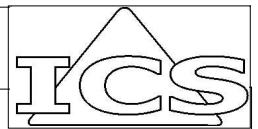


Dateiname: Example\_DL.pro  
Verzeichnis: D:\Andreas\_Schiff\ICSGmbH\ICS\Internet\Internet-PDFs\Deutschf  
Geändert am: 19.11.05 19:30:06 / V2.2  
Bezeichnung: Example\_DL  
Autor: A. Schiff, ICS GmbH  
Version: 1.0 vom 4.10.2004  
Beschreibung: Das Programm lädt Texte in das Anzeigemodul AM002/AM003 und liest sie anschließend der Reihe nach wieder aus.



```

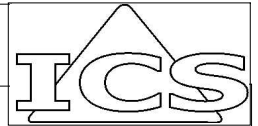
0001 FUNCTION_BLOCK CodeVar2
0002 (* Dieser Funktionsblock codiert eine String-Variable mit Ausgabertext um in ein Datenfeld.
0003 Gleichzeitig können eine oder zwei Variablen in den Text eingefügt werden. Die Positionen und
0004 Formatierungen der Variablen werden durch den Platzhalter '&' gekennzeichnet. Beispiele:
0005 INT=1 und Formatierung: &&&& ergibt: ' 0,1'
0006 INT=-1 und Formatierung: &&&& ergibt: '-0,01'
0007 INT=99 und Formatierung: &&& ergibt: ' 99'
0008 Bei Überlauf wird keine Konvertierung vorgenommen. Größte darstellbare Zahl: 9999, kleinste: -999
0009 größte negative Zahl: -0,01
0010 Achtung: Das Zeichen $ darf im Text nicht vorkommen! *)
0011 VAR_INPUT
0012     Text: STRING(16); (* Unterlegter Text; &&& wird belegt mit Variablen-Werten *)
0013     Var1:INT; (* Variable 1, wird entsprechend Vorgabe durch && umcodiert *)
0014     Var2:INT; (* Variable 2, wird entsprechend Vorgabe durch && umcodiert *)
0015     Var3:INT; (* Variable 3, wird entsprechend Vorgabe durch && umcodiert *)
0016 END_VAR
0017 VAR_OUTPUT
0018     Feld: ARRAY[0..15] OF BYTE;
0019 END_VAR
0020 VAR
0021     i: BYTE;
0022     pByteArray: POINTER TO ARRAY [0..16] OF BYTE;
0023     Zahl1: INT;
0024     Zahl2: INT;
0025     Zahl3: INT;
0026     Zahl4: INT;
0027     ZahlN: INT;
0028     Variable:INT;
0029     Zeichen1: BYTE;
0030     Zeichen2: BYTE;
0031     Zeichen3: BYTE;
0032     Zeichen4: BYTE;
0033     j: BYTE;
0034     k: BYTE;
0035     k2: BYTE;
0036     k1: BYTE;
0037     k3: BYTE;
0038     Komma: BYTE;
0039     Error: BOOL;
0040     k4: BYTE;
0041 END_VAR
0001 pByteArray:=ADR(Text);
0002 FOR i:=0 TO 15 DO
0003     Feld[i]:=pByteArray^[i];
0004     (* ä, ö, ü, °, µ und ß umcodieren *)
0005     IF Feld[i]=16#E4 THEN Feld[i]:=16#E1; END_IF; (* ä *)
0006     IF Feld[i]=16#F6 THEN Feld[i]:=16#EF; END_IF; (* ö *)
0007     IF Feld[i]=16#FC THEN Feld[i]:=16#F5; END_IF; (* ü *)
0008     IF Feld[i]=16#DF THEN Feld[i]:=16#E2; END_IF; (* ß *)
0009     IF Feld[i]=16#B0 THEN Feld[i]:=16#DF; END_IF; (* ° *)
0010     IF Feld[i]=16#B5 THEN Feld[i]:=16#E4; END_IF; (* µ *)
0011     IF Feld[i]=16#A7 THEN Feld[i]:=16#F4; END_IF; (* Ohm *)
0012 END_FOR;
0013
0014 (* Eintragen der Variablen *)
0015 Variable:=Var1;
0016 FOR k:=1 TO 3 DO
0017     IF Variable<=-9999 AND Variable>=-999 THEN
0018         IF Variable>=0 THEN

```

```

0019 (* positiver Zahlenbereich zwischen 0 und +9999 *)
0020   Zahl1:=Variable/1000;
0021   Zahl2:=Variable/100-Zahl1*10;
0022   Zahl3:=Variable/10-Zahl1*100-Zahl2*10;
0023   Zahl4:=Variable -Zahl1*1000-Zahl2*100-Zahl3*10;
0024   Zeichen1:=INT_TO_BYTE(Zahl1)+48;
0025   Zeichen2:=INT_TO_BYTE(Zahl2)+48;
0026   Zeichen3:=INT_TO_BYTE(Zahl3)+48;
0027   Zeichen4:=INT_TO_BYTE(Zahl4)+48;
0028   ELSE
0029 (* negativer Zahlenbereich zwischen -999 und -0,01 *)
0030   ZahlN:=-Variable;
0031   Zahl2:=ZahlN/100;
0032   Zahl3:=ZahlN/10-Zahl2*10;
0033   Zahl4:=ZahlN-Zahl2*100-Zahl3*10;
0034   Zeichen1:=16#2D;
0035   Zeichen2:=INT_TO_BYTE(Zahl2)+48;
0036   Zeichen3:=INT_TO_BYTE(Zahl3)+48;
0037   Zeichen4:=INT_TO_BYTE(Zahl4)+48;
0038   END_IF;
0039   ELSE
0040   Zeichen1:=16#23;
0041   Zeichen2:=16#23;
0042   Zeichen3:=16#23;
0043   Zeichen4:=16#23;
0044   END_IF;
0045
0046 (* Suchen der Stelle, an der die Zahl eingetragen werden soll *)
0047   Komma:=0;
0048   FOR i:=0 TO 15 DO
0049     IF Feld[i]=16#26 THEN
0050       k1:=i; (* Position Feldanfang *)
0051       Feld[i]:=16#23;
0052       FOR j:=1 TO 5 DO
0053         IF Feld[i+j]=16#2C OR Feld[i+j]=16#2E THEN
0054           Komma:=i+j; (* Position Komma oder Punkt*)
0055           k4:=Feld[i+j]; (* Punkt oder Komma *)
0056         END_IF;
0057         IF Feld[i+j]=16#26 THEN
0058           Feld[i+j]:=16#23;
0059         END_IF;
0060         IF Feld[i+j]<>16#23 AND Feld[i+j]<>k4 THEN
0061           k2:=i+j-1; (* Position Feldende *)
0062           k3:=Feld[k2+1]; (* Zeichen hinter Feldende *)
0063           j:=5;
0064         END_IF;
0065       END_FOR;
0066     END_IF;
0067   END_FOR;
0068 END_FOR;
0069
0070 (* Prüfung, ob kein Überlauf oder sonstiger Konflikt bei Darstellung *)
0071 Error:=FALSE;
0072 (* entfällt *)
0073 (* Eintragen der Zeichen *)
0074 IF Error=FALSE THEN
0075   IF k2>=k1 AND Feld[k2]=16#23 THEN
0076     Feld[k2]:=Zeichen4;
0077     k2:=k2-1;

```



```
0078 END_IF;
0079 IF Feld[k2]=k4 THEN k2:=k2-1; END_IF;
0080 IF k2>=k1 AND Feld[k2]=16#23 THEN
0081     Feld[k2]:=Zeichen3;
0082     k2:=k2-1;
0083 END_IF;
0084 IF Feld[k2]=k4 THEN k2:=k2-1; END_IF;
0085 IF k2>=k1 AND Feld[k2]=16#23 THEN
0086     Feld[k2]:=Zeichen2;
0087     k2:=k2-1;
0088 END_IF;
0089 IF Feld[k2]=k4 THEN k2:=k2-1; END_IF;
0090 IF k2>=k1 AND Feld[k2]=16#23 THEN
0091     Feld[k2]:=Zeichen1;
0092     k2:=k2-1;
0093 END_IF;
0094 (* Führende Nullen unterdrücken, Minuszeichen ggf. verschieben *)
0095 IF Feld[k1]=16#30 AND Feld[k1+1]<>k4 AND Feld[k1+1]<>k3 THEN Feld[k1]:=16#20; END_IF;
0096 IF Feld[k1]=16#20 AND Feld[k1+1]=16#30 AND Feld[k1+2]<>k4 AND Feld[k1+2]<>k3 THEN
0097     Feld[k1+1]:=16#20; END_IF;
0098 IF Feld[k1+1]=16#20 AND Feld[k1+2]=16#30 AND Feld[k1+3]<>k4 AND Feld[k1+3]<>k3 THEN
0099     Feld[k1+2]:=16#20; END_IF;
0100 IF Feld[k1]=16#2D AND Feld[k1+1]=16#30 AND Feld[k1+2]<>k4 AND Feld[k1+2]<>k3 THEN
0101     Feld[k1]:=16#20;
0102     Feld[k1+1]:=16#2D;
0103 END_IF;
0104 IF Feld[k1+1]=16#2D AND Feld[k1+2]=16#30 AND Feld[k1+3]<>k4 THEN
0105     Feld[k1+1]:=16#20;
0106     Feld[k1+2]:=16#2D;
0107 END_IF;
0108 END_IF;
0109 IF k=1 THEN
0110     Variable:=Var2;
0111 ELSE
0112     Variable:=Var3;
0113 END_IF;
0114 END_FOR;
```

```

0001 PROGRAM PLC_PRG
0002 (* ----- *)
0003 (*           Anzeige-Text-Ladeprogramm mit Auslesefunktion           *)
0004 (* ----- *)
0005 (* ICS GmbH, Hopfenstraße 1, 88069 Tettnang, Autor: Andreas Schiff       Version 1.0   Datum: 4.10.04 *)
0006 (* ----- *)
0007 (* Dieses Beispielprogramm dient nur zur Demonstration der Funktion des AM002/3. Es kann keine Gewähr dafür *)
0008 (* übernommen werden, dass dieses Programm in einem realen Automatisierungsprojekt fehlerfrei läuft. *)
0009 (* ----- *)
0010
0011 VAR
0012     DI2 AT %IX1.5.2:BOOL;           (* Datenports für serielle Kommunikation *)
0013     DI3 AT %IX1.5.3:BOOL;           (* Slave ist auf Adresse 5 *)
0014     DO0 AT %QX1.5.0: BOOL;
0015     DO1 AT %QX1.5.1:BOOL;
0016     Version: BYTE:=10;              (* Programm- und Versionsnummer des Programms *)
0017     ZNr: BYTE;
0018     Text1: BYTE;
0019     Text2: BYTE;
0020     Texx: ARRAY [5..127] OF STRING:= 'Dies ist Text1..',      (* Hier werden die Soll-Texte der Reihe nach eingetragen *)
0021     '..und dies Text2',
0022     'www.ICS-GmbH.com',
0023     '+++++++';
0024     AnzB1: ARRAY [0..15] OF BYTE;    (* erste Zeile Anzeigemodul (Slaveadresse 4) *)
0025     AnzB2: ARRAY [0..15] OF BYTE;    (* zweite Zeile Anzeigemodul *)
0026     AnzeigeB: SerComA;
0027     Initial:BOOL:=TRUE;
0028     StepNr:BYTE;
0029     Warten:WORD;
0030     Taktgeber1:TON;
0031     Taktgeber2:TON;
0032     HB1:BOOL;
0033     Takt1:BOOL;
0034     Taktgeber3:TON;
0035     Taktgeber4:TON;
0036     HB2:BOOL;
0037     Takt2:BOOL;
0038     Textvar2:CodeVar2;
0039     ZeilNr: BYTE;
0040     ZMax: BYTE;
0041     Z2Nr: BYTE;
0042     i: BYTE;
0043 END_VAR
0044
0045 (* Ende des Deklarationsteils *)
0001 (* Takt2: 500ms *)
0002 Taktgeber3(IN:=Takt2,PT:=t#250ms);
0003 HB2:=NOT Taktgeber3.Q;
0004 Taktgeber4(IN:=HB2,PT:=t#250ms);
0005 Takt2:=Taktgeber4.Q;
0006
0007 AnzeigeB(DI2:=DI2, DI3:=DI3, DatIn1:=AnzB1, DatIn2:=AnzB2, Zeile1:=ZNr, Zeile2:=Z2Nr,
0008     TextNr1:=Text1, TextNr2:=Text2);
0009 DO0:=AnzeigeB.DO0;
0010 DO1:=AnzeigeB.DO1;
0011
0012 (* ----- Initialisieren ----- *)
0013 (* Initialisieren beim Zuschalten der Betriebsspannung *)
0014 IF Initial=TRUE THEN

```

```

0015 IF StepNr=0 THEN
0016     TextVar2(Text:='Ladeprogramm &.&',Var1:=Version);
0017     AnzB1:=TextVar2.Feld;
0018     TextVar2(Text:='lade Texte ...');
0019     AnzB2:=TextVar2.Feld;
0020     Text1:=0;
0021     Text2:=0;
0022     ZNr:=3;
0023     Z2Nr:=4;
0024     StepNr:=1;
0025     Warten:=0;
0026 END_IF;
0027
0028 IF StepNr=1 THEN
0029     Warten:=Warten+1;
0030     IF Warten>6000 THEN
0031         ZNr:=0;
0032         i:=4;
0033         Z2Nr:=0;
0034         Warten:=0;
0035         StepNr:=StepNr+1;
0036     END_IF;
0037 END_IF;
0038
0039 IF StepNr=2 THEN
0040     Warten:=Warten+1;
0041     IF Warten>1500 OR (AnzeigeB.Diagnosis=0) THEN
0042         i:=i+1;
0043         TextVar2(Text:=Texx[i]);
0044         AnzB1:=TextVar2.Feld;
0045         ZNr:=i;
0046         Warten:=0;
0047         StepNr:=StepNr+1;
0048     END_IF;
0049 END_IF;
0050
0051 IF StepNr=3 THEN
0052     IF Texx[ZNr]='+++++' THEN
0053         ZMax:=ZNr;
0054         Warten:=Warten+1;
0055         IF Warten>2000 THEN
0056             Initial:=FALSE;
0057             ZeilNr:=5;
0058             ZNr:=0;
0059         END_IF;
0060     ELSE
0061         StepNr:=2;
0062     END_IF;
0063 END_IF;
0064
0065 ELSE
0066
0067 (* ----- Hauptprogramm ----- *)
0068 (* Hier werden alle geladenen Texte der Reihe nach angezeigt *)
0069
0070 Warten:=Warten+1;
0071 IF Warten>1000 THEN
0072     ZNr:=0;
0073 END_IF;

```

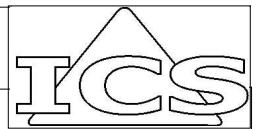
```
0074 IF Warten>1600 THEN
0075     TextVar2(Text:='Text Nr. &&&: ',Var1:=ZeilNr);
0076     AnzB1:=TextVar2.Feld;
0077     ZNr:=3;
0078     Text1:=3;
0079     Text2:=ZeilNr;
0080     ZeilNr:=ZeilNr+1;
0081     IF ZeilNr>ZMax THEN
0082         ZeilNr:=5;
0083     END_IF;
0084     Warten:=0;
0085     END_IF;
0086
0087 END_IF;
0088
0089
0090
```

Beispielprogramm

```

0001 FUNCTION_BLOCK SerComA
0002 (* ----- *)
0003 (* Dieser Funktionsbaustein wickelt die gesamte serielle Kommunikation zu einem AS-Interface Slave nach dem *)
0004 (* Profil S-B.A.5 ab. Gilt nur für 2zeilige Ausgabe! *)
0005 (* *)
0006 (* ICS GmbH, Hopfenstraße 1, 88069 Tettnang, Autor: Andreas Schiff Version 0.1 Datum: 4.12.03 *)
0007 (* *)
0008 (* ----- *)
0009
0010 VAR_INPUT
0011     DI2: BOOL;
0012     DI3: BOOL;
0013     DatIn1: ARRAY [0..15] OF BYTE; (* Parameter-Solldaten Zeile 1 *)
0014     DatIn2: ARRAY [0..15] OF BYTE; (* Parameter-Solldaten Zeile 2 *)
0015     Zeile1: BYTE; (* Zeilennummer fürs Schreiben bzw. Zeile 1 *)
0016     Zeile2: BYTE; (* Zeilennummer für Zeile 2 *)
0017     TextNr1: BYTE; (* Zeilennummer 1 fürs Ausgeben *)
0018     TextNr2: BYTE; (* Zeilennummer 2 fürs Ausgeben *)
0019 END_VAR
0020
0021 VAR_OUTPUT
0022     Diagnosis: BYTE; (* Diagnose-Daten: Bit 7: Slave nicht im seriellen Modus
0023                       Bit 6: ID nicht korrekt
0024                       Bit 5: ---
0025                       Bit 4: Slave busy
0026                       Bit 0..3: Diagnoseinfo vom Slave *)
0027     ID_Data: ARRAY [0..15] OF BYTE; (* ID-Daten des Slaves *)
0028     DO0: BOOL;
0029     DO1: BOOL;
0030 END_VAR
0031
0032 VAR
0033     InputD: ARRAY [0..20] OF BYTE; (* Eingabe-Datenfeld *)
0034     InputData: ARRAY [0..20] OF BYTE; (* Eingabe-Datenfeld *)
0035     OutputData: ARRAY [0..20] OF BYTE; (* Ausgabe-Datenfeld *)
0036     OutputLen: BYTE; (* Länge der Ausgabedaten in Byte; Zahlenbereich 1..15 *)
0037     InputL: BYTE; (* Länge der Eingabedaten in Byte; Zahlenbereich 1..15 *)
0038     InputLen: BYTE;
0039     Taktalt: BOOL;
0040     In: BYTE; (* aktuell empfangene Nachricht *)
0041     Inalt: BYTE; (* letzte empfangene Nachricht *)
0042     Out: BYTE:= 3;
0043     Parameter: BYTE:=1;
0044     Watchdog: BOOL;
0045     Timeout: BYTE;
0046     LastIn: BYTE; (* letztes gültiges empfangenes Informationsbit *)
0047     i: BYTE;
0048     IBitzeiger: BYTE;
0049     Outalt: BYTE;
0050     Outzeiger: BYTE;
0051     OBitzeiger: BYTE;
0052     x: BYTE;
0053     Send: BOOL;
0054     y: BYTE;
0055     Outx: BYTE;
0056     Inx: BYTE;
0057     NewData: BOOL;
0058     SCSlave: BOOL;
0059     SSW: WORD;

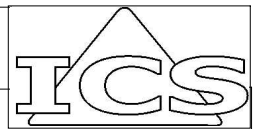
```



```

0060 Inst1:TON;
0061 Inst2:TON;
0062 Takt:BOOL;
0063 Var2: BOOL;
0064 Acyclic: WORD;
0065 Para_Data: ARRAY [0..20] OF BYTE;
0066 Para_Data2: ARRAY [0..20] OF BYTE;
0067 Toggle: BYTE;
0068 Timeout2: BYTE;
0069 MerkDat1: BOOL;
0070 MerkDat2: BOOL;
0071 Text1alt: BYTE;
0072 Text2alt: BYTE;
0073 Zeile1alt: BYTE;
0074 Zeile2alt: BYTE;
0075 END_VAR
0076
0077 (* -----Ende des Deklarationsteils----- *)
0001 (* Hier wird ein Takt mit einer Zykluszeit von 20ms generiert *)
0002 Inst1(IN:=Takt, PT:=t#10ms);
0003 Var2:=NOT Inst1.Q;
0004 Inst2(IN:=Var2, PT:=t#10ms);
0005 Takt:=Inst2.Q;
0006
0007 (* Eingänge des Slaves lesen, ausmaskieren und auf Veränderung prüfen *)
0008 IF DI2=FALSE AND DI3=FALSE THEN In:=0; END_IF;
0009 IF DI2=FALSE AND DI3=TRUE THEN In:=2; END_IF;
0010 IF DI2=TRUE AND DI3=FALSE THEN In:=1; END_IF;
0011 IF DI2=TRUE AND DI3=TRUE THEN In:=3; END_IF;
0012
0013 IF In<>Inalt THEN
0014     (* eine Antwort ist eingetroffen *)
0015     IF Out=1 AND In=1 THEN
0016         (* die eine der erwarteten Antworten ist eingetroffen: Separator *)
0017         Watchdog:=FALSE;
0018         SCSlave:=TRUE;
0019         Timeout:=0;
0020         Lastin:=Inalt;
0021         (* Aktion 2: Trenn-Nachricht empfangen; Master sendet Information oder Idle *)
0022         IF Send=TRUE THEN
0023             IF Outalt=3 THEN
0024                 (* Ausgabe initialisieren *)
0025                 Outzeiger:=0;
0026                 OBitzeiger:=0;
0027                 y:=OutputData[0];
0028             END_IF;
0029             Outalt:=Out;
0030             x:=y AND 2#1000_0000;
0031             IF x<>0 THEN
0032                 Out:=2;
0033             ELSE
0034                 Out:=0;
0035             END_IF;
0036             y:=SHL(y,1);
0037             OBitzeiger:=OBitzeiger+1;
0038             (* ein Byte fertig? Zeiger auf nächstes Byte stellen *)
0039             IF OBitzeiger>7 THEN
0040                 Outzeiger:=Outzeiger+1;
0041                 y:=OutputData[Outzeiger];

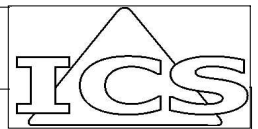
```



```

0042     OBitzeiger:=0;
0043     IF Outzeiger>=OutputLen OR Outzeiger>=19 THEN
0044         Send:=FALSE;
0045         SCSSlave:=TRUE;
0046     END_IF;
0047     END_IF;
0048     ELSE
0049         Outalt:=Out;
0050         Out:=3;
0051     END_IF;
0052     IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0053     IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0054     IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0055     IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0056     ELSIF Out<>1 AND In<>1 THEN
0057         (* die andere der erwarteten Antworten ist eingetroffen: Idle oder Daten *)
0058         Watchdog:=FALSE;
0059         SCSSlave:=TRUE;
0060         Timeout:=0;
0061         IF (In=0 AND Inalt<>2) OR (In=2 AND Inalt<>0) THEN
0062             (* Aktion 0/1: Slave hat Informationsbit gesendet: abspeichern; Master sendet Trenn-Nachricht *)
0063             IF Lastin=3 THEN
0064                 (* Beginn einer Nachricht: Zeiger und Feld Initialisieren, wenn Feld bereit *)
0065                 NewData:=FALSE;
0066                 InputL:=0;
0067                 IBitzeiger:=0;
0068                 FOR i:=0 TO 19 DO
0069                     InputD[i]:=0;
0070                 END_FOR;
0071             END_IF;
0072             InputD[InputL]:=SHL(InputD[InputL],1);
0073             IF In=2 THEN
0074                 InputD[InputL]:=InputD[InputL]+1;
0075             END_IF;
0076             IBitzeiger:=IBitzeiger+1;
0077             IF IBitzeiger>7 THEN
0078                 InputL:=InputL+1;
0079                 IF InputL>19 THEN
0080                     InputL:=19;
0081                 END_IF;
0082                 IBitzeiger:=0;
0083             END_IF;
0084             Outalt:=Out;
0085             Out:=1;
0086             IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0087             IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0088             IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0089             IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0090         ELSE
0091             (* Aktion 3: Slave hat Idle gesendet: Abschluss einer Nachricht?; Master sendet Trenn-Nachricht *)
0092             IF Lastin<>3 THEN
0093                 (* Abschluss einer Nachricht; wenn diese nicht vollständig (also ganzzahlige Anzahl von Bytes),
0094                 dann wird sie verworfen! *)
0095                 IF IBitzeiger=0 THEN
0096                     FOR i:=0 TO InputL DO
0097                         InputData[i]:=InputD[i];
0098                     END_FOR;
0099                     NewData:=TRUE;
0100                     InputLen:=InputL;

```



```

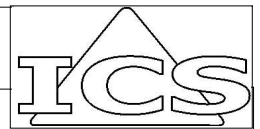
0101         SCSSlave:=TRUE;
0102         END_IF;
0103     END_IF;
0104     Outalt:=Out;
0105     Out:=1;
0106     IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0107     IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0108     IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0109     IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0110     END_IF;
0111     END_IF;
0112 ELSIF (Takt<>Taktalt AND Takt=TRUE) THEN
0113     (* Slave alle 120ms testen, ob er den seriellen Kommunikationsmodus unterstützt, dient auch zum Anlauf
0114     der Kommunikation *)
0115     Timeout:=Timeout+1;
0116     IF Timeout>4 THEN
0117         Outalt:=Out;
0118         IF (Out=0 OR Out=3) THEN
0119             Out:=1;
0120         ELSIF (Out=1 OR Out=2) THEN
0121             Out:=3;
0122         END_IF;
0123         Timeout:=0;
0124         SCSSlave:=FALSE;
0125         SSW:=0;
0126         Para_Data[1]:=0;
0127         Para_Data2[1]:=0;
0128         Send:=FALSE;
0129         IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0130         IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0131         IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0132         IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0133     END_IF;
0134 END_IF;
0135
0136 (* Hier läuft die Timeoutüberwachung für azyklische Dienste ab: 1s *)
0137 IF Taktalt<>Takt AND Takt=FALSE AND Acyclic<>0 THEN
0138     Timeout2:=Timeout2+1;
0139     IF Timeout2>151 THEN
0140         Acyclic:=0;
0141         Timeout2:=0;
0142         MerkDat1:=FALSE;
0143         MerkDat2:=FALSE;
0144         Diagnosis:=Diagnosis AND 16#EF;
0145     END_IF;
0146 END_IF;
0147
0148 (* Liegt ein Kommandoaufruf vor? *)
0149 IF ID_Data[1]=0 AND Acyclic=0 AND Send=FALSE AND SCSSlave=TRUE THEN
0150 (* ID des Slaves abfragen *)
0151     OutputData[0]:=16;
0152     OutputData[1]:=0;
0153     OutputData[2]:=6;
0154     OutputLen:=3;
0155     Send:=TRUE;
0156     Acyclic:=100;
0157     Diagnosis:=Diagnosis OR 16#10;
0158 END_IF;
0159

```

```

0160 (* Wenn Veränderung in Datenzeile, dann entsprechende Zeile schreiben *)
0161 IF Acyclic=0 AND Send=FALSE AND SCSlave=TRUE THEN
0162     IF Zeile1>2 AND Zeile1<>Zeile1alt THEN
0163         OutputData[0]:=17;
0164         OutputData[1]:=Zeile1;
0165         OutputData[2]:=16;
0166         FOR i:=0 TO 15 DO
0167             OutputData[i+3]:=DatIn1[i];
0168         END_FOR;
0169         OutputLen:=19;
0170         Send:=TRUE;
0171         Zeile1alt:=Zeile1;
0172         Acyclic:=103;
0173         Diagnosis:=Diagnosis OR 16#10;
0174     END_IF;
0175 END_IF;
0176
0177 (* Wenn Veränderung in Datenzeile, dann entsprechende Zeile schreiben *)
0178 IF Acyclic=0 AND Send=FALSE AND SCSlave=TRUE THEN
0179     IF Zeile2>2 AND Zeile2<>Zeile2alt THEN
0180         OutputData[0]:=17;
0181         OutputData[1]:=Zeile2;
0182         OutputData[2]:=16;
0183         FOR i:=0 TO 15 DO
0184             OutputData[i+3]:=DatIn2[i];
0185         END_FOR;
0186         OutputLen:=19;
0187         Send:=TRUE;
0188         Zeile2alt:=Zeile2;
0189         Acyclic:=104;
0190         Diagnosis:=Diagnosis OR 16#10;
0191     END_IF;
0192 END_IF;
0193
0194 (* Wenn TextNr1 oder TextNr2 >4, dann cyclic Aufruf absetzen *)
0195 IF Send=FALSE AND Acyclic=0 AND SCSlave=TRUE THEN
0196     IF (TextNr1>4 OR TextNr2>4) AND (TextNr1<>Text1alt OR TextNr2<>Text2alt) THEN
0197         OutputData[0]:=1;
0198         OutputData[1]:=TextNr1;
0199         OutputData[2]:=TextNr2;
0200         OutputLen:=3;
0201         Text1alt:=TextNr1;
0202         Text2alt:=TextNr2;
0203         Send:=TRUE;
0204     END_IF;
0205 END_IF;
0206
0207 IF Newdata=TRUE THEN
0208     (* ID-Daten sind angekommen *)
0209     IF InputData[0]=80 AND Acyclic=100 THEN
0210         FOR i:=0 TO 5 DO
0211             ID_Data[i]:=InputData[i+1];
0212         END_FOR;
0213         NewData:=FALSE;
0214         Timeout2:=0;
0215         Acyclic:=0;
0216         IF ID_Data[1]<>1 THEN
0217             Diagnosis:=Diagnosis OR 16#40;
0218         ELSE

```



```

0219     Diagnosis:=Diagnosis AND 16#BF;
0220     END_IF;
0221     Diagnosis:=Diagnosis AND 16#EF;
0222     END_IF;
0223 (* Diagnosedaten sind angekommen *)
0224     IF InputData[0]=80 AND Acyclic=101 THEN
0225         Diagnosis:=InputData[1];
0226         NewData:=FALSE;
0227         Timeout2:=0;
0228         Acyclic:=0;
0229         Diagnosis:=Diagnosis AND 16#EF;
0230     END_IF;
0231 (* Parameter-Istdaten sind angekommen *)
0232     IF InputData[0]=81 AND Acyclic=102 THEN
0233         NewData:=FALSE;
0234         Timeout2:=0;
0235         Acyclic:=0;
0236         Diagnosis:=Diagnosis AND 16#EF;
0237     END_IF;
0238 (* Acknowledge für Zeile 1 Schreiben ist angekommen *)
0239     IF InputData[0]=81 AND Acyclic=103 THEN
0240         NewData:=FALSE;
0241         Timeout2:=0;
0242         Acyclic:=0;
0243         Diagnosis:=Diagnosis AND 16#EF;
0244     END_IF;
0245 (* Acknowledge für Zeile 2 Schreiben ist angekommen *)
0246     IF InputData[0]=81 AND Acyclic=104 THEN
0247         NewData:=FALSE;
0248         Timeout2:=0;
0249         Acyclic:=0;
0250         Diagnosis:=Diagnosis AND 16#EF;
0251     END_IF;
0252 (* Fehlerhafte Aufrufe*)
0253     IF InputData[0]=145 OR InputData[0]=144 THEN
0254         Diagnosis:=InputData[1];
0255         NewData:=FALSE;
0256         Timeout2:=0;
0257         Acyclic:=0;
0258         Diagnosis:=Diagnosis AND 16#EF;
0259     END_IF;
0260 (* irgendein Fehler ist angekommen *)
0261     IF (InputData[0]=81 OR InputData[0]=0) AND Acyclic=0 THEN
0262         NewData:=FALSE;
0263         Diagnosis:=Diagnosis AND 16#EF;
0264     END_IF;
0265 END_IF;
0266
0267 Taktalt:=Takt;
0268 IF Zeile1<3 THEN
0269     Zeile1alt:=Zeile1;
0270 END_IF;
0271 IF Zeile2<3 THEN
0272     Zeile2alt:=Zeile2;
0273 END_IF;
0274 Inalt:=In;

```

Projektinformationen  
CodeVar2 (FB-ST)  
PLC\_PRG (PRG-ST)  
SerComA (FB-ST)

A  
1-1  
2-1  
3-1

Beispielprogramm