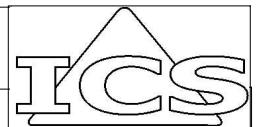


Dateiname: Example_TempSens.pro
Verzeichnis: D:\Andreas_Schiff\ICSGmbH\ICS\Internet\Internet-PDFs\Deutschf
Geändert am: 19.11.05 19:30:43 / V2.2
Bezeichnung: Example_TempSens
Autor: A. Schiff, ICS GmbH
Version: 0.6 vom 8.10.2004
Beschreibung: Das Programm kommuniziert mit dem
Temperatursensor TS001, liest die
Eingangsvariable und schreibt die
Parameter



```

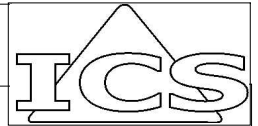
0001 PROGRAM PLC_PRG
0002 (* ----- *)
0003 (*           Parameter-Ausleseprogramm für AS-Interface TS001, 2004           *)
0004 (* ----- *)
0005 (* Dieses Beispielprogramm dient nur zur Demonstration der Funktion des AM004. Es kann keine Gewähr dafür *)
0006 (* übernommen werden, dass dieses Programm in einem realen Automatisierungsprojekt fehlerfrei läuft. *)
0007 (* ----- *)
0008 VAR
0009 (* Variablen für E/As *)
0010     DI2 AT %IX1.5.2: BOOL;                (* Ein- und Ausgänge für die serielle Kommunikation *)
0011     DI3 AT %IX1.5.3: BOOL;
0012     DO0 AT %QX1.5.0: BOOL;
0013     DO1 AT %QX1.5.1: BOOL;
0014 (* Hilfsvariablen *)
0015     Limit1:WORD;
0016     Limit2:WORD;
0017     Limit3:WORD;
0018     Limit4:WORD;
0019     Parameter: ARRAY[1..8] OF BYTE:=0,0,0,0,0,0,0,0;
0020     TempSens: SerComTS;
0021     iw:INT;
0022     SW2O: WORD;
0023     SW1U: WORD;
0024     Product_ID: WORD;
0025     Vendor_ID: WORD;
0026     Comp: BYTE;
0027     Vers: BYTE;
0028 END_VAR
0029
0030 (* Ende des Deklarationsteils *)
0001 Parameter[1]:=INT_TO_BYTE(Limit1/256);
0002 Parameter[2]:=INT_TO_BYTE(Limit1-Parameter[1]*256);
0003 Parameter[3]:=INT_TO_BYTE(Limit2/256);
0004 Parameter[4]:=INT_TO_BYTE(Limit2-Parameter[3]*256);
0005 Parameter[5]:=INT_TO_BYTE(Limit3/256);
0006 Parameter[6]:=INT_TO_BYTE(Limit3-Parameter[5]*256);
0007 Parameter[7]:=INT_TO_BYTE(Limit4/256);
0008 Parameter[8]:=INT_TO_BYTE(Limit4-Parameter[7]*256);
0009
0010 TempSens(DI2:=DI2, DI3:=DI3, Para_DatIn:=Parameter);
0011 DO1:=TempSens.DO1;
0012 DO0:=TempSens.DO0;
0013 iw:=TempSens.InputWord;
0014 Vendor_ID:=TempSens.ID_Data[1]*256+TempSens.ID_Data[2];
0015 Product_ID:=TempSens.ID_Data[3]*256+TempSens.ID_Data[4];
0016 Comp:=TempSens.ID_Data[5];
0017 Vers:=TempSens.ID_Data[6];
0018 SW1U:=TempSens.Para_DatOut[3]*256+TempSens.Para_DatOut[4];
0019 SW2O:=TempSens.Para_DatOut[5]*256+TempSens.Para_DatOut[6];

```

```

0001 FUNCTION_BLOCK SerComTS
0002 (* Dieser Funktionsbaustein wickelt die gesamte serielle Kommunikation zu einem AS-Interface Slave nach dem
0003 Profil S-7.A.5 ab. *)
0004
0005 VAR_INPUT
0006     DI2:      BOOL;          (* Eingänge für serielle Kommunikation *)
0007     DI3:      BOOL;
0008     Para_DatIn: ARRAY [1..8] OF BYTE :=2,0,7,208,3,232; (* Parameter-Solldaten *)
0009 END_VAR
0010
0011 VAR_OUTPUT
0012     InputWord: INT;          (* Eingabe-Datenfeld: Zählerstand *)
0013     Diagnosis: BYTE;        (* Diagnose-Daten *)
0014     ID_Data:   ARRAY [0..15] OF BYTE; (* ID-Daten des Slaves *)
0015     Para_DatOut: ARRAY [1..8] OF BYTE; (* Parameter-Istdaten des Slaves *)
0016     DO1:      BOOL;          (* Ausgänge für serielle Kommunikation *)
0017     DO0:      BOOL;
0018 END_VAR
0019
0020 VAR
0021     InputD:   ARRAY [0..15] OF BYTE; (* Eingabe-Datenfeld *)
0022     InputData: ARRAY [0..15] OF BYTE; (* Eingabe-Datenfeld *)
0023     OutputData: ARRAY [0..15] OF BYTE; (* Ausgabe-Datenfeld *)
0024     OutputLen:  BYTE;              (* Länge der Ausgabedaten in Byte; Zahlenbereich 1..15 *)
0025     InputL:    BYTE;              (* Länge der Eingabedaten in Byte; Zahlenbereich 1..15 *)
0026     InputLen:  BYTE;
0027     Taktalt:   BOOL;
0028     In:        BYTE;              (* aktuell empfangene Nachricht *)
0029     Inalt:     BYTE;              (* letzte empfangene Nachricht *)
0030     Out:       BYTE:= 3;
0031     Watchdog:  BOOL;
0032     Timeout:   BYTE;
0033     Lastin:    BYTE;              (* letztes gültiges empfangenes Informationsbit *)
0034     i:         BYTE;
0035     IBitzeiger: BYTE;
0036     Outalt:    BYTE;
0037     Outzeiger: BYTE;
0038     OBitzeiger: BYTE;
0039     x:         BYTE;
0040     Send:      BOOL;
0041     y:         BYTE;
0042     NewData:   BOOL;
0043     SSW:       WORD;
0044     Inst1:TON;
0045     Inst2:TON;
0046     Takt:BOOL;
0047     Var2:      BOOL;
0048     Acyclic:   WORD;
0049     Para_Data: ARRAY [0..8] OF BYTE;
0050     MerkPara:  BOOL;
0051     SCSlave:   BOOL;
0052     Schluss:   BOOL;
0053     Timeout2:  BYTE;
0054     ARS0Request: BOOL;
0055     ARSResponse_0: BOOL;
0056     ARS1Request: BOOL;
0057     ARSResponse_1: BOOL;
0058     AWS2Request: BOOL;
0059     ARSResponse_2: BOOL;

```

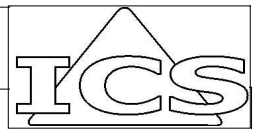


```

0060   AWSResponse_2: BOOL;
0061   ARS2Request: BOOL;
0062 END_VAR
0063
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             IF Schluss=FALSE THEN
0030                 Outalt:=Out;
0031                 x:=y AND 2#1000_0000;
0032                 IF x<>0 THEN
0033                     Out:=2;
0034                 ELSE
0035                     Out:=0;
0036                 END_IF;
0037                 y:=SHL(y,1);
0038                 OBitzeiger:=OBitzeiger+1;
0039                 (* ein Byte fertig? Zeiger auf nächstes Byte stellen *)
0040                 IF OBitzeiger>7 THEN
0041                     Outzeiger:=Outzeiger+1;
0042                     y:=OutputData[Outzeiger];
0043                     OBitzeiger:=0;
0044                     IF Outzeiger>=OutputLen OR Outzeiger>=19 THEN
0045                         Schluss:=TRUE;
0046                         SCSlave:=TRUE;
0047                     END_IF;
0048                 END_IF;
0049             ELSE
0050                 Outalt:=Out;
0051                 Out:=3;
0052                 Send:=FALSE;
0053                 Schluss:=FALSE;
0054             END_IF;
0055         ELSE

```

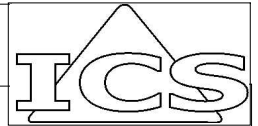
(* zum Abschluss einer Nachricht wird mindestens 1 Idle gesendet! *)



```

0056      Outalt:=Out;
0057      Out:=3;
0058      END_IF;
0059      IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0060      IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0061      IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0062      IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0063      ELSIF Out<>1 AND In<>1 THEN
0064          (* die andere der erwarteten Antworten ist eingetroffen: Idle oder Daten *)
0065          Watchdog:=FALSE;
0066          SCSlave:=TRUE;
0067          Timeout:=0;
0068          IF (In=0 AND Inalt<>2) OR (In=2 AND Inalt<>0) THEN
0069              (* Aktion 0/1: Slave hat Informationsbit gesendet: abspeichern; Master sendet Trenn-Nachricht *)
0070              IF Lastin=3 THEN
0071                  (* Beginn einer Nachricht: Zeiger und Feld Initialisieren, wenn Feld bereit *)
0072                  NewData:=FALSE;
0073                  InputL:=0;
0074                  IBitzeiger:=0;
0075                  FOR i:=0 TO 19 DO
0076                      InputD[i]:=0;
0077                  END_FOR;
0078                  END_IF;
0079                  InputD[InputL]:=SHL(InputD[InputL],1);
0080                  IF In=2 THEN
0081                      InputD[InputL]:=InputD[InputL]+1;
0082                  END_IF;
0083                  IBitzeiger:=IBitzeiger+1;
0084                  IF IBitzeiger>7 THEN
0085                      InputL:=InputL+1;
0086                      IF InputL>19 THEN
0087                          InputL:=19;
0088                      END_IF;
0089                      IBitzeiger:=0;
0090                  END_IF;
0091                  Outalt:=Out;
0092                  Out:=1;
0093                  IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0094                  IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0095                  IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0096                  IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0097              ELSE
0098                  (* Aktion 3: Slave hat Idle gesendet: Abschluss einer Nachricht?; Master sendet Trenn-Nachricht *)
0099                  IF Lastin<>3 THEN
0100                      (* Abschluss einer Nachricht; wenn diese nicht vollständig (also ganzzahlige Anzahl von Bytes),
0101                      dann wird sie verworfen! *)
0102                      IF IBitzeiger =0 THEN
0103                          FOR i:=0 TO InputL DO
0104                              InputData[i]:=InputD[i];
0105                          END_FOR;
0106                          NewData:=TRUE;
0107                          InputLen:=InputL;
0108                          SCSlave:=TRUE;
0109                      END_IF;
0110                  END_IF;
0111                  Outalt:=Out;
0112                  Out:=1;
0113                  IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0114                  IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;

```



```

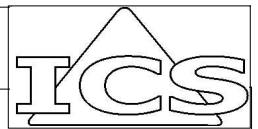
0115     IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0116     IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0117     END_IF;
0118     END_IF;
0119 ELSIF (Takt<>Taktalt AND Takt=TRUE) THEN
0120     (* Slave alle 120ms testen, ob er den seriellen Kommunikationsmodus unterstützt, dient auch zum Anlauf
0121     der Kommunikation *)
0122     Timeout:=Timeout+1;
0123     IF Timeout>4 THEN
0124         Outalt:=Out;
0125         IF (Out=0 OR Out=3) THEN
0126             Out:=1;
0127         ELSIF (Out=1 OR Out=2) THEN
0128             Out:=3;
0129         END_IF;
0130         Timeout:=0;
0131         SCSlave:=FALSE;
0132         SSW:=0;
0133         IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0134         IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0135         IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0136         IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0137     END_IF;
0138 END_IF;
0139
0140 (* und hier läuft die übergeordnete Steuerung des seriellen Slaves ab. Zyklus: 16s *)
0141 IF Taktalt<>Takt AND Takt=FALSE AND Acyclic<>0 THEN
0142     Timeout2:=Timeout2+1;
0143     IF Timeout2>151 THEN
0144         Timeout2:=0;
0145         Acyclic:=0;
0146         Diagnosis:=Diagnosis AND 16#EF;
0147     END_IF;
0148 END_IF;
0149
0150 IF SCSlave=FALSE THEN
0151     Diagnosis:=Diagnosis OR 16#80;
0152     ARS0Request:=FALSE;
0153     ARSResponse_0:=FALSE;
0154     ARS1Request:=FALSE;
0155     ARSResponse_1:=FALSE;
0156     AWS2Request:=FALSE;
0157     ID_Data[1]:=0;
0158 ELSE
0159     Diagnosis:=Diagnosis AND 16#7F;
0160 END_IF;
0161
0162 (* Kommandoaufrufe: ID des Slaves schon bekannt? *)
0163 IF ID_Data[1]=0 AND Acyclic=0 AND Send=FALSE AND SCSlave=TRUE THEN
0164     (* ID des Slaves abfragen *)
0165     OutputData[0]:=16;
0166     OutputData[1]:=0;
0167     OutputData[2]:=6;
0168     OutputLen:=3;
0169     Send:=TRUE;
0170     ARS0Request:=TRUE;
0171     Diagnosis:=Diagnosis OR 16#10;
0172     Acyclic:=100;
0173 END_IF;

```

```

0174
0175 (* Zyklisch das Diagnose-Byte abfragen *)
0176 IF ARSResponse_0=TRUE AND ARSResponse_1=FALSE AND Acyclic=0 AND Send=FALSE AND SCSlave=TRUE THEN
0177 (* Diagnose des Slaves abfragen *)
0178   OutputData[0]:=16;
0179   OutputData[1]:=1;
0180   OutputData[2]:=1;
0181   OutputLen:=3;
0182   Send:=TRUE;
0183   ARS1Request:=TRUE;
0184   Diagnosis:=Diagnosis OR 16#10;
0185   Acyclic:=101;
0186 END_IF;
0187
0188 (* Parameter des Slaves schon bekannt? *)
0189 IF ARSResponse_2=FALSE AND Acyclic=0 AND Send=FALSE AND SCSlave=TRUE THEN
0190   OutputData[0]:=16;
0191   OutputData[1]:=2;
0192   OutputData[2]:=6;
0193   OutputLen:=3;
0194   Send:=TRUE;
0195   ARS2Request:=TRUE;
0196   Acyclic:=102;
0197   Diagnosis:=Diagnosis OR 16#10;
0198 END_IF;
0199
0200 (* Wenn Veränderung, dann Parameter schreiben *)
0201 IF ARSResponse_2=TRUE AND SCSlave=TRUE AND Acyclic=0 AND Send=FALSE THEN
0202   MerkPara:=FALSE;
0203   FOR i:=1 TO 6 DO
0204     IF Para_DatIn[i]<>Para_Data[i] THEN
0205       MerkPara:=TRUE;
0206     END_IF;
0207   END_FOR;
0208   IF MerkPara=TRUE THEN
0209     OutputData[0]:=17;
0210     OutputData[1]:=2;
0211     OutputData[2]:=6;
0212     FOR i:=1 TO 6 DO
0213       OutputData[i+2]:=Para_DatIn[i];
0214     END_FOR;
0215     OutputLen:=9;
0216     AWS2Request:=TRUE;
0217     ARSResponse_2:=FALSE;
0218     Send:=TRUE;
0219     Acyclic:=103;
0220   END_IF;
0221 END_IF;
0222
0223 IF Newdata=TRUE THEN
0224   (* ID-Daten sind angekommen *)
0225   IF InputData[0]=80 AND Acyclic=100 THEN
0226     FOR i:=1 TO 6 DO
0227       ID_Data[i]:=InputData[i+1];
0228     END_FOR;
0229     NewData:=FALSE;
0230     ARSResponse_0:=TRUE;
0231     Timeout2:=0;
0232     Acyclic:=0;

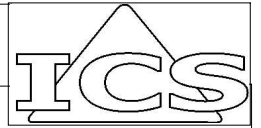
```



```

0233 IF ID_Data[1]<>1 OR ID_Data[3]<>4 THEN (* richtiger Slave? *)
0234     Diagnosis:=Diagnosis OR 16#40;
0235 ELSE
0236     Diagnosis:=Diagnosis AND 16#BF;
0237 END_IF;
0238     Diagnosis:=Diagnosis AND 16#EF;
0239 END_IF;
0240 (* Diagnosedaten sind angekommen *)
0241 IF InputData[0]=80 AND Acyclic=101 THEN
0242     InputData[1]:=InputData[1] AND 16#0F;
0243     Diagnosis:=Diagnosis AND 16#F0;
0244     Diagnosis:=Diagnosis OR InputData[1];
0245     NewData:=FALSE;
0246     ARSResponse_1:=TRUE;
0247     Acyclic:=0;
0248     Diagnosis:=Diagnosis AND 16#EF;
0249 END_IF;
0250 (* Parameter-Istdaten sind angekommen *)
0251 IF InputData[0]=80 AND Acyclic=102 THEN
0252     FOR i:=1 TO 7 DO
0253         Para_DatOut[i]:=InputData[i];
0254         Para_Data[i]:=InputData[i];
0255     END_FOR;
0256     NewData:=FALSE;
0257     ARSResponse_2:=TRUE;
0258     Timeout2:=0;
0259     Diagnosis:=Diagnosis AND 16#EF;
0260     Acyclic:=0;
0261 END_IF;
0262 (* Acknowledge for Parameter Schreiben ist angekommen *)
0263 IF InputData[0]=81 AND Acyclic=103 THEN
0264     NewData:=FALSE;
0265     AWSResponse_2:=TRUE;
0266     Timeout2:=0;
0267     Acyclic:=0;
0268     Diagnosis:=Diagnosis AND 16#EF;
0269 END_IF;
0270 (* Messdaten sind angekommen *)
0271 IF InputData[0]=0 AND SCSSlave=TRUE THEN
0272     InputWord:=256*InputData[1]+InputData[2];
0273     NewData:=FALSE;
0274 END_IF;
0275 (* Aufruf nicht ok *)
0276 IF InputData[0]=144 OR InputData[0]=145 THEN
0277     InputData[1]:=InputData[1] AND 16#0F;
0278     Diagnosis:=Diagnosis AND 16#F0;
0279     Diagnosis:=Diagnosis OR InputData[1];
0280     NewData:=FALSE;
0281     Timeout2:=0;
0282     Diagnosis:=Diagnosis AND 16#EF;
0283     Acyclic:=0;
0284 END_IF;
0285 (* irgendein Fehler ist eingetroffen *)
0286 IF (InputData[0]=81 OR InputData[0]=0) AND Acyclic=0 THEN
0287     NewData:=FALSE;
0288     Diagnosis:=Diagnosis AND 16#EF;
0289 END_IF;
0290 END_IF;
0291

```



0292 Taktalt:=Takt;
0293 Inalt:=In;

Beispielprogramm

Projektinformationen
PLC_PRG (PRG-ST)
SerComTS (FB-ST)

A
1-1
2-1

Beispielprogramm