

## Automatisierungstechnik nach internationaler Norm programmieren (9)

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

### Folge 9: Parametrierbare Funktionen und Einstieg in die AWL Programmierung

In Folge 8 wurde die Praxis der Deklaration globaler und lokaler Variablen nach IEC 61131-3 dargelegt. Nach Überführung einer Vielzahl von Variablen aus lokaler zu globaler Deklaration und Abstellung angezeigter Syntaxfehler konnte das Programm geladen und getestet werden. Logische Fehler wurden mit Hilfe spezieller Softwaretools eingegrenzt. Die gewünschten Funktionen verlaufen danach fehlerfrei. In der vorliegenden Fortsetzung soll nun der Umgang mit der POE „Funktion“ näher untersucht werden. Die Kenntnisse über Variablen und Bausteinparameter werden weiter vertieft.

#### Parametrierbare Funktionen

In der Automatisierungstechnik müssen oftmals gleichartige Programmteile mehrfach angewendet werden. Das erfolgt mittels parametrierbarer Bausteine (vergl. Folge 6 Bild 31). Während in Step7 solche Aufgaben oft gleichwertig sowohl mit parametrierbaren Funktionen als auch mit parametrierbaren Funktionsbausteinen gelöst werden können, ist der Einsatz von Funktionen nach IEC 61131-3 deutlich begrenzt. Bisher haben wir Programmteile stets in Funktionsbausteine (FB) geschrieben, was aber leisten Funktionen?

Wir wollen diese Frage mit einem Beispiel beantworten: Für die Kontrolle eines chemischen Prozesses sei an verschiedenen Orten wiederholt erforderlich, jeweils zwei Messwerte des Typs REAL (Gleitpunktzahl) nach folgender Vorschrift zu verarbeiten: Beide Zahlen sind zu multiplizieren, aus dem Produkt ist die Quadratwurzel zu ziehen und das Ergebnis ist gerundet bereitzustellen.

Man könnte nun dieses Teilprogramm beliebig oft mit aktuellen Werten zweier Variablen Var1 und Var2 schreiben. Eleganter löst man dies aber mit einer parametrierbaren Funktion und Bausteinparametern des Typs INPUT. In der Programmiersprache FUP lösen wir zunächst die arithmetische Aufgabe wie im **Bild 49** dargestellt.

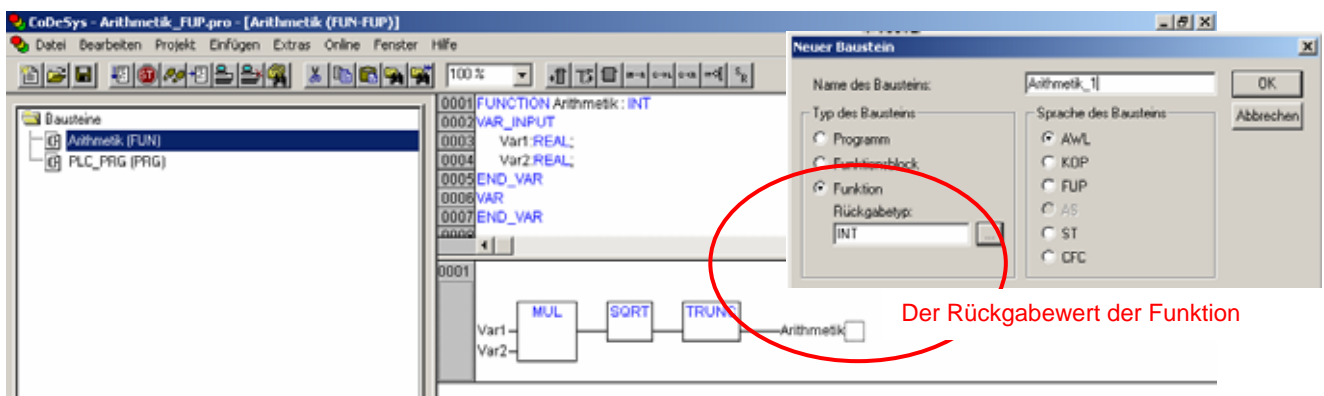


Bild 49: Programm der Funktion „Arithmetik“, rechts dargestellt das Anlegen der Funktion mit einem Rückgabewert vom Typ Integer

Es wurde eine Funktion mit dem Namen „Arithmetik“, ihrem Rückgabewert vom Typ Integer und zwei lokalen Variablen vom Typ Parameter INPUT deklariert (siehe Deklarationsteil oben). Im Anweisungsteil erfolgt die rechnerische Verknüpfung der beiden Variablen. Im Anweisungsteil unten erkennen wir die Operatoren für Multiplikation (MUL), Quadratwurzel (SRQT) und Runden (TRUNC). Das Ergebnis erscheint als Rückgabewert der Funktion und hat den gleichen Namen (Bezeichner) wie die Funktion selbst. Diesem Rückgabewert (auch als Return\_Value geführt) kommt bei der Funktion eine besondere Bedeutung bei.

**Bild 50** zeigt den zweimaligen Aufruf der Funktion in der POE PLC\_PRG. Im graphischen Baustein der Funktion erscheinen links die bereits deklarierten Parameter vom Typ INPUT. Beim Aufruf werden hier nun die aktuellen Variablen angetragen: Werte A und B an den Ablesestelle 1 bzw. 2 (siehe Deklarationsteil oben). Die VAR\_INPUT stellen quasi die Schnittstelle der Funktion zum Programm dar und bei jedem Aufruf werden zutreffende Aktualwerte übergeben. Im Beispiel wurden den Aktualparametern bestimmte Werte zugewiesen, in der Praxis würden stattdessen per Programm Messwerte eingetragen.

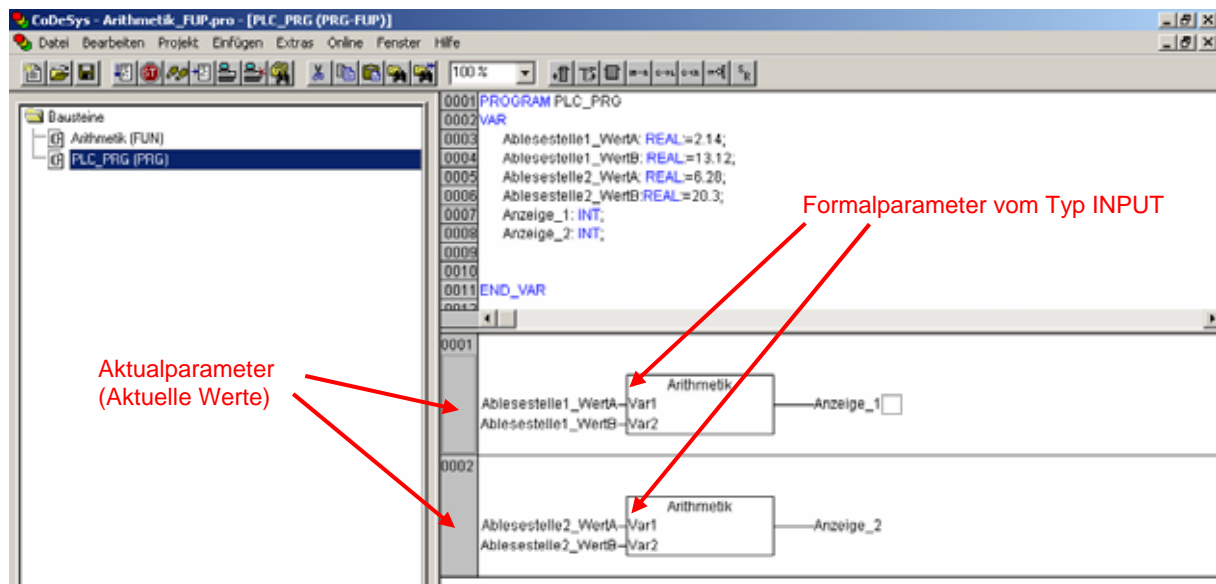


Bild 50: Programm PLC\_PRG mit zwei Aufrufen der Funktion „Arithmetik“ und Übergabe aktueller Werte

Einmal geschrieben könnte die von uns programmierte Funktion nun beliebig oft genutzt (aufgerufen) werden. Mit diesem Beispiel sind uns nachfolgende Regeln der Nutzung von Funktionen nach IEC 61131 verständlich:

Eine Funktion ist ein Baustein, der als Ergebnis der Ausführung genau ein Datum (den Rückgabewert) liefert. Innere Werte kennt die Funktion nicht.

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

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 von Funktionsbausteinen.

Um auch in Step7 IEC-konform arbeiten zu können, wurde in neueren Versionen der Deklarationsteil von Funktionen (FC) um den Rückgabewert (RET\_VAL) ergänzt (**Bild 51**).

Zu sehen ist hier, dass in Step 7 Funktionen auch Parameter vom Typ OUT und IN\_OUT haben. Wie mehrfach erwähnt, ist dies in IEC Programmiersystemen nicht der Fall!

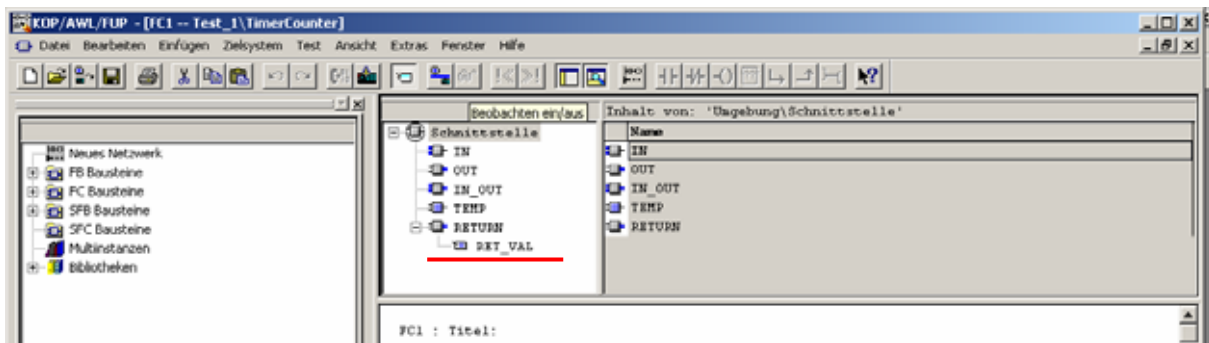


Bild 51: Zum Vergleich: Deklarationsteil von Funktionen (FC) in Step7

### Vorteile der Programmiersprache AWL

In Folge 8 wurde bei der Fehlersuche zum ersten Male auf Vorteile des Programmierens in AWL verwiesen. Für anspruchsvolle Programme wird man stets AWL wählen, auch wenn Einsteiger die bequemen graphischen Sprachen (insbesondere Funktionsbausteinsprache FUP) bevorzugen. Wir wollen uns in dieser Folge schrittweise der Arbeit mit Anweisungsliste zuwenden. **Tabelle 10** zeigt einige Vor- und Nachteile dieser Programmiersprache.

| Graphische Programmiersprachen<br>FUP und KOP  | Programmiersprache AWL  |
|--|---|
| <b>Vorteile</b>  | <b>Vorteile</b>   |
| Befehle können aus Listen ausgewählt werden, die Syntax muss nicht im einzelnen bekannt sein | Detaillierte Darstellung aller Befehle durch aufeinanderfolgende Anweisungen; die Reihenfolge der Anweisungen bestimmt der Programmierer<br>Keine Beschränkung durch Netzwerke<br>Auch komplexe Operationen – für die es keine Graphik gibt – können programmiert werden.<br>Gute Vorbereitung auf höhere Programmiersprachen wie SCL / ST<br>Spezielle Testmethoden wie Einzelschrittbetrieb sind nur in AWL möglich |
| <b>Nachteile</b>   | <b>Nachteile</b>  |
| Komplexe Operationen, für die es keine   | Der Programmierer muss die Schreibweise   |

|   |   |
|---|---|
| fertigen graphischen Operationen gibt, sind häufig nicht programmierbar | aller Befehle selbst kennen, es gibt keine Vorlagen |
|---|---|

Tabelle 10: Vor- und Nachteile der Programmierung in AWL

### Lösung der arithmetischen Aufgabenstellung in AWL

In AWL schreibt man die parametrierbare Funktion wie nachstehend aufgezeigt. Die in die Zeichen (\* ... \*) eingeschlossenen Kommentare erläutern die einzelnen Operationen. Die Schlüsselwörter sind besonders zu beachten! In **Tabelle 11** sind erste IEC Operatoren aufgeführt, diese Tabelle werden wir fortlaufend zu ergänzen haben.

#### Deklarationsteil:

```

FUNCTION Arithmetik: INT (* Definition einer Funktion mit Namen Arithmetik, der *)
                          (* Rückgabewert ist vom Typ Integer *)
VAR_INPUT
Var1:REAL;              (* Erste ("Eingangs")Variable vom Typ Real *)
Var2:REAL;              (* Zweite ("Eingangs")Variable vom Typ Real *)
END_VAR

```

#### Anweisungsteil:

```

LD Var1                  (* Lesen der ersten Variablen*)
MUL Var2                 (* Multiplizieren mit dem Wert der zweiten Variablen*)
SQRT                     (* Ziehen der Quadratwurzel *)
TRUNC                    (* Runden (Umformung Real in Integer) *)
ST Arithmetik            (* Das Ergebnis wird in den Rückgabewert geschrieben, *)
                          (*dieser trägt den gleichen Namen wie die Funktion selbst *)

```

Nach den Regeln für parametrierbare Funktionen rufen wir diese nun beliebig oft auf. Nachfolgend erfolgt dies wiederum zweimal im Programm PLC\_PRG. Dort sind noch die Variablen „Anzeige\_1“ und „Anzeige\_2“ zu deklarieren, in welche das spezielle Ergebnis jedes der Aufrufe eingetragen wird. Abweichend von der Darstellung in FUP wurden diesmal keine Aktualparameter als Variable eingeführt. Stattdessen wurde direkt mit Gleitpunktzahlen gearbeitet.

#### Deklarationsteil:

```

PROGRAM PLC_PRG
VAR
Anzeige_1: INT;
Anzeige_2: INT;
END_VAR

```

#### Anweisungsteil:

|            |           |
|------------|-----------|
| LD         | 2.14      |
| Arithmetik | 13.12     |
| ST         | Anzeige_1 |
| LD         | 6.28      |
| Arithmetik | 20.3      |
| ST         | Anzeige_2 |

Deutlich ist zu sehen, wie die Funktion „Arithmetik“ die Rolle eines komplexen Operators übernimmt. Umgekehrt erkennen wir rückschauend, dass Operatoren wie z.B. MUL, DIV oder ADD vom Grundsatz her selbst Funktionen sind.

Die Online-Sicht beider Aufrufe der Funktion bei Simulation zeigt **Bild 52**. Rechts bzw. oben sind die gerundeten Ergebnisse zu sehen, welche in die Variablen „Anzeige\_1“ und „Anzeige\_2“ geschrieben wurden.

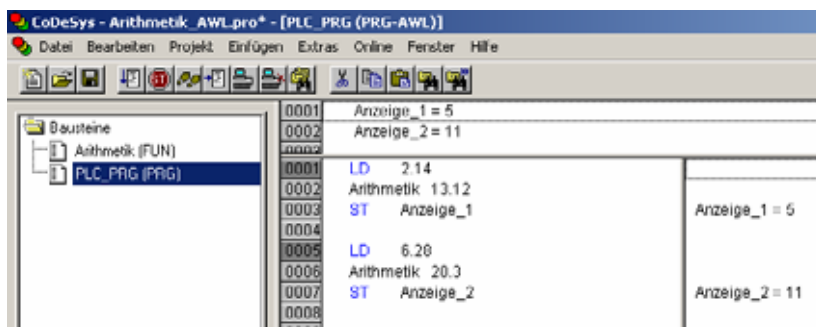


Bild 52: Online Simulation zweier Aufrufe der Funktion „Arithmetik“

| Operation | Erklärung                         | In Step7:   |
|-----------|-----------------------------------|---|
| LD        | Lesen eines Wertes                | U bzw. O der Erstabfrage für binäre, L für digitale Werte |
| ST        | Schreiben (Zuweisen) eines Wertes | = für binäre Werte, T für digitale Werte                  |
| AND, ANDN | UND-Funktion (auch &)             | U, UN   |
| OR, ORN   | ODER-Funktion (auch +)            | O, ON   |
| XOR       | Exklusiv-ODER (Antivalenz)        | XOR   |
| NOT       | Nicht-Funktion (Negation)         | NOT   |
| ADD       | Adition                           | +I, +DI, +R je nach Datentyp                              |
| SUB       | Subtraktion                       | -I, -DI, -R je nach Datentyp                              |
| MUL       | Multiplikation                    | *I, *DI, *R je nach Datentyp                              |
| DIV       | Division                          | /I, /DI, /R je nach Datentyp                              |
| SQRT      | Quadratwurzel                     | SQRT  |
| TRUNC     | Rundung                           | TRUNC   |

|     |                        |                                  |
|-----|------------------------|----------------------------------|
| CAL | Aufruf eines Bausteins | CALL: Unbedingter Bausteinaufruf |
|     |                        |                                  |

Tabelle 11: IEC Operatoren (Teil 1)

### Schlussbemerkung:

In dieser Folge wurde der IEC-gerechte Gebrauch von Funktionen und deren Parametrierung vorgestellt. Funktionen wirken wie komplexe Operatoren. Gegenüber Step 7 ist hier an verschiedenen Stellen ein Umdenken erforderlich. Weiter wurden Vorteile der Programmierung in Anweisungsliste genannt und es wurde begonnen, in dieser Programmiersprache zu arbeiten. In der nächsten Folge werden gleichartige Überlegungen zu Funktionsbausteinen angestellt und das IEC-gerechte Programmieren in AWL vertieft.

### Glossar:

|  |  |
|--|--|
| Siehe auch Glossar der Folgen 1, 3 und 5 |  |
| Rückgabewert                             | Eine Funktion berechnet aus den Eingabewerten diesen eindeutigen Wert und gibt ihn an das Programm zurück. Er wird engl. auch als Return_Value (bei Step 7 auch RET_VAL) bezeichnet.<br>Ab Vers. 5.3 führt auch Step 7 in FC eine solche Variable in der Deklarationstabelle.  |
| SCL                                      | engl. Structured Control Language<br>"Hochsprache" für die Programmierung von Automatisierungstechnik Simatic S7. Sie orientiert sich an der Programmiersprache PASCAL und wurde für die Zertifizierung nach der Norm IEC 61131 vorbereitet. Sie entspricht damit weitgehend der Sprache ST.   |
| ST                                       | engl. Structured Text<br>Eine der 6 in IEC 61131 festgelegten Programmiersprachen. Wie die Anweisungsliste (AWL) ist sie textlich orientiert, jedoch werden die Operationen kompakter geschrieben. Auch gibt es Operatoren, die AWL nicht kennt. ST ist besonders geeignet für die Programmierung komplexer Aufgaben und für Daten- und Rezeptverwaltung. Ursprung ist wie bei SCL die Sprache PASCAL. |