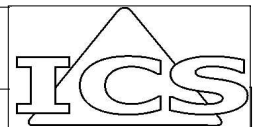


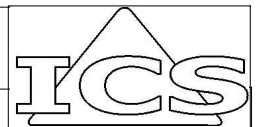
Dateiname: Example_SL.pro
Verzeichnis: D:\Andreas_Schiff\ICSGmbH\ICS\Internet\Internet-PDFs\Deutsch
Geändert am: 19.11.05 19:29:40 / V2.2
Bezeichnung: Example_SL
Autor: Andreas Schiff, ICS GmbH
Version: 1.3 vom 14.10.2004
Beschreibung: Das Programm steuert das AS-Interface
Anzeigemodul AM001 an und gibt im
Sekundentakt eine um 1 erhöhte Zahl aus.



```

0001 PROGRAM PLC_PRG
0002 (* ----- *)
0003 (*           Anzeige-Text-Beispielprogramm für AM001           *)
0004 (* ----- *)
0005 (* ICS GmbH, Hopfenstraße 1, 88069 Tettngang, Autor: Andreas Schiff           Version 1.3 *)
0006 (* ----- *)
0007 (* Dieses Beispielprogramm dient nur zur Demonstration der Funktion des AM001. Es kann keine Gewähr dafür *)
0008 (* übernommen werden, dass dieses Programm in einem realen Automatisierungsprojekt fehlerfrei läuft. *)
0009 (* ----- *)
0010
0011 VAR
0012     AnzeigeB: SerComSL; (* Funktionsbaustein für Text-Wandlung und Ausgabe *)
0013     Taktgeber3: TON;
0014     Takt2: BOOL;
0015     HB2: BOOL;
0016     Taktgeber4: TON;
0017     Takt1: BOOL;
0018     Zaehler: INT; (* Zahl, die angezeigt wird *)
0019     Plus: BYTE:=0; (* Pluszeichen soll ausgegeben werden *)
0020     Text: ARRAY [0..7] OF BYTE:=16#7E, 16#26, 16#26, 16#26, 16#26, 16#26, 16#26, 16#7F; (* Textformatierung
0021     für unterlagerten Text: Die Zeichen &&&&& werden von der Zahl
0022     überschrieben *)
0023     Diagnose: BYTE;
0024     DI2 AT %IX1.5.2: BOOL; (* Im Beispiel hat das Anzeigemodul die Adresse 5 *)
0025     DI3 AT %IX1.5.3: BOOL;
0026     DO0 AT %QX1.5.0: BOOL;
0027     DO1 AT %QX1.5.1: BOOL;
0028 END_VAR
0029
0030 (* Ende des Deklarationsteils *)
-----
0001
0002 (* ----- Hauptprogramm ----- *)
0003 (* In diesem Programm wird alle 1s die Variable "Zähler" um 1 erhöht. Die Variable wird anschließend auf
0004 dem einzeiligen Textdisplay ausgegeben. Daraus kann die Zeit, die für die Aktualisierung der Anzeige benötigt wird,
0005 abgeschätzt werden. *)
0006
0007 (* Takt: 1000ms *)
0008 Taktgeber3(IN:=Takt2,PT:=t#100ms);
0009 HB2:=NOT Taktgeber3.Q;
0010 Taktgeber4(IN:=HB2,PT:=t#900ms);
0011 Takt2:=Taktgeber4.Q;
0012
0013 AnzeigeB(DI2:=DI2, DI3:=DI3, Variable:=Zaehler, Format:=Plus,Text:=Text);
0014 Diagnose:=AnzeigeB.Diagnosis;
0015 DO0:=AnzeigeB.DO0;
0016 DO1:=AnzeigeB.DO1;
0017
0018 IF Takt1<>Takt2 AND Takt2=TRUE THEN
0019     Zaehler:=Zaehler+1;
0020 END_IF;
0021
0022 Takt1:=Takt2;

```

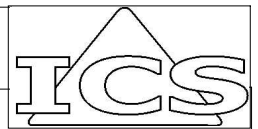


0001	FUNCTION_BLOCK SerComSL		
0002	(* ----- *)		
0003	(* Dieser Funktionsbaustein wickelt die gesamte serielle Kommunikation zu einem AS-Interface Slave nach dem *)		
0004	(* Profil S-B.A.5 ab. Gilt nur für einzeilige Ausgabe. Der Funktionsbaustein schreibt eine Zeile ASCII-Text im *)		
0005	(* zyklischen Mode und trägt die übergebene Zahl in den Text ein, wenn sich die Zahl ändert *)		
0006	(* ----- *)		
0007	(* ICS GmbH, Hopfenstraße 1, 88069 Tettnang, Autor: Andreas Schiff		Version 0.4 Datum: 14.10.2004 *)
0008	(* verbesserter Abschluss eines gesendeten Telegramms *)		
0009	(* ----- *)		
0010			
0011	VAR_INPUT		
0012	Variable:	INT;	(* auszugebene Zahl nach Maßgabe der Formatierung *)
0013	Format:	BYTE;	(* Formatierung der Variablen *)
0014	(* =0: Ganze Zahl im Bereich von -32767 bis 32767 *)		
0015	(* =1: wie vor, Pluszeichen und führende Nullen werden zusätzlich		
0016	ausgegeben: -32767 bis +32767 *)		
0017	Text:	ARRAY [0..7] OF BYTE:=16#7E,16#26,16#26,16#26,16#26,16#26,16#26,16#7F;	
0018	DI2:	BOOL;	(* Eingang serielles Protokoll *)
0019	DI3:	BOOL;	(* Eingang serielles Protokoll *)
0020	END_VAR		
0021			
0022	VAR_OUTPUT		
0023	Diagnosis:	BYTE;	(* Diagnose-Daten: Bit 7: Slave nicht im seriellen Modus
0024	Bit 6: ID nicht ok (Soll: Vendor-ID: 1, Product-ID: 5)		
0025	Bit 5: --		
0026	Bit 4 Slave busy		
0027	Bit 0..3: Diagnoseinfo vom Slave *)		
0028	DO0:	BOOL;	(* Ausgang serielles Protokoll *)
0029	DO1:	BOOL;	(* Ausgang serielles Protokoll *)
0030	END_VAR		
0031			
0032	VAR		
0033	DatIn:	ARRAY [0..7] OF BYTE; (* Zyklischen Daten *)	
0034	ID_Data:	ARRAY [0..8] OF BYTE; (* ID-Daten des Slaves *)	
0035	InputD:	ARRAY [0..8] OF BYTE;	
0036	InputData:	ARRAY [0..20] OF BYTE; (* Eingabe-Datenfeld *)	
0037	OutputData:	ARRAY [0..20] OF BYTE; (* Ausgabe-Datenfeld *)	
0038	OutputLen:	BYTE;	(* Länge der Ausgabedaten in Byte; Zahlenbereich 1..15 *)
0039	InputL:	BYTE;	(* Länge der Eingabedaten in Byte; Zahlenbereich 1..15 *)
0040	InputLen:	BYTE;	
0041	Taktalt:	BOOL;	
0042	In:	BYTE;	(* aktuell empfangene Nachricht *)
0043	Inalt:	BYTE;	(* letzte empfangene Nachricht *)
0044	Out:	BYTE:= 3;	
0045	Watchdog:	BOOL;	
0046	Timeout:	BYTE;	
0047	Lastin:	BYTE;	(* letztes gültiges empfangenes Informationsbit *)
0048	i:	BYTE;	
0049	IBitzeiger:	BYTE;	
0050	Outalt:	BYTE;	
0051	Outzeiger:	BYTE;	
0052	OBitzeiger:	BYTE;	
0053	x:	BYTE;	
0054	Send:	BOOL;	
0055	y:	BYTE;	
0056	NewData:	BOOL;	
0057	SCSlave:	BOOL;	
0058	SSW:	WORD;	
0059	Inst1:	TON;	

```

0060 Inst2:TON;
0061 Takt:BOOL;
0062 Var2: BOOL;
0063 Acyclic: WORD;
0064 Para_DatIn: ARRAY [0..8] OF BYTE;
0065 Timeout2: BYTE;
0066 MerkDat1: BOOL;
0067 MerkDat2: BOOL;
0068 Schluss: BOOL;
0069 ARS0Request: BOOL;
0070 ARSResponse_0: BOOL;
0071 ARS1Request: BOOL;
0072 ARSResponse_1: BOOL;
0073 ARS2Request: BOOL;
0074 ARSResponse_2: BOOL;
0075 AWS2Request: BOOL;
0076 AWSResponse_2: BOOL;
0077 Varalt: INT;
0078 Zahl1: INT;
0079 Zahl2: INT;
0080 Zahl3: INT;
0081 Zahl4: INT;
0082 Zeichen: ARRAY[0..5] OF BYTE;
0083 j: BYTE;
0084 k2: BYTE;
0085 k1: BYTE;
0086 k3: BYTE;
0087 k4: BYTE;
0088 Zahl5: INT;
0089 Vorzeichen: BYTE;
0090 NeuDat: BOOL;
0091 END_VAR
0092
0093 (* -----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 (* ----- Hier wird die auszugebende Zahl in das richtige Format gebracht ----- *)
0007
0008 IF Variable<>Varalt THEN
0009   (* Ausgabefeld suchen und Länge bestimmen *)
0010   NeuDat:=TRUE;
0011   k1:=0;
0012   k2:=0;
0013   FOR i:=0 TO 7 DO
0014     DatIn[i]:=Text[i];
0015     IF Text[i]=16#26 AND k1=0 THEN
0016       IF k1=0 THEN k1:=i; END_IF;           (* Position Feldanfang *)
0017       k2:=1;                                (* Anzahl der Stellen *)
0018       FOR j:=1 TO 5 DO
0019         IF i+j<8 THEN
0020           IF Text[i+j]=16#26 THEN
0021             k2:=k2+1;
0022           END_IF;
0023         END_IF;
0024       END_FOR;
0025     END_IF;

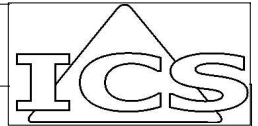
```



```

0026 END_FOR;
0027 k3:=k1;
0028 k4:=k2;
0029 IF k2>6 THEN k2:=6; END_IF;
0030 FOR i:=0 TO k2 DO
0031     Zeichen[i]:=32;
0032 END_FOR;
0033
0034 IF Variable>=0 AND (k2=6 OR ((k2=5 AND Format=1 AND Variable<10000) OR (k2=5 AND Format=0)) OR
0035 ((k2=4 AND Format=0 AND Variable<10000) OR (k2=4 AND Format=1 AND Variable<1000)) OR ((k2=3
0036 AND Format=0 AND Variable<1000) OR (k2=3 AND Format=1 AND Variable<100)) OR ((k2=2 AND
0037 Format=0 AND Variable<100) OR (k2=2 AND Format=1 AND Variable<10))) THEN
0038 (* positiver Zahlenbereich zwischen 0 und +32767 *)
0039     Zahl1:=Variable/10000;
0040     Zahl2:=Variable/1000-Zahl1*10;
0041     Zahl3:=Variable/100-Zahl1*100-Zahl2*10;
0042     Zahl4:=Variable/10-Zahl1*1000-Zahl2*100-Zahl3*10;
0043     Zahl5:=Variable-Zahl1*10000-Zahl2*1000-Zahl3*100-Zahl4*10;
0044     Vorzeichen:=16#2B;
0045     IF Variable<10 THEN
0046         Zeichen[5]:=INT_TO_BYTE(Zahl5)+48;
0047         IF Format=1 THEN Zeichen[4]:=Vorzeichen; END_IF;
0048     ELSIF Variable<100 THEN
0049         Zeichen[4]:=INT_TO_BYTE(Zahl4)+48;
0050         Zeichen[5]:=INT_TO_BYTE(Zahl5)+48;
0051         IF Format=1 THEN Zeichen[3]:=Vorzeichen; END_IF;
0052     ELSIF Variable<1000 THEN
0053         Zeichen[3]:=INT_TO_BYTE(Zahl3)+48;
0054         Zeichen[4]:=INT_TO_BYTE(Zahl4)+48;
0055         Zeichen[5]:=INT_TO_BYTE(Zahl5)+48;
0056         IF Format=1 THEN Zeichen[2]:=Vorzeichen; END_IF;
0057     ELSIF Variable<10000 THEN
0058         Zeichen[2]:=INT_TO_BYTE(Zahl2)+48;
0059         Zeichen[3]:=INT_TO_BYTE(Zahl3)+48;
0060         Zeichen[4]:=INT_TO_BYTE(Zahl4)+48;
0061         Zeichen[5]:=INT_TO_BYTE(Zahl5)+48;
0062         IF Format=1 THEN Zeichen[1]:=Vorzeichen; END_IF;
0063     ELSE
0064         Zeichen[1]:=INT_TO_BYTE(Zahl1)+48;
0065         Zeichen[2]:=INT_TO_BYTE(Zahl2)+48;
0066         Zeichen[3]:=INT_TO_BYTE(Zahl3)+48;
0067         Zeichen[4]:=INT_TO_BYTE(Zahl4)+48;
0068         Zeichen[5]:=INT_TO_BYTE(Zahl5)+48;
0069         IF Format=1 THEN Zeichen[0]:=Vorzeichen; END_IF;
0070     END_IF;
0071     ELSIF (k2=6 AND Variable>-32768 AND Variable<0) OR (k2=5 AND Variable>-10000 AND Variable<0) OR
0072 (k2=4 AND Variable>-1000 AND Variable<0) OR (k2=3 AND Variable>-100 AND Variable<0) OR (k2=2
0073 AND Variable>-10 AND Variable<0) THEN
0074 (* negativer Zahlenbereich zwischen -32768 und -1 *)
0075     Variable:=-Variable;
0076     Zahl1:=Variable/10000;
0077     Zahl2:=Variable/1000-Zahl1*10;
0078     Zahl3:=Variable/100-Zahl1*100-Zahl2*10;
0079     Zahl4:=Variable/10-Zahl1*1000-Zahl2*100-Zahl3*10;
0080     Zahl5:=Variable-Zahl1*10000-Zahl2*1000-Zahl3*100-Zahl4*10;
0081     Vorzeichen:=16#2D;
0082     IF Variable<10 THEN
0083         Zeichen[5]:=INT_TO_BYTE(Zahl5)+48;
0084         Zeichen[4]:=Vorzeichen;

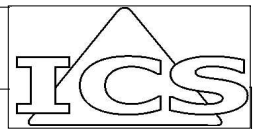
```



```

0085 ELSIF Variable<100 THEN
0086     Zeichen[4]:=INT_TO_BYTE(Zahl4)+48;
0087     Zeichen[5]:=INT_TO_BYTE(Zahl5)+48;
0088     Zeichen[3]:=Vorzeichen;
0089 ELSIF Variable<1000 THEN
0090     Zeichen[3]:=INT_TO_BYTE(Zahl3)+48;
0091     Zeichen[4]:=INT_TO_BYTE(Zahl4)+48;
0092     Zeichen[5]:=INT_TO_BYTE(Zahl5)+48;
0093     Zeichen[2]:=Vorzeichen;
0094 ELSIF Variable<10000 THEN
0095     Zeichen[2]:=INT_TO_BYTE(Zahl2)+48;
0096     Zeichen[3]:=INT_TO_BYTE(Zahl3)+48;
0097     Zeichen[4]:=INT_TO_BYTE(Zahl4)+48;
0098     Zeichen[5]:=INT_TO_BYTE(Zahl5)+48;
0099     Zeichen[1]:=Vorzeichen;
0100 ELSE
0101     Zeichen[1]:=INT_TO_BYTE(Zahl1)+48;
0102     Zeichen[2]:=INT_TO_BYTE(Zahl2)+48;
0103     Zeichen[3]:=INT_TO_BYTE(Zahl3)+48;
0104     Zeichen[4]:=INT_TO_BYTE(Zahl4)+48;
0105     Zeichen[5]:=INT_TO_BYTE(Zahl5)+48;
0106     Zeichen[0]:=Vorzeichen;
0107 END_IF;
0108 ELSE
0109     Zeichen[0]:=16#23;
0110     Zeichen[1]:=16#23;
0111     Zeichen[2]:=16#23;
0112     Zeichen[3]:=16#23;
0113     Zeichen[4]:=16#23;
0114     Zeichen[5]:=16#23;
0115 END_IF;
0116
0117 (* Eintragen der Zeichen anstelle der &&&&& *)
0118 IF k2=6 THEN   DatIn[k1]:=Zeichen[0]; k1:=k1+1;   k2:=k2-1;   END_IF;
0119 IF k2=5 THEN   DatIn[k1]:=Zeichen[1]; k1:=k1+1;   k2:=k2-1;   END_IF;
0120 IF k2=4 THEN   DatIn[k1]:=Zeichen[2]; k1:=k1+1;   k2:=k2-1;   END_IF;
0121 IF k2=3 THEN   DatIn[k1]:=Zeichen[3]; k1:=k1+1;   k2:=k2-1;   END_IF;
0122 IF k2=2 THEN   DatIn[k1]:=Zeichen[4]; k1:=k1+1;   k2:=k2-1;   END_IF;
0123 IF k2=1 THEN   DatIn[k1]:=Zeichen[5]; k1:=k1+1;   k2:=k2-1;   END_IF;
0124 END_IF;
0125 (* ----- Hier wird das serielle Protokoll bedient ----- *)
0126 (* Eingänge des Slaves lesen, ausmaskieren und auf Veränderung prüfen *)
0127 (* Hier wird umcodiert: In=3: Idle; In=1: Separator; In=0: DATA0; In=2: DATA1 *)
0128 IF DI2=FALSE AND DI3=FALSE THEN In:=0; END_IF;
0129 IF DI2=FALSE AND DI3=TRUE THEN In:=2; END_IF;
0130 IF DI2=TRUE AND DI3=FALSE THEN In:=1; END_IF;
0131 IF DI2=TRUE AND DI3=TRUE THEN In:=3; END_IF;
0132
0133 IF In<>Inalt THEN
0134     (* eine Antwort ist eingetroffen *)
0135     IF Out=1 AND In=1 THEN
0136         (* die eine der erwarteten Antworten ist eingetroffen: Separator *)
0137         Watchdog:=FALSE;
0138         SCSlave:=TRUE;
0139         Timeout:=0;
0140         Lastin:=Inalt;
0141         (* Aktion 2: Trenn-Nachricht empfangen; Master sendet Information oder Idle *)
0142         IF Send=TRUE THEN
0143             IF Outalt=3 THEN

```



```

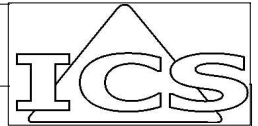
0144      (* Ausgabe initialisieren *)
0145      Outzeiger:=0;
0146      OBitzeiger:=0;
0147      y:=OutputData[0];
0148      END_IF;
0149      IF Schluss=FALSE THEN
0150          Outalt:=Out;
0151          x:=y AND 2#1000_0000;
0152          IF x<>0 THEN
0153              Out:=2;
0154          ELSE
0155              Out:=0;
0156          END_IF;
0157          y:=SHL(y,1);
0158          OBitzeiger:=OBitzeiger+1;
0159      (* ein Byte fertig? Zeiger auf nächstes Byte stellen *)
0160      IF OBitzeiger>7 THEN
0161          Outzeiger:=Outzeiger+1;
0162          y:=OutputData[Outzeiger];
0163          OBitzeiger:=0;
0164          IF Outzeiger>=OutputLen OR Outzeiger>=19 THEN
0165              Schluss:=TRUE;
0166              SCSlave:=TRUE;
0167          END_IF;
0168      END_IF;
0169      ELSE
0170          Outalt:=Out;
0171          Out:=3;
0172          Send:=FALSE;
0173          Schluss:=FALSE;
0174      END_IF;
0175      ELSE
0176          Outalt:=Out;
0177          Out:=3;
0178      END_IF;
0179      IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0180      IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0181      IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0182      IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0183      ELSIF Out<>1 AND In<>1 THEN
0184      (* die andere der erwarteten Antworten ist eingetroffen: Idle oder Daten *)
0185      Watchdog:=FALSE;
0186      SCSlave:=TRUE;
0187      Timeout:=0;
0188      IF (In=0 AND Inalt<>2) OR (In=2 AND Inalt<>0) THEN
0189      (* Aktion 0/1: Slave hat Informationsbit gesendet: abspeichern; Master sendet Trenn-Nachricht *)
0190      IF Lastin=3 THEN
0191      (* Beginn einer Nachricht: Zeiger und Feld Initialisieren, wenn Feld bereit *)
0192      NewData:=FALSE;
0193      InputL:=0;
0194      IBitzeiger:=0;
0195      FOR i:=0 TO 19 DO
0196      InputD[i]:=0;
0197      END_FOR;
0198      END_IF;
0199      InputD[InputL]:=SHL(InputD[InputL],1);
0200      IF In=2 THEN
0201      InputD[InputL]:=InputD[InputL]+1;
0202      END_IF;

```

```

0203     IBitzeiger:=IBitzeiger+1;
0204     IF IBitzeiger>7 THEN
0205         InputL:=InputL+1;
0206         IF InputL>19 THEN
0207             InputL:=19;
0208         END_IF;
0209         IBitzeiger:=0;
0210     END_IF;
0211     Outalt:=Out;
0212     Out:=1;
0213     IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0214     IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0215     IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0216     IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0217 ELSE
0218     (* Aktion 3: Slave hat Idle gesendet: Abschluss einer Nachricht?; Master sendet Trenn-Nachricht *)
0219     IF Lastin<>3 THEN
0220         (* Abschluss einer Nachricht; wenn diese nicht vollständig (also ganzzahlige Anzahl von Bytes),
0221         dann wird sie verworfen! *)
0222         IF IBitzeiger =0 THEN
0223             FOR i:=0 TO InputL DO
0224                 InputData[i]:=InputD[i];
0225             END_FOR;
0226             NewData:=TRUE;
0227             InputLen:=InputL;
0228             SCSlave:=TRUE;
0229         END_IF;
0230     END_IF;
0231     Outalt:=Out;
0232     Out:=1;
0233     IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0234     IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0235     IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0236     IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0237 END_IF;
0238 END_IF;
0239 ELSIF (Takt<>Taktalt AND Takt=TRUE) THEN
0240     (* Slave alle 120ms testen, ob er den seriellen Kommunikationsmodus unterstützt, dient auch zum Anlauf
0241     der Kommunikation *)
0242     Timeout:=Timeout+1;
0243     IF Timeout>4 THEN
0244         Outalt:=Out;
0245         IF (Out=0 OR Out=3) THEN
0246             Out:=1;
0247         ELSIF (Out=1 OR Out=2) THEN
0248             Out:=3;
0249         END_IF;
0250         Timeout:=0;
0251         SCSlave:=FALSE;
0252         SSW:=0;
0253         Send:=FALSE;
0254         IF Out=0 THEN DO0:=TRUE; DO1:=TRUE; END_IF;
0255         IF Out=1 THEN DO0:=FALSE; DO1:=TRUE; END_IF;
0256         IF Out=2 THEN DO0:=TRUE; DO1:=FALSE; END_IF;
0257         IF Out=3 THEN DO0:=FALSE; DO1:=FALSE; END_IF;
0258     END_IF;
0259 END_IF;
0260
0261 (* Hier läuft die Timeoutüberwachung für azyklische Dienste ab: 1s *)

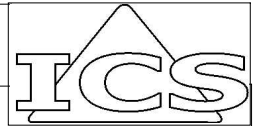
```



```

0262 IF Taktalt<>Takt AND Takt=FALSE AND Acyclic<>0 THEN
0263     Timeout2:=Timeout2+1;
0264     IF Timeout2>151 THEN
0265         Acyclic:=0;
0266         Timeout2:=0;
0267         MerkDat1:=FALSE;
0268         MerkDat2:=FALSE;
0269         Diagnosis:=Diagnosis AND 16#EF;
0270     END_IF;
0271 END_IF;
0272
0273 IF SCSSlave=FALSE THEN
0274     Diagnosis:=Diagnosis OR 16#80;
0275     ARS0Request:=FALSE;
0276     ARSResponse_0:=FALSE;
0277     ARS1Request:=FALSE;
0278     ARSResponse_1:=FALSE;
0279     AWS2Request:=FALSE;
0280     ID_Data[1]:=0;
0281 ELSE
0282     Diagnosis:=Diagnosis AND 16#7F;
0283 END_IF;
0284
0285 (* Kommandoaufrufe: ID des Slaves schon bekannt? *)
0286 IF ID_Data[1]=0 AND Acyclic=0 AND Send=FALSE AND SCSSlave=TRUE THEN
0287 (* ID des Slaves abfragen *)
0288     OutputData[0]:=16;
0289     OutputData[1]:=0;
0290     OutputData[2]:=6;
0291     OutputLen:=3;
0292     Send:=TRUE;
0293     ARS0Request:=TRUE;
0294     Acyclic:=100;
0295     Diagnosis:=Diagnosis OR 16#10;
0296 END_IF;
0297
0298 (* Kommandoaufrufe: Diagnose vom Slave zyklisch abrufen *)
0299 IF ARSResponse_0=TRUE AND ARSResponse_1=FALSE AND Acyclic=0 AND Send=FALSE AND
0300     SCSSlave=TRUE THEN
0301 (* Diagnose des Slaves abfragen *)
0302     OutputData[0]:=16;
0303     OutputData[1]:=1;
0304     OutputData[2]:=1;
0305     OutputLen:=3;
0306     Send:=TRUE;
0307     ARS1Request:=TRUE;
0308     Acyclic:=101;
0309     Diagnosis:=Diagnosis OR 16#10;
0310 END_IF;
0311
0312 (* Kommandoaufrufe: Parameter des Slaves schon bekannt? *)
0313 IF AWS2Request=FALSE AND Acyclic=0 AND Send=FALSE AND SCSSlave=TRUE THEN
0314 (* Parameter an Slave senden *)
0315     OutputData[0]:=17;
0316     OutputData[1]:=2;
0317     OutputData[2]:=8;
0318     FOR i:=0 TO 7 DO
0319         OutputData[i+3]:=DatIn[i];
0320     END_FOR;

```

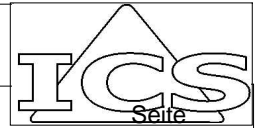


```

0321 OutputLen:=11;
0322 Send:=TRUE;
0323 AWS2Request:=TRUE;
0324 ARSResponse_2:=FALSE;
0325 Acyclic:=103;
0326 Diagnosis:=Diagnosis OR 16#10;
0327 END_IF;
0328
0329 (* Kommandoaufrufe: Parameter des Slaves schon bekannt? *)
0330 IF ARSResponse_2=FALSE AND Acyclic=0 AND Send=FALSE AND SCSlave=TRUE THEN
0331 (* Parameter des Slaves abfragen *)
0332   OutputData[0]:=16;
0333   OutputData[1]:=2;
0334   OutputData[2]:=8;
0335   OutputLen:=3;
0336   Send:=TRUE;
0337   ARS2Request:=TRUE;
0338   Acyclic:=102;
0339   Diagnosis:=Diagnosis OR 16#10;
0340 END_IF;
0341
0342 (* Wenn einzeliges Display, dann zyklische Ausgabe! *)
0343 IF Send=FALSE AND SCSlave=TRUE AND NeuDat=TRUE THEN
0344   OutputData[0]:=1;
0345   FOR i:=0 TO 7 DO
0346     OutputData[i+1]:=DatIn[i];
0347   END_FOR;
0348   OutputLen:=9;
0349   NeuDat:=FALSE;
0350   Send:=TRUE;
0351 END_IF;
0352
0353 IF Newdata=TRUE THEN
0354 (* ID-Daten sind angekommen *)
0355   IF InputData[0]=80 AND Acyclic=100 THEN
0356     FOR i:=0 TO 5 DO
0357       ID_Data[i]:=InputData[i+1];
0358     END_FOR;
0359     NewData:=FALSE;
0360     ARSResponse_0:=TRUE;
0361     Timeout2:=0;
0362     Acyclic:=0;
0363     IF ID_Data[1]<>1 THEN
0364       Diagnosis:=Diagnosis OR 16#40;
0365     ELSE
0366       Diagnosis:=Diagnosis AND 16#BF;
0367     END_IF;
0368     Diagnosis:=Diagnosis AND 16#EF;
0369   END_IF;
0370 (* Diagnosedaten sind angekommen *)
0371   IF InputData[0]=80 AND Acyclic=101 THEN
0372     InputData[1]:=InputData[1] AND 16#0F;
0373     Diagnosis:=Diagnosis AND 16#F0;
0374     Diagnosis:=Diagnosis OR InputData[1];
0375     NewData:=FALSE;
0376     ARSResponse_1:=TRUE;
0377     Timeout2:=0;
0378     Acyclic:=0;
0379     Diagnosis:=Diagnosis AND 16#EF;

```

```
0380 END_IF;
0381 (* Parameter-Istdaten sind angekommen *)
0382 IF InputData[0]=80 AND Acyclic=102 THEN
0383     NewData:=FALSE;
0384     ARSResponse_2:=TRUE;
0385     Timeout2:=0;
0386     Acyclic:=0;
0387     FOR i:=0 TO 7 DO
0388         Para_DatIn[i]:=InputData[i+1];
0389     END_FOR;
0390     Diagnosis:=Diagnosis AND 16#EF;
0391 END_IF;
0392 (* Acknowledge für Parameter Schreiben ist angekommen *)
0393 IF InputData[0]=81 AND Acyclic=103 THEN
0394     NewData:=FALSE;
0395     AWSResponse_2:=TRUE;
0396     Timeout2:=0;
0397     Acyclic:=0;
0398     Diagnosis:=Diagnosis AND 16#EF;
0399 END_IF;
0400
0401 (* Fehlerhafte Aufrufe*)
0402 IF InputData[0]=145 OR InputData[0]=144 THEN
0403     InputData[1]:=InputData[1] AND 16#0F;
0404     Diagnosis:=Diagnosis AND 16#F0;
0405     Diagnosis:=Diagnosis OR InputData[1];
0406     NewData:=FALSE;
0407     Timeout2:=0;
0408     Acyclic:=0;
0409     Diagnosis:=Diagnosis AND 16#EF;
0410 END_IF;
0411 (* irgendein Fehler ist angekommen *)
0412 IF (InputData[0]=81 OR InputData[0]=0) AND Acyclic=0 THEN
0413     NewData:=FALSE;
0414     Diagnosis:=Diagnosis AND 16#EF;
0415 END_IF;
0416 END_IF;
0417
0418 Taktalt:=Takt;
0419 Inalt:=In;
0420 Varalt:=Variable;
```



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

A
1-1
2-1

Beispielprogramm