**. Erfolgt dies als parametrierbare Funktion, so kann man diese mehrfach in Projekten verwenden. Gemäss den o.a. Eigenschaften der Funktion stehen bei der Parametrierung nur Variablen vom Typ VAR_INPUT, nicht aber solche des Typs VAR_IN_OUT oder VAR_OUTPUT zur Verfügung!

Zusammenfassung:

Eine Funktion ist ein Baustein, der als Ergebnis der Ausführung genau ein Datum (den Rückgabewert) liefert. Innere Werte können nicht gespeichert und ausgegeben werden. Salopp kann man das Wirken einer Funktion als Durchgangslogik beschreiben.

Bei der Deklaration wird der Datentyp der Funktion festgelegt. Der Rückgabewert ersetzt den Funktionsaufruf und wird wie eine Ausgabevariable benutzt.

Parametrierbare Funktionen kennen nur Parameter vom Typ VAR_INPUT. Die Typen VAR_OUTPUT und VAR_IN_OUT sind nicht erlaubt! Ein Aufruf mit dem Befehl CAL ist nicht möglich. Dadurch unterscheiden sie sich wesentlich von Funktionsbausteinen.

Häufig werden Funktionen ohne spezielle Deklaration als Operatoren eingesetzt. IEC-Programmiersysteme schreiben eine Vielzahl von Operatoren vor und erlauben zusätzliche Standard- und selbstdefinierte Operatoren. Diese werden von Bibliotheken bereitgestellt.

8.2 Einsatz von Funktionen nach PLCopen

Zur Verdeutlichung der Besonderheiten beim Einsatz von Funktionen soll nachfolgende Aufgabe beispielhaft auf unterschiedliche Weise gelöst werden:

- Zwei Messwerte vom Typ REAL sollen verglichen werden. Wenn der erste Wert das Doppelte des zweiten Messwertes überschreitet, soll ein Signal mit Namen Grenzwert vom Typ BOOL ausgegeben werden.
  Das Signal soll auch den Wert TRUE haben, wenn das Verhältnis beider Messwerte den Wert 1,2 unterschreitet.

Wenn diese Aufgabe einmalig in einem Projekt zu lösen wäre, könnte sie auch in einer POE vom **Typ PROGRAM** programmiert werden wie nachfolgend im Bild aufgeführt. Es wurde der Name Ueberwachung gewählt. Die zyklische Bearbeitung erfolgt durch Aufruf im Baustein PLC_PRG. Zur Verdeutlichung der Quelle aktueller Messwerte wurden die Variablen auf Eingangsadressen gelegt. An die Eingangskanäle könnten z.B. Messwertgeber angeschaltet sein.

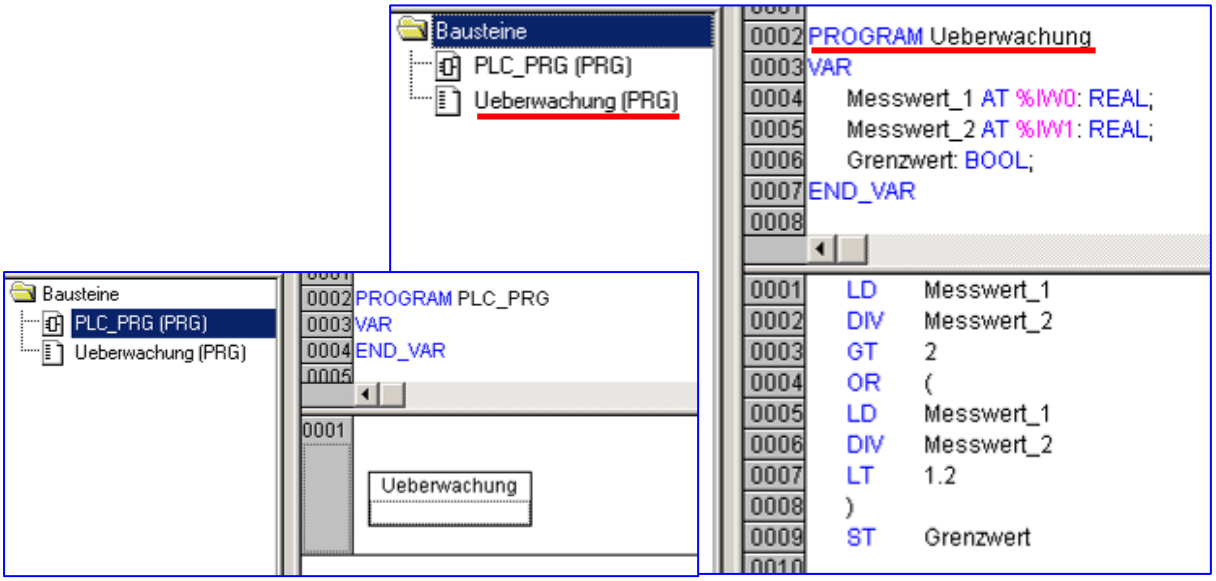


Bild 8-1: Lösung der Aufgabe in einer POE vom Typ PROGRAM

Alternativ könnte das Problem – weil es sich um „Durchgangslogik“ handelt - mit einer POE vom **Typ FUNCTION** gelöst werden. Es wurde der gleiche Namen gewählt. Das Fenster „Neuer Baustein“ zeigt, wie man den Rückgabewert der Funktion deklariert. Links unten ist in Sprache FUP zu sehen, wie die Funktion in der POE PLC_PRG aufgerufen wird. Das unterscheidet sich wenig vom Aufruf des o.a. Programms.

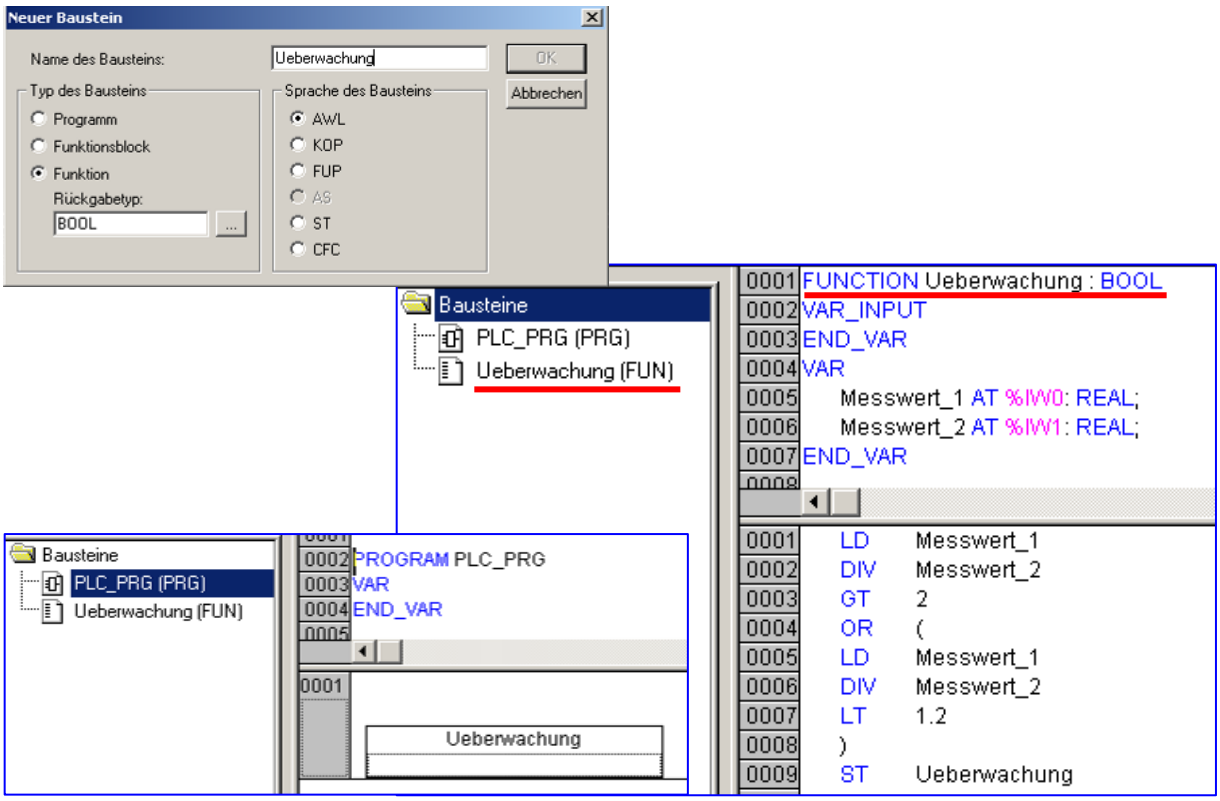


Bild 8-2: Nichttypische Lösung der Aufgabe in einer POE vom Typ FUNCTION

Nachfolgendes ONLINE-Fenster zeigt die Simulation mit zwei vorgegebenen Messwerten. Die Funktion gibt den Rückgabewert Ueberwachung fehlerfrei aus.

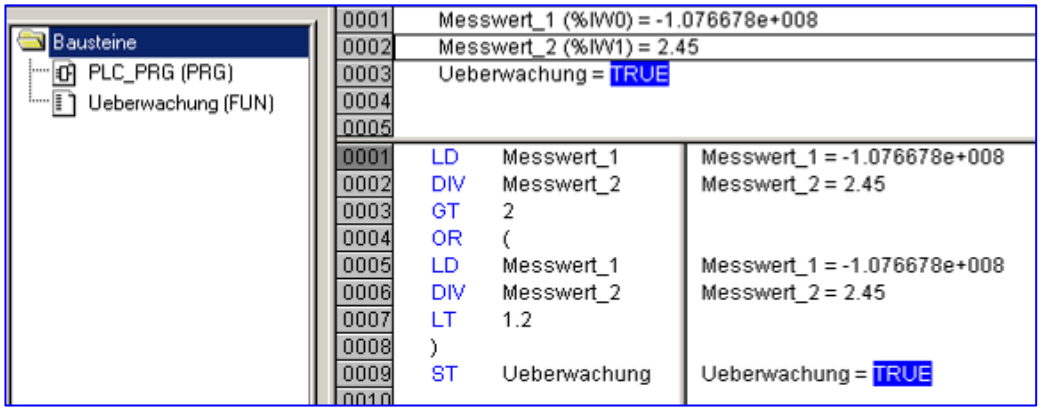


Bild 8-3: Simulation zur Demonstration der Lösung mit POE FUNCTION

Eine derartige Anwendung des Typs FUNCTION ist jedoch **nicht typisch!** Wollte man die Funktion für unterschiedliche Signale oder Messstellen anwenden, so müsste man den Programmcode jedes Mal neu schreiben! (Dies gilt genauso bei der vorherigen Anwendung des Typs PROGRAM!)

Typisch ist stattdessen der Einsatz von **parametrierbaren Funktionen**. Diese ist wie jede parametrierbare POE grundsätzlich dann von Vorteil, wenn das vorliegende Programmdetail mehrfach Verwendung für unterschiedliche Eingangsvariablen Var1 und Var2 finden soll. Eine parametrierbare Funktion wird wie ein Bibliotheksbaustein ohne Bezug auf konkrete Anwendung und konkrete Hardware geschrieben. Lediglich die Variablen müssen fehlerfrei deklariert werden. Der Bezug zu den konkreten Daten bzw. Signalen wird erst **beim jeweiligen Einsatz der Funktion** hergestellt.

Nachfolgend wird die parametrierbare Funktion mit Namen Ueberwachung angegeben. Der Programmcode in AWL ändert sich nicht. Die Variablen, mit denen die Funktion rechnet, sind vom Typ VAR_INPUT zu deklarieren! Der Rückgabewert heißt wie die Funktion selbst und ist vom Typ BOOL.

Zusätzlich wurde rechts der Programmcode in CFC angeführt.

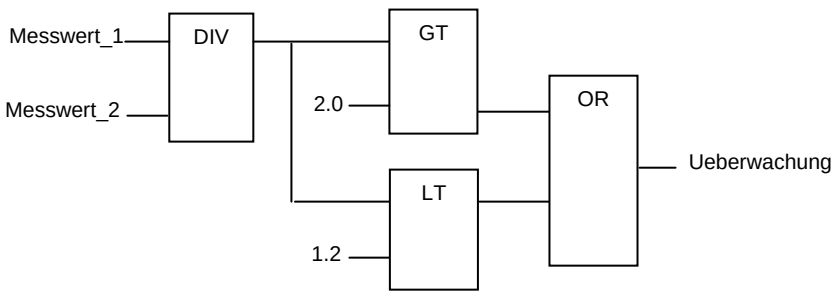
- **Deklaration:**

FUNCTION Ueberwachung:BOOL
VAR_INPUT
Messwert_1:REAL;
Messwert_2:REAL;
END_VAR

- **Programmcode in AWL:**

```
LD  Messwert_1
DIV Messwert_2
GT 2.0
OR  (
LD  Messwert_1
DIV Messwert_2
LT 1.2
)
ST  Ueberwachung
```

- **Programmcode in CFC:**



- **Programmcode in FUP unter Nutzung von „Zwischenergebnissen“:**

Im Abschnitt 8.1 wurde beschrieben, dass in Funktionen keine „Zwischenergebnisse“ gespeichert und von außen gelesen werden können. Das bedeutet nicht, dass man keine lokale Variablen von Typ VAR als Zwischenwerte benutzen darf! Will man beispielsweise den Programmcode in FUP schreiben, ist dies wegen der begrenzten Möglichkeiten der Netzwerke sogar erforderlich. Es bleibt aber bei der Wirkung der Funktion als „Durchgangslogik“ ohne innere Speicher! Solche Probleme sind bei Programmieren in Anweisungsliste selbstverständlich weniger zu erwarten (siehe Kasten)!

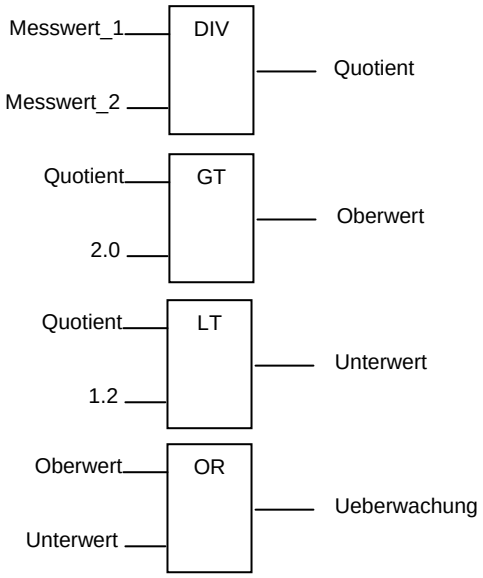
Bei der Programmierung in FUP muss der CFC-Block in einzelne Netzwerke zerlegt werden. Diese werden unter Nutzung lokaler Variablen als „Zwischenwerte“ verbunden. Oben angegebene Deklaration ist dann um den Block VAR zu ergänzen.

- **Deklaration:**

FUNCTION Ueberwachung:BOOL
VAR_INPUT
Messwert_1:REAL;
Messwert_2:REAL;
END_VAR

VAR
Quotient:REAL;
Oberwert:BOOL;
Unterwert:BOOL;
END_VAR

- **Programmcode in CFC:**



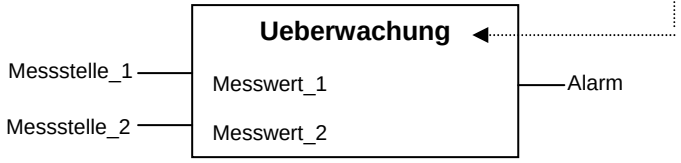
- **Einsatz der parametrierbaren Funktion**

Nun soll die FUNCTION Ueberwachung: BOOL angewendet werden. Dies kann z. B. durch Aufruf im Hauptprogramm PLC_PRG erfolgen. Dabei werden aktuelle Signale angeschaltet bzw. aktuelle Messwerte übergeben. Der Vorgang unterscheidet sich nicht von dem, wenn eine Funktion aus einer Bibliothek verwendet wird.
Dem Rückgabewert (RET_VAL) der Funktion kommt eine besondere Bedeutung zu. Er hat den gleichen Namen wie die Funktion selbst. Das ist besonders im AWL-Code zu beachten!

PLC_PRG

VAR
Messstelle_1:REAL;
Messstelle_2:REAL;
Alarm: BOOL;

END_VAR



AWL:

```
LD Messstelle_1
Ueberwachung Messstelle_2
ST Alarm
```

zum Vergleich: Operator ADD

```
LD Variable_1
ADD Variable_2
ST Ergebnis
```

Die einmal erstellte parametrierbare Funktion kann nun beliebig oft für unterschiedliche Paare von Messwerten verwendet werden. In nachfolgender Simulation wurden in diesem Sinne die Variablennamen um „Messstelle1...n“ erweitert und es wurden konkrete REAL-Werte vorgegeben. Bei einer praktischen Anwendung könnten diese über Messgeber an Eingangskartenbaugruppen bereitgestellt werden.

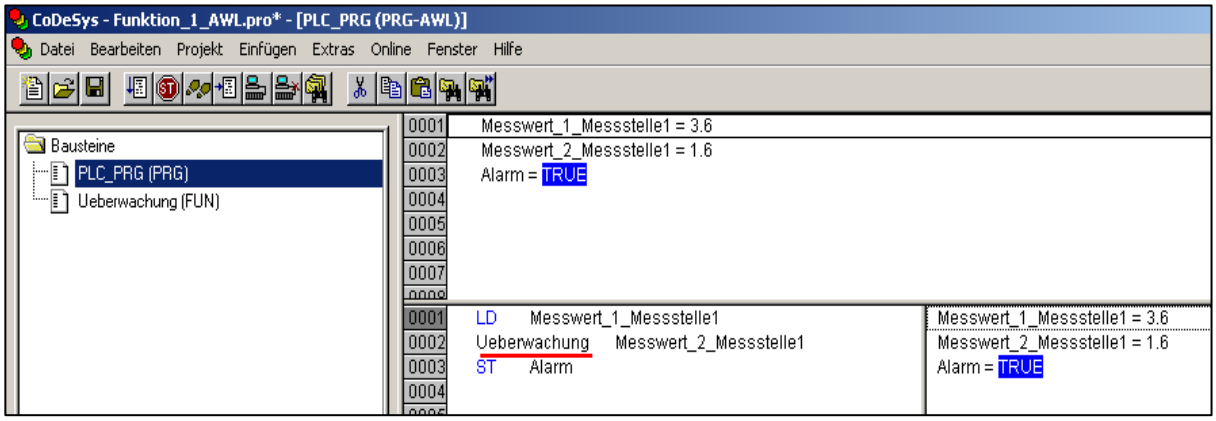


Bild 8-4: Online-Simulation der POE PLC_PRG mit Nutzung der Funktion Ueberwachung und Übergabe der Messwerte

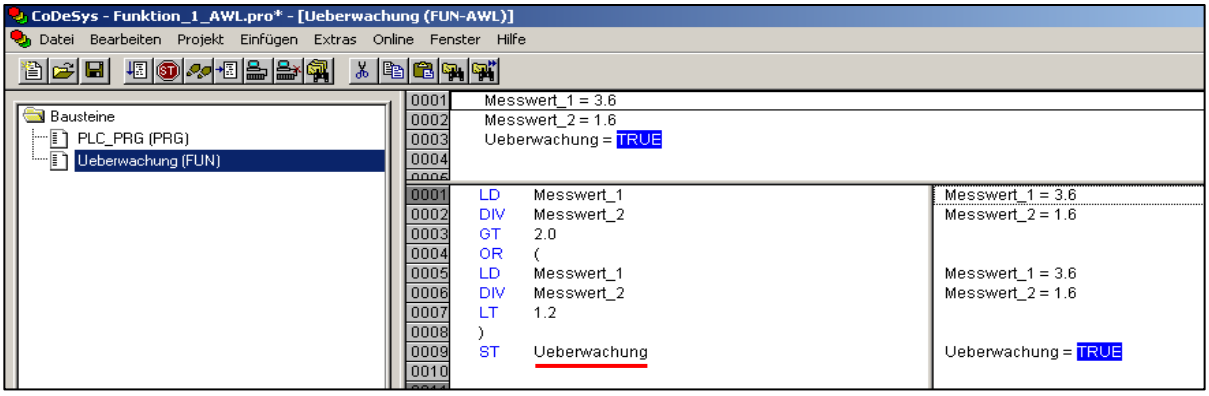


Bild 8-5: Online-Simulation der POE Ueberwachung (FUN) in AWL

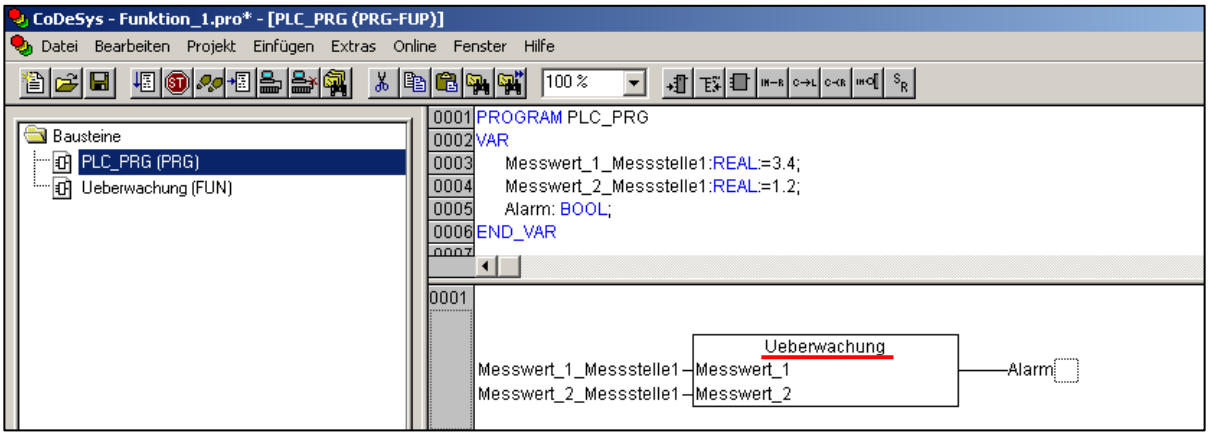


Bild 8-6: Online-Simulation der POE PLC_PRG in FUP mit Nutzung der Funktion Ueberwachung und Übergabe der Messwerte

8.3 Zum Umgang mit Funktionen in Bibliotheken

Die Standardbibliothek des Programmiersystems CoDeSys enthält neben zahlreichen Funktionsbausteinen auch Funktionen. Die Bibliotheksverwaltung **Bild 8-7** zeigt die verfügbaren Funktionen mit Datentyp STRING. Alle diese Funktionen sind parametrierbare Funktionen. Ausgewählt wurde die Funktion INSERT. Rechts sind die Parameter sowie die Anwendung der Funktion in Sprache FUP gezeigt.

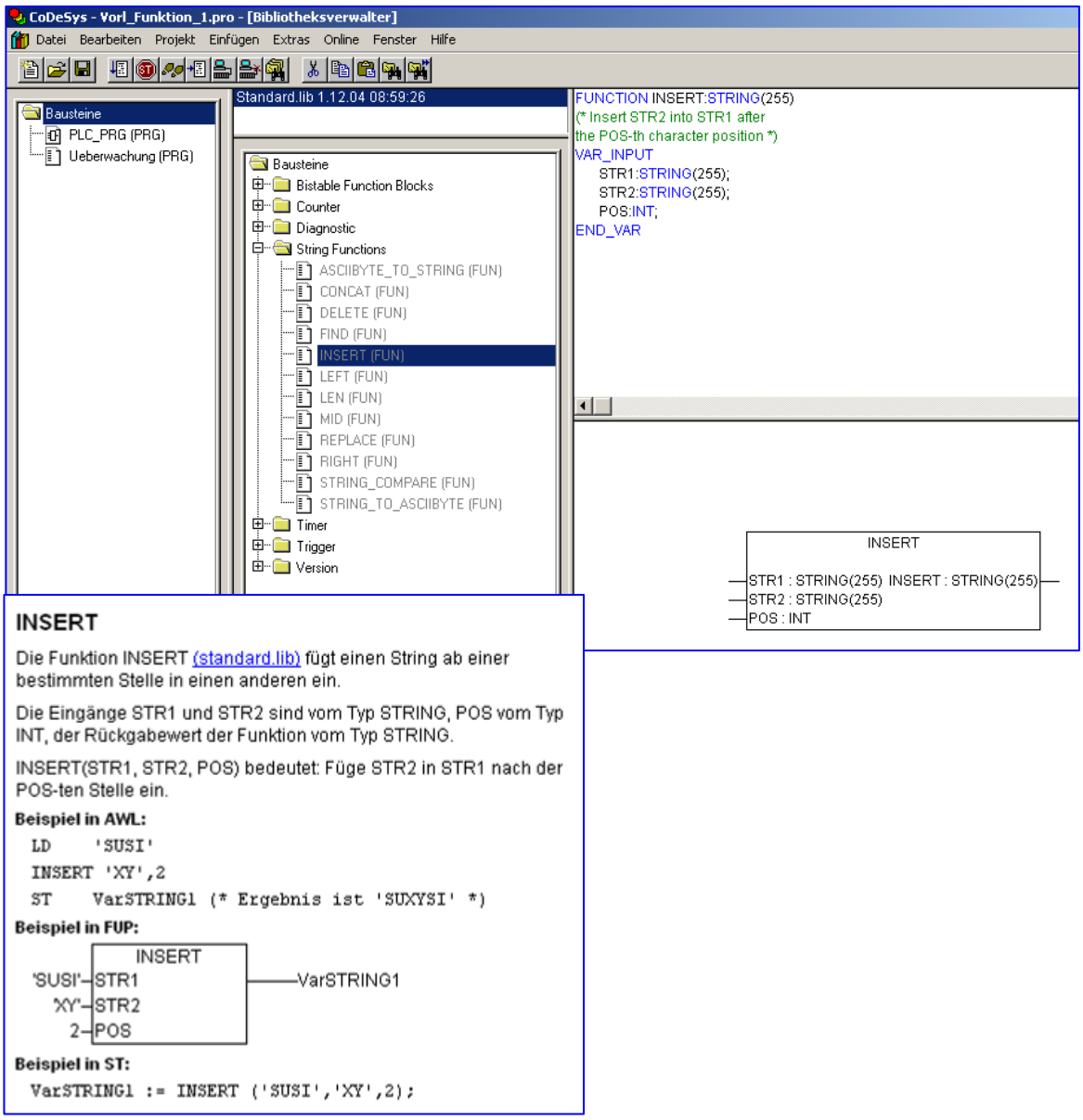


Bild 8-7: Bibliotheksverwaltung mit Anzeige der verfügbaren String Funktionen und Hilfetext. Ausgewählt wurde die Funktion INSERT

8.4 Zum Vergleich: Einsatz von Funktionen im System Simatic / Step7

Bei Funktionen bestehen **erhebliche Unterschiede** zwischen Step7 und Programmiersystemen nach PLCopen: In Step7 kann grundsätzlich der gesamte Befehlsvorrat – also z. B. auch die Befehle Setzen und Rücksetzen - sowohl in Funktionen (FC) als auch in Funktionsbausteinen (FB) gleichermaßen eingesetzt werden. Es stehen auch in Funktionen Parameter der Typen IN, OUT und IN_OUT zur Verfügung. FC's können nicht nur eine, sondern beliebig viele Ausgangsvariablen haben. Sofern man den Merkerbereich oder globalen Datenbereich nutzt, kann man in FC's beliebig viele innere Variablen mit Speicherfunktion einsetzen. FC's werden in Step7 wie FB's mit dem CAL-Befehl aufgerufen.

Um die Unterschiede zwischen Funktionen und Funktionsbausteinen in Step7 zu verstehen, muss man sich mit der **lokalen** Datenhaltung im System Step7 befassen. Der entscheidende Unterschied zwischen FC und FB besteht darin, dass in Funktionsbausteinen neben temporären (lokalen) Variablen TEMP auch statische (lokale) Variablen STAT zur Verfügung stehen. Für ihre Speicherung steht der Instanzdatenbaustein zur Verfügung.

> Temporäre Variablen dienen als Zwischenspeicher, um Informationen innerhalb eines Bausteins von „oben nach unten" weiter zu geben. Die Information der TEMP-Variablen geht nach jedem Programmzyklus verloren.
>
> Statische Variablen STAT werden im zugeordneten Instanzdatenbaustein gespeichert. Sie stehen deshalb auch im nächsten Programmzyklus und über Bausteinaufrufe hinweg zur Verfügung.

In neueren Versionen von Step7 wurde die Deklarationstabelle von Funktionen um den Rückgabewert (RET_VAL) mit Namen RETURN ergänzt.
Bild 8-8 zeigt einen Ausschnitt des Programmcodes einer Funktion FC20 in Step7. Die Code wird in FUP mit symbolischen Operanden dargestellt. Im Beispiel wurden weder Parameter noch temporäre Variablen benutzt.

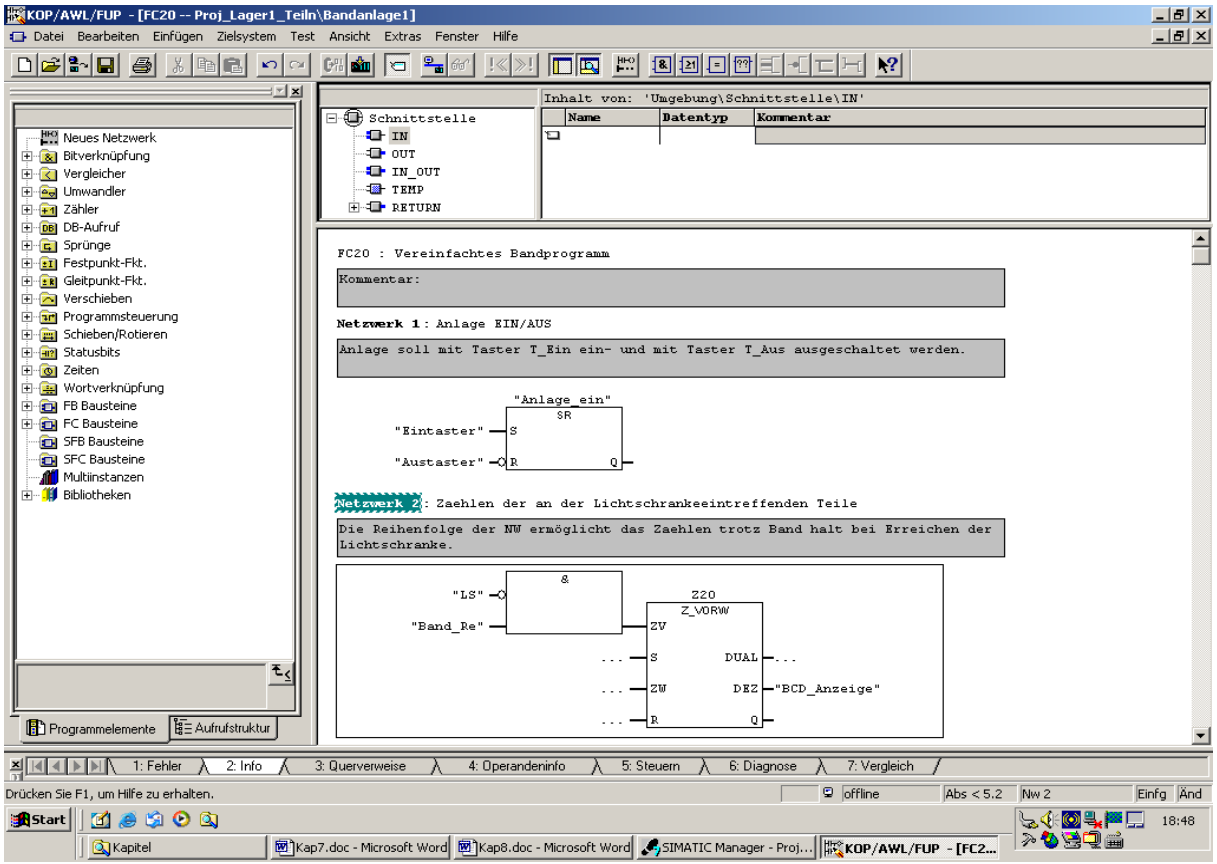


Bild 8-8: Ausschnitt des Programmcodes einer nichtparametrierbaren Funktion in Step7

Bild 8-9 zeigt den Aufruf der Funktion in FUP und AWL im Organisationsbaustein OB1 als eine mögliche Voraussetzung ihrer zyklischen Bearbeitung

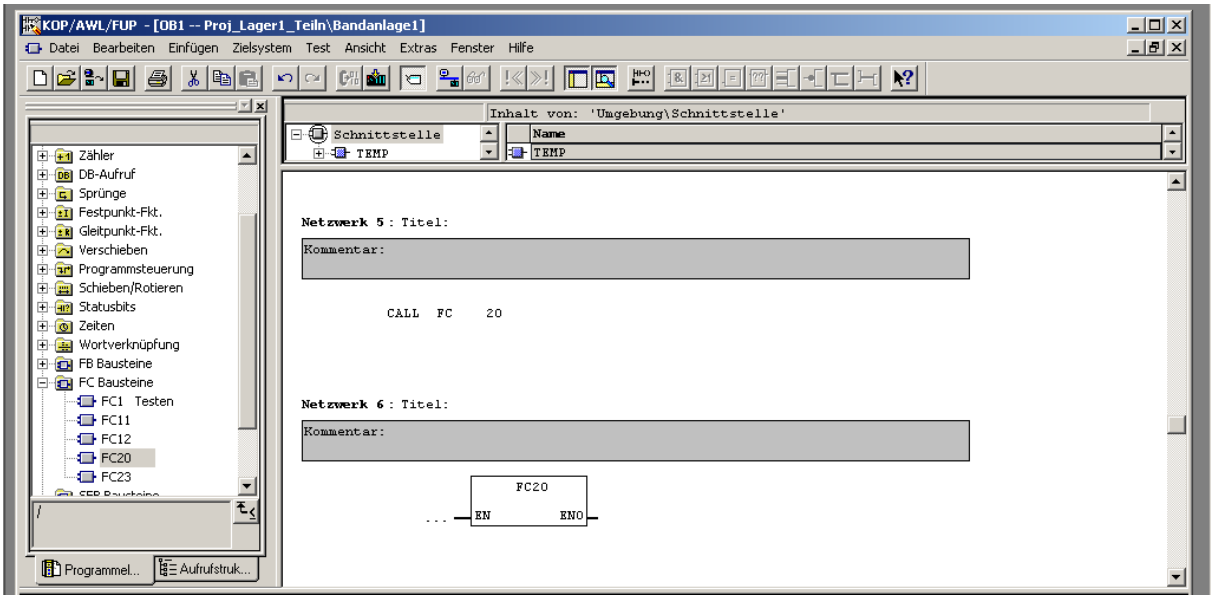


Bild 8-9: Abweichend von IEC 61131-3: Aufruf der Funktion in Step7 mit dem CALL-Befehl in AWL, darunter Aufruf in FUP

- **Welche Bedeutung haben die Parameter EN und ENO?**

Das Programmiersystem Step7 verwendet **in den Sprachen FUP und KOP** an Blöcken häufig den Eingangsparameter Freigabe ENABLE (EN) und den Ausgangsparameter ENABLE OUT (ENO).

Im Hintergrund verbergen sich bedingte Sprünge hinter diesen Parametern. Damit wird das Programmieren erleichtert, denn man kann ohne Einsatz von Sprungbefehlen entscheiden, ob ein Block wie beispielsweise eine Funktion FC oder ein Funktionsblock FB ausgeführt wird oder nicht.

Dabei gilt:
- Wird der Parameter EN nicht beschaltet, wird der Block immer ausgeführt.
- Wird eine Variable vom Typ BOOL angeschaltet, so wird
  - bei deren Wert TRUE der Block ausgeführt
  - bei deren Wert FALSE der Block nicht ausgeführt.

Wird an den Ausgangsparameter ENO eine Variable vom Typ BOOL angeschaltet, so hat diese bei fehlerfreier Ausführung den Wert TRUE.

Durch Verbindung von ENO mit EN nachfolgender Blöcke können in graphischen Sprachen Ketten programmiert werden.

.