## 14. Zusammengesetzte Datentypen

Neben elementaren Datentypen (Abschnitt 4.4) gibt es zusammengesetzte Datentypen, bei denen gleiche oder auch unterschiedliche elementare Typen zu größeren Einheiten zusammengefasst werden. Vorteil ist dann der elegante Zugriff auf die Gesamtheit der Daten oder aber auf deren einzelnen Elemente.

### 14.1 ARRAY (Feld)

Mit Arrays vereinbart man eine Menge aufeinanderfolgender Daten **gleichen Typs**, z.B. INT oder WORD. Das einzelne Datum des Feldes spricht man dann mit seinem **Index** an. Arrays können **lokal** im Deklarationsteil der POE oder aber **global** deklariert werden.

Neben derartigen eindimensionalen Feldern ermöglicht IEC 61131-3 für komplexe Datenhaltung auch zwei- und dreidimensionale Felder.

- **Beispiel für das Einschreiben von Werten in einen Ringpuffer mit 20 Einträgen**

  Der Eintrag des aktuellen Wertes vom Typ REAL erfolgt im nachfolgenden Beispiel im 2s-Takt zeitgesteuert durch einen 1-Impuls von Zykluszeit am Ausgang des TON-Zeitgliedes. Der Index wurde mit dem Integer 1 initialisiert. Nach jedem Eintrag wird dieser Wert um 1 erhöht. Nach 20 Einträgen wird der Index auf 1 zurückgesetzt, und ältere Werte der Liste werden damit überschrieben.

```
PROGRAM Ringpuffer

VAR
Wert: REAL;                          (*aktueller Gleitpunkt-Wert*)
Liste:ARRAY [1..20] OF REAL;         (*Ringpuffer (Feld) für 20 Werte*)
Index: INT:=1;                       (*laufender Index für die Kennzeichnung der Einträge*)
Takt: TON;                           (*1-Impuls-Geber mit Taktzeit 2 s*)
END_VAR

LDN Takt.Q
ST Takt.IN
CALTakt(PT:= T#2000ms)               (*Zeitsteuerung für Listeneinträge*)

LDN Takt.Q
JMPC ENDE

LD Wert                              (*Listeneintrag an der durch den Index gesteuerten Stelle*)
ST Liste[Index]

LD Index
ADD1                                 (*Nach einem Eintrag wird der Index inkrementiert*)
ST Index

LD Index
EQ 21
JMPCN ENDE

LD 1                                 (*Sofern  20 Listeneinträge erfolgt sind, wird der Index*)
ST Index                             (*erneut auf 1 gesetzt*)

ENDE:
RET
```

### 14.2 Strukturen

Im Gegensatz zu Arrays verwalten Strukturen Daten unterschiedlicher Typen. Sie werden im **CoDeSys Object Organizer** unter der Registerkarte Datentypen deklariert und sind danach **global** im gesamten Projekt bekannt (**Bild 14-1**). Derartige Daten werden auch als **Datentyp UDT** (User defined Typs) geführt.

Die **Schlüsselworte** bei der Deklaration sind:

```
TYPE  Strukturname
STRUCT
Datenelement
:
:
Datenelement
END_STRUCT
END_TYPE
```

Beispiel für eine Struktur:

```
TYPE Datenauswahl_1:
STRUCT
        Auswahlbit:BOOL;
        Kontrollbit:BOOL;
        Fehlermeldung:STRING;
        Stoff_A1:REAL;
        Stoff_A2:REAL;
        Zuschlagstoff:REAL;
        Mischzeit:TIME;
        Zeitstempel:DT;
END_STRUCT
END_TYPE
```

Auf die Elemente einer Struktur greift man mit "**Struktur_Name**" **.** "**Komponentenname**" zu. Will man im Beispiel einen Real-Wert in das Datenelement „Stoff_A" schreiben, so lautet die Anweisung

```
LD 132.54
ST Datenauswahl_1.Stoff_A1
```

- **Anlegen der Struktur im System CoDeSys**

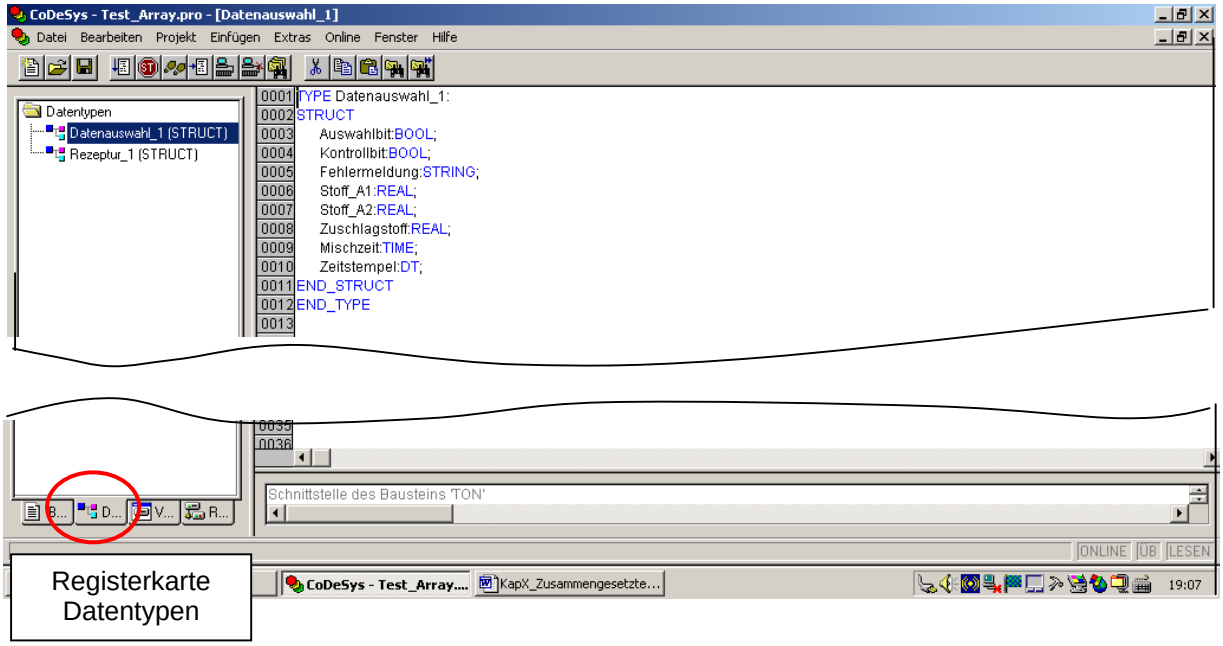



Bild 14-1: Anlegen eines Datentyps

Mit diesem Datentyp würde das in AWL geschriebene Beispiel des zeitgesteuerten Listeneintrages nach Abschnitt 14.1 folgende Form annehmen:

```
PROGRAM Test_Struct

VAR
Daten: Datenauswahl_1;                    (* Datenpaket in der Art der Struktur „Datenauswahl_1“*)
Liste: ARRAY [1..20] OF Datenauswahl_1;   (*Liste mit Einträgen des Datenpaketes*)
Index: INT:=1;
END_VAR

LDN Takt.Q
ST Takt.IN                                (*Zeitsteuerung für Listeneinträge wie  ARRAY
CALTakt(PT := T#2000ms)                   Abschnitt 13.1*)
LDN Takt.Q
JMPC ENDE

LD Daten                                  (*Eintrag des Datenpaketes in eine Liste mit
ST Liste[Index]                           20 Einträgen*)

LD Index
ADD1
ST Index

LD Index
EQ 21
JMPCN ENDE

LD 1
ST Index
ENDE:
RET
```

Im vorliegenden Programm wurde die lokale Variable „Daten“ vom Typ Datenauswahl_1 deklariert, weil der Programmeditor die Anweisungen mit dem UDT

LD Datenauswahl_1
ST Liste[i]

nicht erlaubt.

Um für eine Simulation die Struktur mit Daten zu füllen, könnte man wie nebenstehend ausgeführt Werte in die einzelnen Elemente schreiben.

```
LD      TRUE
ST      Daten.Auswahlbit
LD      FALSE
ST      Daten.Kontrollbit
LD      'Fehler'
ST      Daten.Fehlermeldung
LD      234.56
ST      Daten.Stoff_A1
LD      456.9
ST      Daten.Stoff_A2
LD      12.5
ST      Daten.Zuschlagstoff
LD      t#10s
ST      Daten.Mischzeit
LD      DATE_AND_TIME#1996-05-06-15:36:30
ST      Daten.Zeitstempel
```

Aber auch eine **Initialisierung** der Datenelemente bei der Variablendeklaration ist möglich:

```
VAR
Daten:Datenauswahl_1:=(Auswahlbit:=TRUE,Kontrollbit:=FALSE,Fehlermeldung:='Fehler',
Stoff_A1:=234.56,Stoff_A2:=456.9,Zuschlagstoff:=12.5, Mischzeit:=T#10s,Zeitstempel:=DT#1996-
05-06-15:36:30);
END_VAR
```

Ein Legen auf Adressen ist bei derartigen Datentypen jedoch **nicht** möglich!

### 14.3 Beispiel: Datenverwaltung durch zeitzyklischen Task und Strukturierten Text

Die Beispiele für den Eintrag von Daten in eine Liste in den Abschnitten 13 und 14 zeigen in der Sprache AWL eine relativ große Zahl notwendiger Anweisungen. Solche Aufgaben lassen sich denkbar einfacher mit einem zeitgesteuerten Task anstelle zyklischer Programmbearbeitung und Sprungoperationen lösen. Im **Bild 14-2** wurde ein zeitzyklischer Task mit einer Intervallzeit von 2 Sekunden und einer Priorität von 10 parametriert. Damit wird das Programm „Listeneintrag“ zyklisch alle zwei Sekunden einmal ausgeführt.

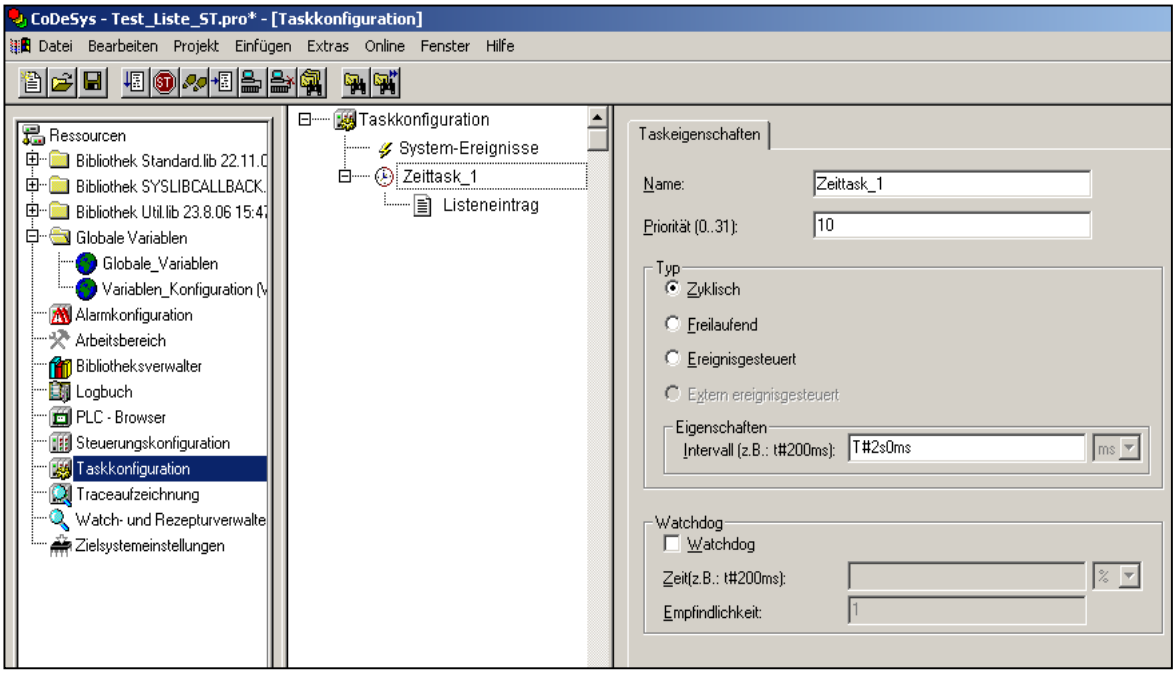



Bild 14-2: Parametrierung eines zeitzyklischen Task im System CoDeSys

In der Programmiersprache ST lautet das gleiche Programm wie oben beschrieben mit einer Initialisierung der Datenelemente für die Zwecke der Simulation:

```
PROGRAM Listeneintrag
VAR
Daten:Datenauswahl_1:=(Auswahlbit:=TRUE,Kontrollbit:=FALSE,Fehlermeldung:='Fehler'
Stoff_A1:=234.56,Stoff_A2:=456.9,Zuschlagstoff:=12.5, Mischzeit:=T#10s,
Zeitstempel:=DT#1996-05-06-15:36:30);

Liste: ARRAY [1..20] OF Datenauswahl_1;
i:INT:=0;
END_VAR

i:=i+1;

IF i>20
THEN i:=1;                          (* IF – Anweisung für 20 Einträge*)
END_IF

Liste[i]:=Daten;                    (*Eintrag der Daten in Zeile i der Liste*)
```

Im vorstehenden Kasten wird die vollständig initialisierte Struktur durch eine einzige Anweisung „Liste[i]:=Daten;“ in die Liste eingetragen.

Das Programm leistet damit den Eintrag aller Elemente der Datenauswahl_1 in den Ringpuffer mit 20 Listeneinträgen (siehe Online-Ansicht der Simulation des Programms **Bild 14-3**). Der Vorteil der Sprache ST gegenüber AWL ist hier deutlich zu erkennen.

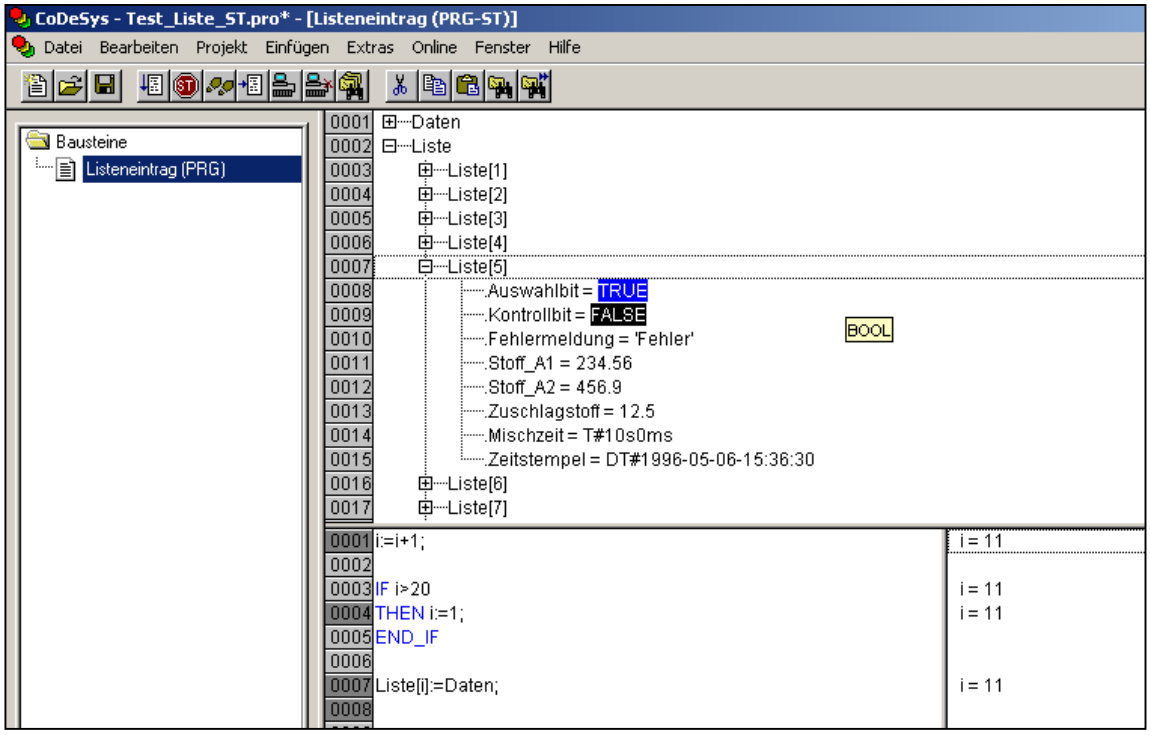



Bild 14-3: Online-Ansicht der Simulation des Programms mit feststehenden Daten: Aufgeblendet ist der Eintrag von Datenauswahl_1 in die Liste Zeile Index 6. Der aktuelle Task steuert derweil bereits den Eintrag in Zeile Index 11.

Wenn veränderliche Wert eines Prozesses in die Liste eingetragen werden sollen, kann dies durch Zugriff auf die Einzelelemente erfolgen:

Anstelle
Liste [i]:=Daten
ist dann zu schreiben:

```
Liste[i].Auswahlbit:=Auswahlbit_1;
Liste[i].Kontrollbit:=Kontrollbit_1;
Liste[i].Fehlermeldung:=Fehlermeldung_1;
Liste[i].Stoff_A1:=Stoff_A1_1;
Liste[i].Stoff_A2:=Stoff_A2_1;
Liste[i].Zuschlagstoff:=Zuschlagstoff_1;
Liste[i].Mischzeit:=Mischzeit_1;
Liste[i].Zeitstempel:=Zeitstempel_1;
```

So kann beispielsweise das Schreiben der Liste für die **Protokollierung eines Prozesses** in einem parametrierbaren Funktionsbaustein erfolgen, dem die Daten über Parameter VAR_INPUT übergeben werden. **Bild 14-4** zeigt ein Beispiel in der Sprache CFC.

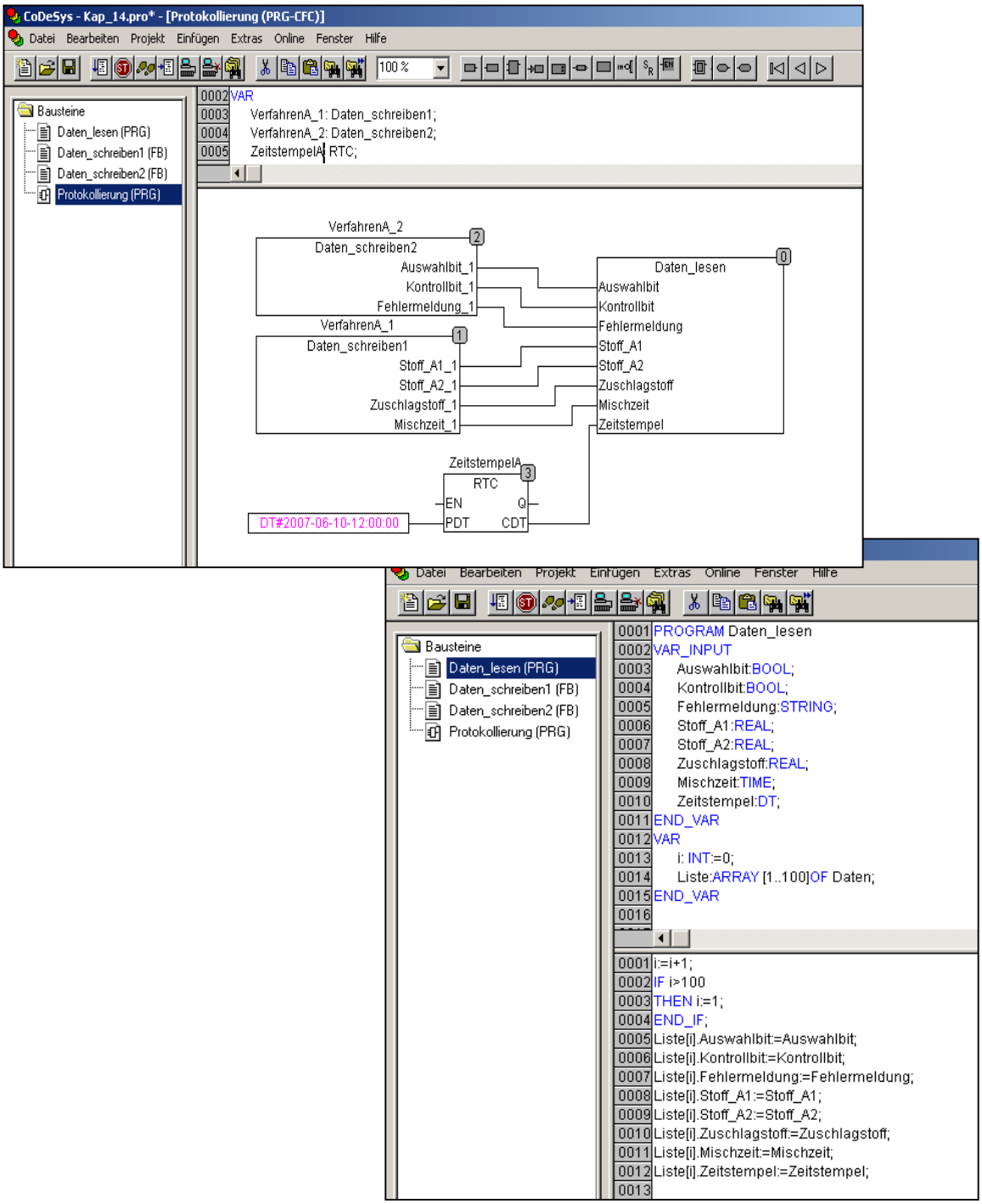
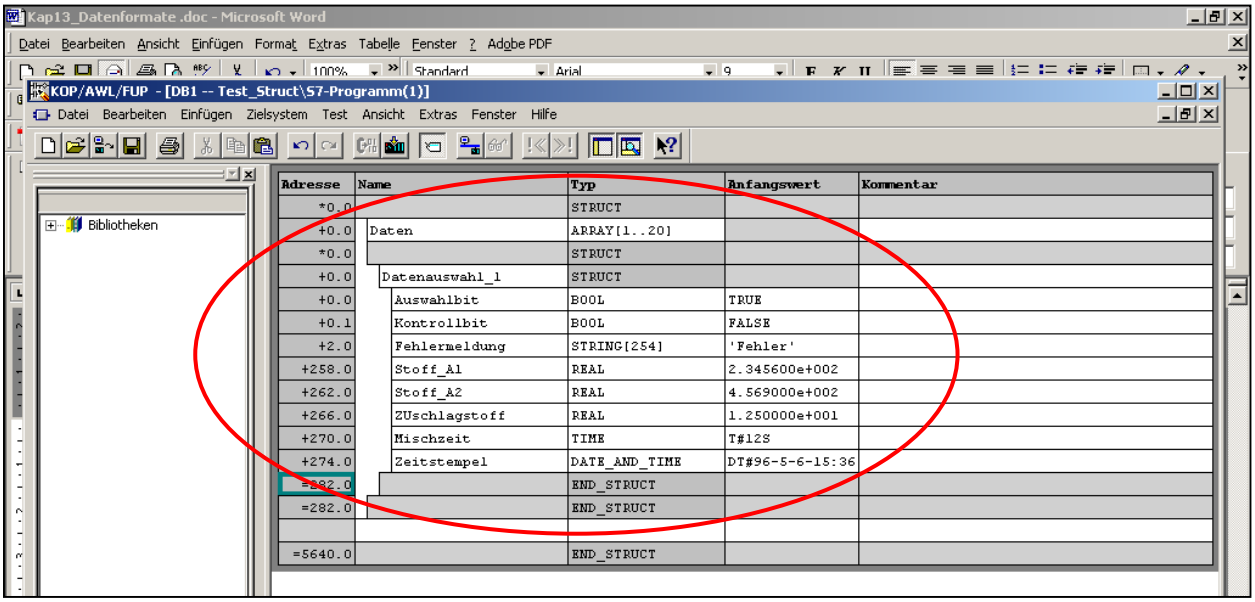



Bild 14-4: Datenübergabe für die Protokollierung eines Prozesses: Übersicht durch Programmierung in CFC, Detailprogrammierung in ST

### 14.4 Zum Vergleich: Komplexe Daten im System Step7

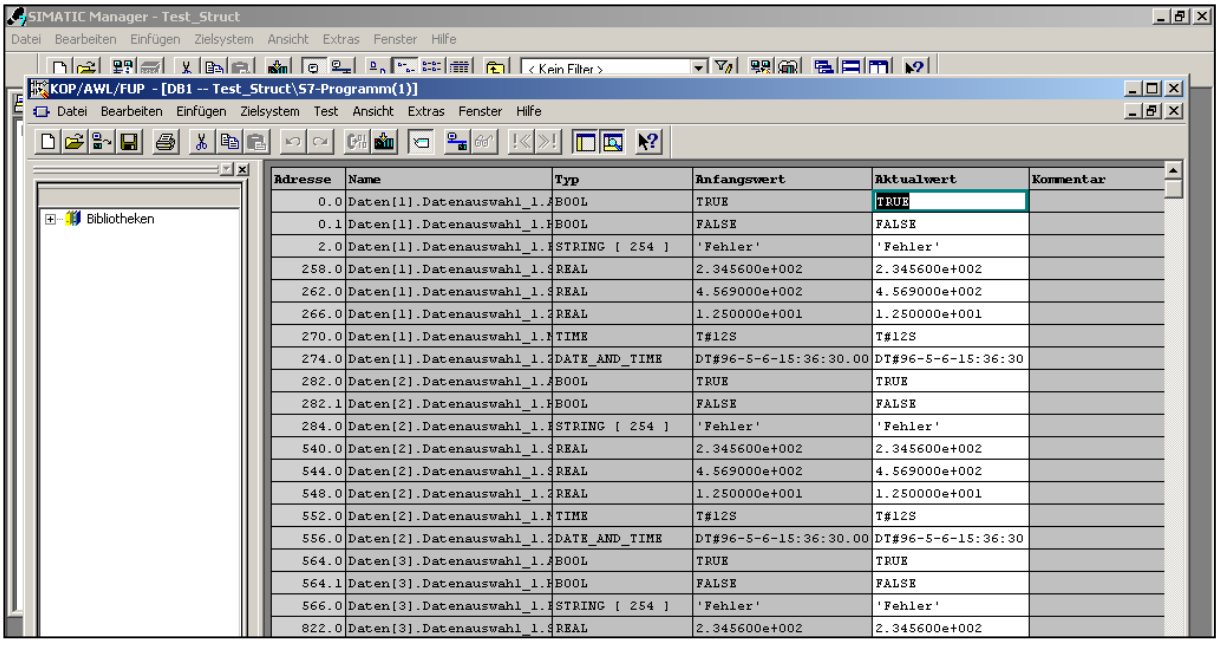
Die hier im Beispiel gezeigte Deklaration von Daten und der Eintrag in eine Liste ist in gleicher Weise im System Step7 in einem globalen Datenbaustein möglich (siehe **Bild 14-5**). Auch eine Vorgabe von Anfangswerten kann dort erfolgen. Diese müssen allerdings gesondert initialisiert werden.




| Adresse | Name | Typ | Anfangswert | Kommentar |
|---|---|---|---|---|
| *0.0 | | STRUCT | | |
| +0.0 | Daten | ARRAY[1..20] | | |
| *0.0 | | STRUCT | | |
| +0.0 | Datenauswahl_1 | STRUCT | | |
| +0.0 | Auswahlbit | BOOL | TRUE | |
| +0.1 | Kontrollbit | BOOL | FALSE | |
| +2.0 | Fehlermeldung | STRING[254] | 'Fehler' | |
| +258.0 | Stoff_A1 | REAL | 2.345600e+002 | |
| +262.0 | Stoff_A2 | REAL | 4.569000e+002 | |
| +266.0 | ZUschlagstoff | REAL | 1.250000e+001 | |
| +270.0 | Mischzeit | TIME | T#12S | |
| +274.0 | Zeitstempel | DATE_AND_TIME | DT#96-5-6-15:36 | |
| =282.0 | | END_STRUCT | | |
| =282.0 | | END_STRUCT | | |
| | | | | |
| =5640.0 | | END_STRUCT | | |

Bild 14-4: Deklaration der Datenstruktur nach Abschnitt 14.3 in Step7 in einem globalen Datenbaustein (Deklarationssicht)

In der Datensicht des Bausteins nach **Bild 14-5** sieht man die Elemente der Liste mit den Eintragungen der hier unveränderlichen Datenelemente entsprechend der vorgegebenen Anfangswerte. Im praktischen Fall würden hier aktuelle veränderliche Werte eingetragen.




| Adresse | Name | Typ | Anfangswert | Aktualwert | Kommentar |
|---|---|---|---|---|---|
| 0.0 | Daten[1].Datenauswahl_1.A | BOOL | TRUE | TRUE | |
| 0.1 | Daten[1].Datenauswahl_1.K | BOOL | FALSE | FALSE | |
| 2.0 | Daten[1].Datenauswahl_1.F | STRING [ 254 ] | 'Fehler' | 'Fehler' | |
| 258.0 | Daten[1].Datenauswahl_1.S | REAL | 2.345600e+002 | 2.345600e+002 | |
| 262.0 | Daten[1].Datenauswahl_1.S | REAL | 4.569000e+002 | 4.569000e+002 | |
| 266.0 | Daten[1].Datenauswahl_1.Z | REAL | 1.250000e+001 | 1.250000e+001 | |
| 270.0 | Daten[1].Datenauswahl_1.M | TIME | T#12S | T#12S | |
| 274.0 | Daten[1].Datenauswahl_1.Z | DATE_AND_TIME | DT#96-5-6-15:36:30.00 | DT#96-5-6-15:36:30 | |
| 282.0 | Daten[2].Datenauswahl_1.A | BOOL | TRUE | TRUE | |
| 282.1 | Daten[2].Datenauswahl_1.K | BOOL | FALSE | FALSE | |
| 284.0 | Daten[2].Datenauswahl_1.F | STRING [ 254 ] | 'Fehler' | 'Fehler' | |
| 540.0 | Daten[2].Datenauswahl_1.S | REAL | 2.345600e+002 | 2.345600e+002 | |
| 544.0 | Daten[2].Datenauswahl_1.S | REAL | 4.569000e+002 | 4.569000e+002 | |
| 548.0 | Daten[2].Datenauswahl_1.S | REAL | 1.250000e+001 | 1.250000e+001 | |
| 552.0 | Daten[2].Datenauswahl_1.M | TIME | T#12S | T#12S | |
| 556.0 | Daten[2].Datenauswahl_1.Z | DATE_AND_TIME | DT#96-5-6-15:36:30.00 | DT#96-5-6-15:36:30 | |
| 564.0 | Daten[3].Datenauswahl_1.A | BOOL | TRUE | TRUE | |
| 564.1 | Daten[3].Datenauswahl_1.K | BOOL | FALSE | FALSE | |
| 566.0 | Daten[3].Datenauswahl_1.F | STRING [ 254 ] | 'Fehler' | 'Fehler' | |
| 822.0 | Daten[3].Datenauswahl_1.S | REAL | 2.345600e+002 | 2.345600e+002 | |

Bild 14-5: Auszug der Datensicht des o.a. globalen Datenbausteins