Handbuch Integrierte Systemfunktionen Serie xx7

Dokument 26/797; Ausgabe DE05 | 2013-12-19

0

0 Inhalt

0.1 Dokumentversionen

Version	Datum	Geändert	Anmerkungen
DE01	2004-04-01	-	Publizierte Ausgabe
DE02	2004-08-12	2004-08-15	Änd. in Kap. 8
DE03	2005-11-10	2005-12-01	Flash Speicher, Neue CPUs
DE04	2006-03-02	2006-03-15	Generelle Änderungen
DE05	2013-12-19	-	Neues Logo und neuer Firmanname

0.2 Handelsmarken und Warenzeichen

Saia PCD® und Saia PG5®
sind registrierte Warenzeichen der Saia-Burgess Controls AG.

Step7® ,SIMATIC®, S7-300®, S7-400®, and Siemens® are registered trademarks of Siemens AG

Technische Veränderungen basieren auf dem aktuellen technischen Stand.

Publiziert in der Schweiz

1 Einleitung

Die Steuerungen der Saia PCD® Serie xx7 sind mit zahlreichen Systemfunktionen ausgestattet. Diese Systemfunktionen unterteilen sich in 3 Bereiche:

- Standard-Systemfunktionen (Bausteinnummer < 200)
- Erweiterte-Systemfunktionen (Bausteinnummer ≥ 200)
- Konfigurationsdatenblock (CDB)

Die Standard-Systemfunktionen sind die System Function Blocks (SFB) und System Function Calls (SFC), deren Funktionen äquivalent zu den gleichnamigen SIMATIC®-S7-Steuerungen von Siemens® sind.

Die erweiterten Systemfunktionen sind Saia Burgess Controls (SBC) spezifische SFBs und SFCs und dienen zur Programmierung der Onboard-Funktionen.

Die SBC spezifischen SFBs und SFCs, die eine umfangreiche Funktionalität aufweisen (z.B. serielle Kommunikation), werden hier nur der Vollständigkeit halber erwähnt und ausführlicher in separaten Kapiteln, bzw. Handbüchern behandelt.

Der Konfigurationsdatenblock ist ein Datenbaustein, der beim Aufstarten der CPU einmalig durchlaufen wird. Mit ihm können diverse Einstellungen der Schnittstellen, der Speicheraufteilung und der Systemfunktionen vorgenommen werden.

1.1 Erweiterte-Systemfunktionen

Sämtliche erweiterten Sytemfunktionen befinden sich im Betriebssystem. Sie können aus der CPU in das Offline-Projekt kopiert werden. Alternativ können die Funktionen auch aus der SBC Bibliothek kopiert werden. Die SBC Bibliothek kann kostenlos unter http://www.sbc-support.com heruntergeladen werden.

Nicht alle erweiterten Systemfunktionen sind in der SBC Bibliothek enthalten. Bei Bedarf können diese Funktionen aus dem Betriebssystem der jeweiligen CPU in das Offline-Projekt kopiert werden. SFB sind generell nicht in der SBC Bibliothek enthalten.

Alle Systemfunktionen setzen das BIE Bit im Statuswort auf 0, wenn die Funktion mit einem Fehler beendet wurde. Zusätzlich geben einige Systemfunktionen als Rückgabewert einen weiteren Fehlercode aus. Bei fehlerfreier Ausführung wird das BIE-Bit auf 1 gesetzt.

2 Standard-Systemfunktionen

Sämtliche unterstützten Standard-Sytemfunktionen befinden sich im Betriebssystem. Sie können alternativ aus der CPU in das Offline-Projekt kopiert, oder aus der entsprechenden Step®7-Bibliothek kopiert werden.

Es werden nicht alle Systemfunktionen von Siemens® unterstützt.

Die unterstützten Standard-Systemfunktionen sind in der folgender Auflistung aufgeführt. Eine ausführliche Beschreibung der Bausteine ist in der Hilfefunktion des SIMATIC®-Managers hinterlegt.

2.1 IEC-Timer und IEC-Counter

SFB Nr.	Name	Bezeichnung	Bemerkung
0	CTU	Vorwärtszähler	
1	CTD	Rückwärtszähler	
2	CTUD	Vorwärts-/Rückwärtszähler	
3	TP	Impuls	
4	TON	Einschaltverzögerung	
5	TOF	Ausschaltverzögerung	

2.2 Kommunikation über projektierte Verbindungen

SFB Nr.	Name	Bezeichnung	Bemerkung
12	BSEND	Blockorientiertes Senden	
13	BRCV	Blockorientiertes Empfangen	
14	GET	Daten vom Partner lesen	

Eine Beschreibung der Bausteine SFB12 - SFB14 ist im Handbuch 26/794 Serielle Kommunikation zu finden.

2.3 Systemdiagnose

SFC Nr.	Name	Bezeichnung	Bemerkung
6	RD_SINFO	Startinformation lesen	
52	WR_USMSG	Eintrag in den Diagnosepuffer	

2.4 CPU-Uhr und Betriebsstundenzähler

SFC Nr.	Name	Bezeichnung	Bemerkung
0	SET_CLK	Uhrzeit stellen	
1	READ_CKL	Uhrzeit lesen	
2	SET_RTM	Betriebsstundenzähler setzen	
3	CTRL_RTM	Betriebsstundenzähler steuern	
4	READ_RTM	Betriebsstundenzähler lesen	
64	TIME_TCK	Systemzeit lesen	

2.5 Datenbaustein-Manipulationen

SFC Nr.	Name	Bezeichnung	Bemerkung
20	BLKMOV	Datenbereich kopieren	
21	FILL	Datenbereich vorbesetzen	
22	CREAT_DB	Datenbaustein erzeugen	
23	DEL_DB	Datenbaustein löschen	
24	TEST_DB	Datenbaustein testen	
25	COMPRESS	Speicher komprimieren	
44	REPL_VAL	Ersatzwert eintragen	

2.6 Dezentrale Peripherie (PROFIBUS-DP)

SFC Nr.	Name	Bezeichnung	Bemerkung
7	DP_PRAL	Prozessalarm beim DP-Master auslösen	Nur bei PCD2.M487
11	DPSYN_FR	SYNC/FREEZE	
13	DPNRM_DG	Diagnosedaten lesen	
14	DPRD_DAT	Slave-Daten lesen	
15	DPWR_DAT	Slave-Daten schreiben	

2.7 Programmkontrolle

SFC Nr.	Name	Bezeichnung	Bemerkung
43	RE_TRIGR	Zykluszeit-Überw. nachtriggern	
46	STP	in den Zustand STOP wechseln	
47	WAIT	Verzögern der Bearbeitung des Anwenderprogramms	Nur bei PCD2.M487 und PCD3

2.8 Prozessabbild-Aktualisierung

SFC Nr.	Name	Bezeichnung	Bemerkung
26	UPDAT_PI	Prozessabbild der Eingänge aktualisieren	
27	UPDAT_PO	Prozessabbild der Ausgänge aktualisieren	

2.9 Unterbrechungsereignisse

SFC Nr.	Name	Bezeichnung	Bemerkung
28	SET_TINT	Uhrzeitalarm stellen	
29	CAN_TINT	Uhrzeitalarm stornieren	
30	ACT_TINT	Uhrzeitalarm aktivieren	
31	QRY_TINT	Uhrzeitalarm abfragen	
32	SRT_DINT	Verzögerungsalarm starten	
33	CAN_DINT	Verzögerungsalarm stornieren	
34	QRY_DINT	Verzögerungsalarm abfragen	
36	MSK_FLT	Synchronfehler maskieren	
37	DMSK_FLT	Synchronfehler demaskieren	

38	READ_ERR	Ereignisstatusregister lesen
39	DIS_IRT	Asynchronfehler sperren
40	EN_IRT	Asynchronfehler (neu) freigeben
41	DIS_AIRT	Asynchronfehler verzögern
42	EN_AIRT	Asynchronfehler (höherprior)freigeben

2.10 Datensatzübertragung

SFC Nr.	Name	Bezeichnung	Bemerkung
58	WR_REC	Datensatz schreiben	(SIWAREX nur nach DP-Norm)
59	RD_REC	Datensatz lesen	(SIWAREX nur nach DP-Norm)

2.11 Globaldaten-Kommunikation

SFC Nr.	Name	Bezeichnung	Bemerkung
60	GD_SND	GD-Paket senden	
61	GD_RCV	GD-Paket übernehmen	

2.12 Kommunikation über nicht projektierte Verbindungen

SFC Nr.	Name	Bezeichnung	Bemerkung
65	X_SEND	Daten senden extern	(nicht bei PCD1.M137)
66	X_RCV	Daten empfangen extern	(nicht bei PCD1.M137)
67	X_GET	Daten lesen extern	(nicht bei PCD1.M137)
68	X_PUT	Daten schreiben extern	(nicht bei PCD1.M137)
69	X_ABORT	Externe Verbindung abbrechen	(nicht bei PCD1.M137)

3 Onboard Funktionen

Die Steuerungen PCDx Serie xx7 sind mit zahlreichen Onboard-Funktionen ausge-stattet. Der Zugriff erfolgt über Systemfunktionen. Zu den Onboard Funktionen zählen:

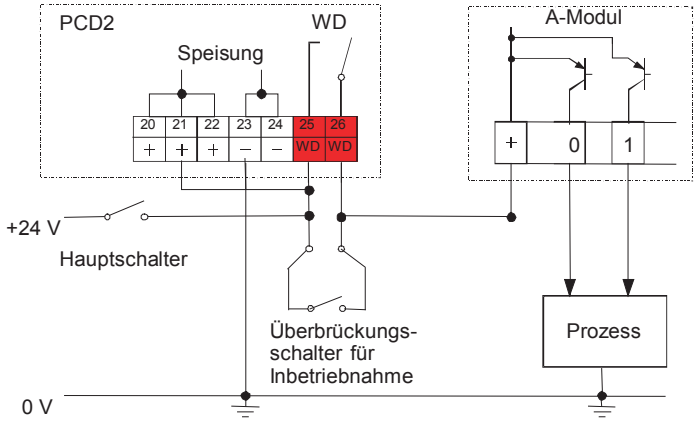
- Hardware-Watchdog, SFC239 WDOG
- Interrupt-Eingänge, SFC250 INP_INT, SFC256 CONF_AL (PCD2.M487)
- Unidirektionaler Zähler, SFC251 INTCNTR, SFC252 READCNTR
- Bidirektionaler Zähler, SFC248 INTDIR, SFB256 BOARDCNT (PCD2.M487)
- Integrierte SSI-Schnittstelle, SFC253 READ_SSI, SFC254 GRAY2BIN

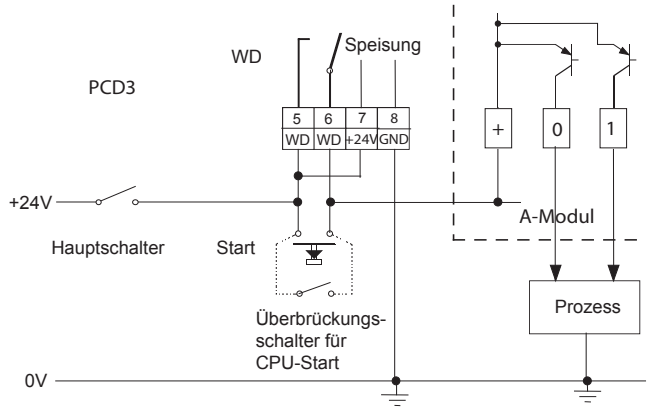
3.1 Hardware-Watchdog (SFC239 WDOG)

Auf den PCDx-CPU ist ein Watchdog-Relais integriert, über das z.B. die Freigabe für die Lastspannung geführt werden kann. Nach Aktivierung muss der Watchdog innerhalb einer Zeitspanne von ca. 350 ms (320...380 ms) durch Aufruf der SFC239 (WDOG) getriggert werden, um das Watchdog-Relais geschlossen zu halten. Dadurch wird eine besonders hohe Sicherheit selbst bei Ausfallen des SPS-Prozessors gewährleistet.

Das Watchdog-Relais befindet sich nur auf Steuerungen des Typs PCD2/3. Dieser Baustein wird vom Betriebssystem der PCD1 nicht unterstützt.

Schaltungsbeispiel und Anschlussbelegung PCD2 und PCD3:





Funktionsweise

Solange die Zykluszeit der SPS 350 ms (320...380 ms) nicht überschreitet, reicht ein einmaliger Aufruf im zyklischen SPS-Programm, um das Watchdog-Relais geschlossen zu halten. Bei längerer Zykluszeit kann der SFC239 mehrfach im SPS-Zyklus aufgerufen werden, oder in einem entsprechend parametrierten Weckalarm (z.B. OB35, Ausführung alle 300 ms) programmiert werden.

Beispiel:

```
Netzwerk 1: Wenn SPS im RUN-Zustand, Watchdog-Relais geschlossen halten

      CALL  "WDOG"                  // Aufruf SFC239
```

Die Adresse 255 ist für den Watchdog reserviert. Nur bei der PCD2.M177 ist zusätzlich die Adresse 511 für den Watchdog reserviert. Unabhängig davon, ob der Watchdog benutzt wird, können nicht alle PCD2 / PCD3-Module uneingeschränkt im Modulsteckplatz 16 genutzt werden. Nachfolgende Tabelle zeigt eine Aufstellung der Module und die Möglichkeit ihrer Verwendung im Modulsteckplatz 16:

PCD2 / PCD3 Module	auf Steckplatz 16 einsetzbar?
16 dig. Inputs	ja
16 dig. Outputs	nein
W1xx	ja
W2xx	nein
W3xx	nein (ausser W3x5)
W4xx	ja
W5xx Inputs	ja
W5xx Outputs	nein
W6xx	nein (ausser W6x5)
H- Module	nein

3.2 Interrupt-Eingänge

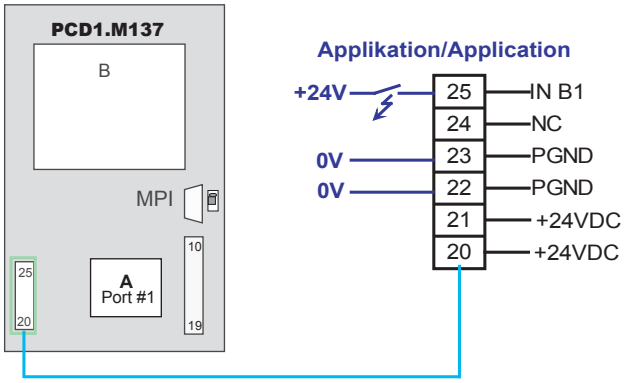
Mit Hilfe der integrierten Interrupt-Eingänge können schnelle Reaktionen unabhängig vom SPS-Zyklus erzielt werden. Nach Auslösen des Interrupts wird der Prozessalarm OB40 bis OB47 aufgerufen.

Auf der PCD2 belegen die Funktionalität der Interrupteingänge und der Zählerfunktionen dieselben Elemente. Es kann entweder die Interrupt- oder die Zählerfunktionalität genutzt werden.

	Anzahl Interrupt Eingänge	Programmierbar mit SFC	Prozessalarme
PCD1	1	250	OB 40
PCD2.M1x7	2	250	OB 40
PCD2.M487	4	256	OB 40 bis OB 47
PCD3.Mxxx7	2	256	OB 40 bis OB 47

3.2.1 Anschlussbelegung und Schaltungsbeispiel

Schaltungsbeispiel PCD1

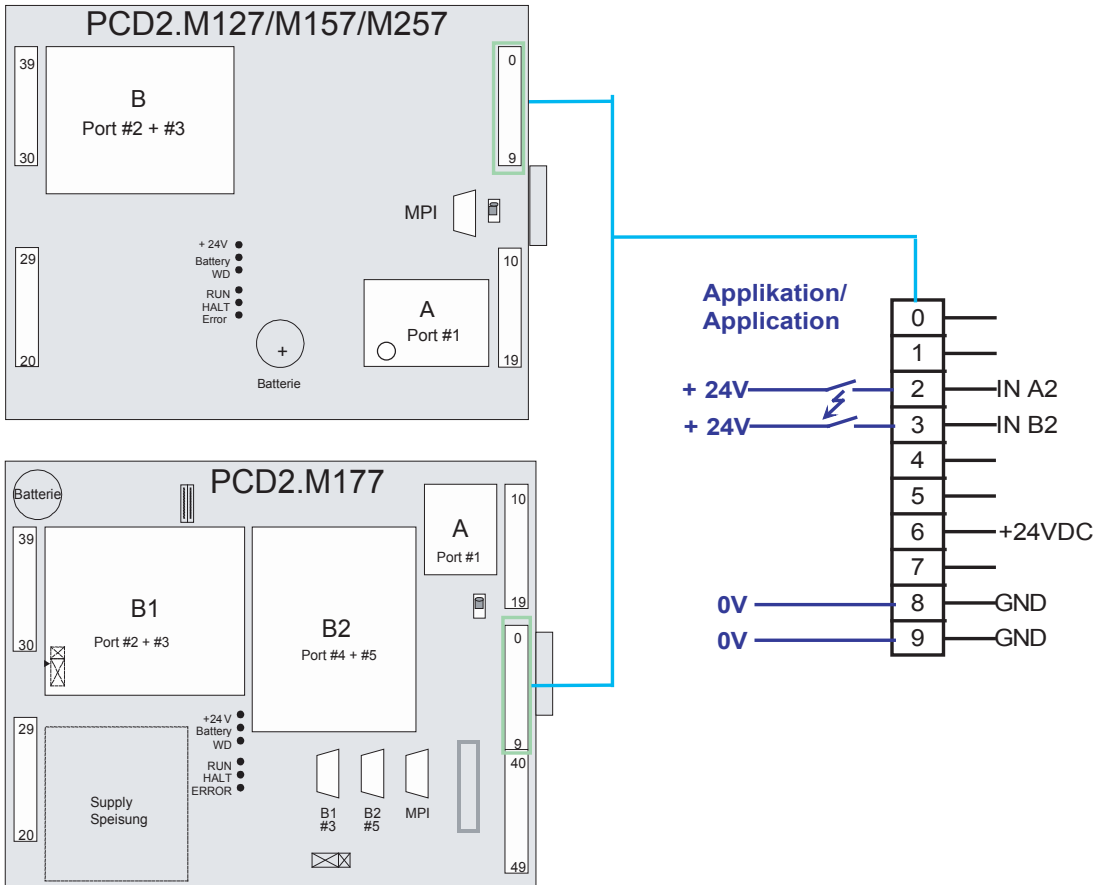


Zuordnung PCD1:

Name	Klemme	Signal	Interrupt wird ausgelöst durch:
Interrupteingang 1	25	IN B1	positive Flanke

3

Schaltungsbeispiel PCD2.M1x7

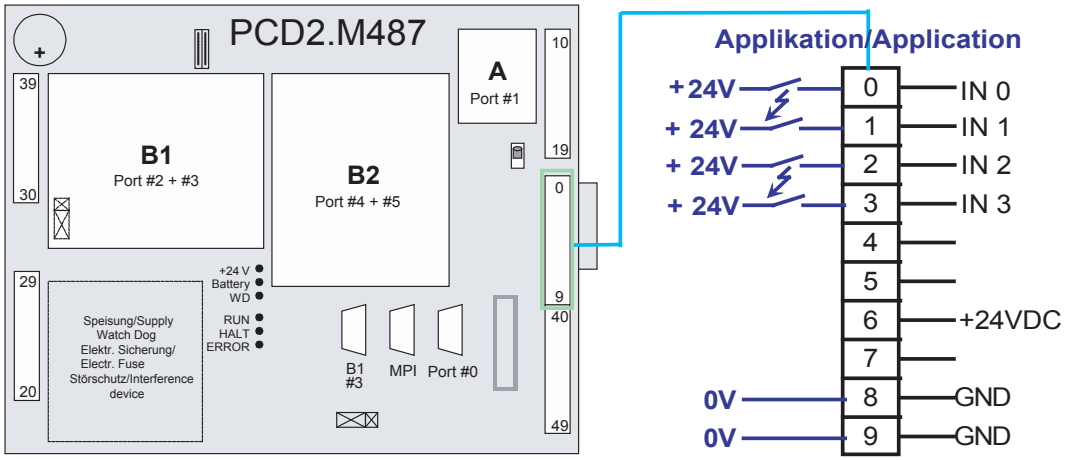


Zuordnung PCD2.M1x7

Name	Klemme	Signal	Interrupt wird ausgelöst durch:
Interrupteingang 0	2	IN A2	negative Flanke
Interrupteingang 1	3	IN B2	positive Flanke

Einschränkung:
Wenn an Klemme 3 (Interrupteingang 1) 24V anliegt, löst eine negative Flanke an der Klemme 2 (Interrupteingang 0) keinen Interrupt aus. Damit ist das Erfassen von der positiven und negativen Flanke eines Signals nicht möglich.

Schaltungsbeispiel PCD2.M487

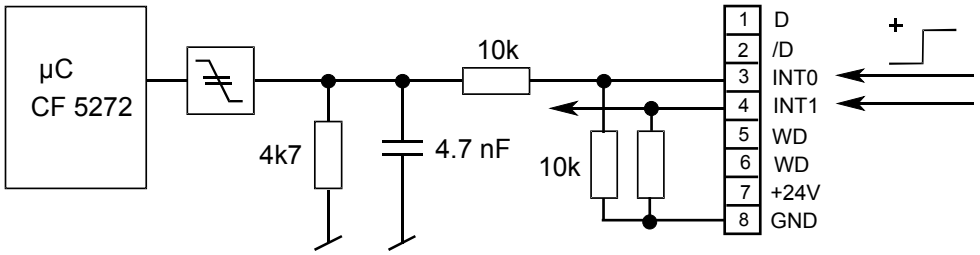


Zuordnung PCD2.M487

Name	Klemme	Signal	Interrupt wird ausgelöst durch:
Interrupteingang 0	0	IN0	positive Flanke
Interrupteingang 1	1	IN1	positive Flanke
Interrupteingang 2	2	IN2	positive Flanke
Interrupteingang 3	3	IN3	positive Flanke

Die vier Interrupteingänge können unabhängig voneinander konfiguriert und freigegeben, bzw. gesperrt werden.

Schaltungsbeispiel PCD3.Mxxx7



Zuordnung PCD3.Mxxx7

Name	Klemme	Signal	Interrupt wird ausgelöst durch:
Interrupteingang 0	3	INT0	positive Flanke
Interrupteingang 1	4	INT1	positive Flanke

Die zwei Interrupteingänge können unabhängig voneinander konfiguriert und freigegeben, bzw. gesperrt werden.

3.2.2 Interrupt Freigeben / Sperren (SFC 250 INP_INT)

Mit dem SFC250 (INP_INT) werden die Interrupteingänge freigegeben oder gesperrt. Auf der PCD2.M1x7 werden beide Interrups gleichzeitig gesperrt, bzw. freigegeben. Für das Freigeben, bzw. Sperren genügt ein einmaliger Aufruf des SFC 250 im Anwenderprogramm.

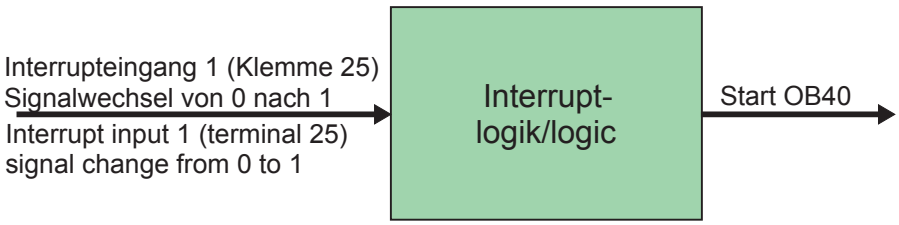
Parameter des SFC 250:

Parameter	Typ	Art	Bereich	Beschreibung
Enable (IN0)	Eingang	BOOL	TRUE / FALSE	TRUE: Interrupts Freigeben FALSE: Interrupts Sperren
Ret_Val	Ausgang	WORD	0	0: Kein Fehler

Die Funktionalität des SFC 250 steht auf der PCD2.M487 und der PCD3.Mxxx7 nicht zur Verfügung. Statt-dessen kann die Funktion SFC 256 CONF_AL benutzt werden.

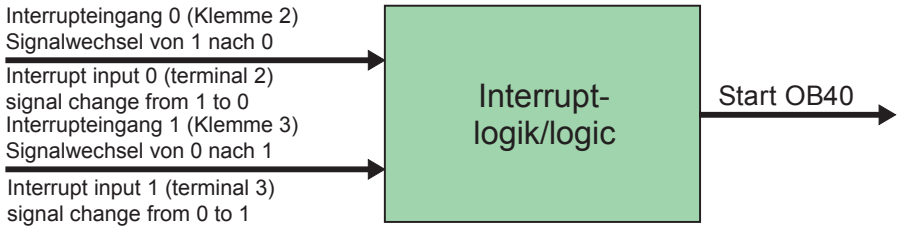
PCD1:

Nach der Freigabe des Interrupteingangs wird der OB40 automatisch aufgerufen, wenn ein Signalwechsel von 0 nach 1 am Interrupteingang 1 (Klemme 25) stattfindet.



PCD2:

Sind die Interrupteingänge freigegeben, wird bei einem Signalwechsel von 1 nach 0 am Interrupteingang 0 (Klemme 2), oder bei einem Signalwechsel von 0 nach 1 am Interrupteingang 1 (Klemme 3), automatisch der OB40 aufgerufen.



Beim Abarbeiten des OB40 kann ausgewertet werden, an welchem Eingang der Interrupt ausgelöst wurde. Dies geschieht durch Auswertung des Lokaldatenbytes “OB40_SRT_INF“. Dabei gilt folgende Zuordung:

PCD2: OB40_SRT_INF = B#16#41 → Interrupteingang 0 (Klemme 2)
OB40_SRT_INF = B#16#42 → Interrupteingang 1 (Klemme 3)
PCD1: OB40_SRT_INF = B#16#42 → Interrupteingang 1 (Klemme 25)

Programmierbeispiele:

Aufruf des SFC 250:

```
Netzwerk 1: Interrupt Freigeben
      U     E 0.7                    // Ein- Ausschalter
      FP    M 1.7
      SPBN  int0                     // Impuls Ein
// Interrupt freigeben
      CALL  "INP_INT"                // SFC 250
       IN0    :=TRUE                 // Interrupt freigeben
       RET_VAL:=MW250                // Rückgabewert

Netzwerk 2: Interrupt Sperren
int0: U     E 0.7                    // Ein- Ausschalter
      FN    M 1.6
      SPBN  int2                     // Impuls Aus
// Interrupt sperren
      CALL  "INP_INT"                // SFC 250
       IN0    :=FALSE                // Interrupt sperren
       RET_VAL:=MW250                // Rückgabewert
int2: NOP   0
```

Auswertung der Interruptquelle im OB40:

```
OB 40:  „Hardware Interrupt"
Netzwerk 1: Auswertung, welcher Interrupt?
      L     #OB40_STRT_INF
      L     B#16#41                  // Interrupteingang 0?
      ==I
      SPB   AlA0                     // Springe nach Alarm 0
      TAK
      L     B#16#42                  // Interrupteingang 1?
      ==I
      SPB   AlA1                     // Springe nach Alarm 1
      BEA

Netzwerk 2: Interrupt von Eingang 0
// Wenn Interrupt 0 ausgelöst wurde, soll der Ausgang A0.2 rückgesetzt
// werden.
AlA0: L     AB     0                 // Aktueller Zustand Ausgangsbyte 0
      L     2#11111011               // Bit 2 ausmaskieren
      UW                             // und
      T     PAB    0                 // sofort zurücksetzen
      BEA

Netzwerk 3: Interrupt von Eingang 1
// Wenn Interrupt 1 ausgelöst wurde, soll der Ausgang A0.3 rückgesetzt
// werden.
AlA1: L     AB     0                 // Aktueller Zustand Ausgangsbyte 0
      L     2#11110111               // Bit 3 ausmaskieren
      UW                             // und
      T     PAB    0                 // sofort zurücksetzen
      BEA
```

Wird während des Abarbeitens des Prozessalarms OB40 ein weiterer Interrupt ausgelöst, so tritt damit ein Zeitfehler auf und der OB80 wird automatisch aufgerufen.

3.2.3 Interrupt Eingänge Konfigurieren (SFC 256 CONF_AL)

Auf der M487 und der PCD3.Mxxx7 können die Interrupt Eingänge unabhängig voneinander konfiguriert werden. Zu jedem Interrupt Eingang wird der aufzurufende OB konfiguriert, sowie die Startinformation die in diesem OB hinterlegt wird. Für das Konfigurieren und Freigeben, bzw. Sperren genügt ein einmaliger Aufruf des SFC 256 im Anwenderprogramm.

Die Funktionalität des SFC 256 steht auf der PCD1 und den PCD2.M1x7 nicht zur Verfügung. Stattdessen kann die Funktion SFC 250 INP_INT benutzt werden.

Parameter des SFC 256:

Parameter	Deklaration	Typ	Bereich	Beschreibung
IRQ_NO (IN0)	Eingang	INT	0...3[1)]	Interrupt Eingang, entspricht den Klemmen IN0...IN3.
REQ_TYPE (IN1)	Eingang	BOOL	TRUE/ FALSE	TRUE: Interrupts Freigeben FALSE: Interrupts Sperren
OB_NR (IN2)	Eingang	INT	40...47	Nummer des bei einem Interrupt aufzurufenden OB.
OB_INFO (IN3)	Eingang	WORD	XXXX[2)]	Startinformation des Interrupt-OBs. Der Wert wird in den Lokaldaten abgelegt.
Ret_VAL	Ausgang	WORD	YYYY[3)]	Fehlermeldung: 0x0000: Kein Fehler 0x8080: IRQ_NO ausserhalb des zulässigen Bereichs (0...3) [4)] 0x8081: OB_NR ausserhalb des zulässigen Bereichs(40...47)

1) PCD3.Mxxx7 0...1 Interrupt Eingang entspricht den Klemmen INT0...INT1
2) 3) 2 Byte Bereiche von 0x0000 bis 0xFFFF
4) PCD3.Mxxx7, zulässiger Bereich 0...1

Durch das Auswerten des Lokaldatenwortes “OB_4x_MDL_ADDR” mit den projektierten Startinformation, kann beim Abarbeiten des Interrupt OBs die Interruptquelle ermittelt werden.

Programmierbeispiel:

Aufruf des SFC 256

```
Netzwerk 1: Interrupt 0 freigeben
      U     E 0.7              // Ein- Ausschalter
      FP    M 1.7
      SPBN  int0               // Impuls ein
// Interrupt 0 freigeben
      CALL  "CONF_AL"
       IN0    :=0              // Interrupt Eingang 0, Klemme IN0
       IN1    :=TRUE           // Enable
       IN2    :=41             // bei positiver Flanke wird OB 41 aufgerufen
       IN3    :=W#16#CAFE      // Startinformation in OB 41
       RET_VAL:=MW250          // MW enthält die Fehlerinformation
```

```
Netzwerk 2: Interrupt 0 Sperren

int0: U     E 0.7                    // Ein- Ausschalter
      FN    M 1.6
      SPBN  int1                     // Impuls Aus
// Interrupt 0 sperren
      CALL  „CONF_AL“
       IN0    :=0                  // Interrupt Eingang 0, Klemme IN0
       IN1    :=FALSE              // Disable
       IN2    :=41                 // Diese Info wird hier nicht ausgewertet
       IN3    :=W#16#CAFE          // Diese Info wird hier nicht ausgewertet
       RET_VAL:=MW250              // MW enthält die Fehlerinformation

int1: NOP   0
```

Auswertung im Interrupt OB41

```
OB 41:  „Hardware Interrupt“
Netzwerk 1:

      L     #OB41_MDL_ADDR         // Startinformation
      L     W#16#CAFE              // Projektierte Startinformation für
      ==I                          // Interrupt 0
      SPB   Int0                   // Springe nach Interrupt 0
      BEA

Netzwerk 2: Interrupt 0 wurde ausgelöst

Int0: NOP   0
// Interrupt Funktion ausführen
```

3.3 OnBoard Zähler

Mit Hilfe der OnBoard-Eingänge können Zähler unabhängig vom SPS-Zyklus realisiert werden. Nach Erreichen eines Vergleichswertes wird ein Prozessalarm OB aufgerufen.

Auf den Saia PCD® Steuerungen belegen die Funktionalität der Interrupteingänge und der Zählerfunktionen dieselben Elemente. Es kann entweder die Interrupt- oder die Zählerfunktionalität genutzt werden.

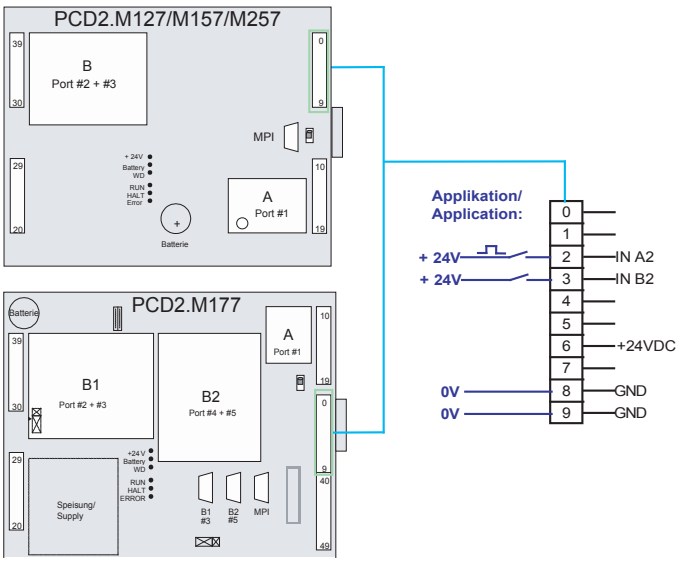
	Anzahl Zähler	Programmierbar	Prozessalarme
PCD1	0	-	-
PCD2.M1x7	1	SFC 251 / 252	OB 40
PCD2.M1x7	1	SFC 248	OB 41
PCD2.M487	2	SFB 256	OB 40 bis OB 47 / OnBoard Ausgang
PCD3.Mxxx7	1	SFB 256	OB 40 bis OB 47

3.3.1 Unidirektionaler Zähler (SFC251 INTCNTR, SFC252 READCNTR)

Mit Hilfe der integrierten Interrupt-Eingänge kann ein Zähler bis ca. 5 kHz genutzt werden. Dieser Zähler ist unidirektional, d.h. es kann nur in eine Richtung gezählt werden. Beim Erreichen von bis zu 2 Vergleichswerten wird ein Interrupt ausgelöst. Nach Auslösen des Interrupts wird der Prozessalarm OB40 aufgerufen. Über ein externes Freigabesignal kann der Zähler gesperrt oder aktiviert werden. Der Zählerstand kann mit Hilfe des SFC252 gelesen werden.

Die Funktionalität des unidirektionalen Zählers steht in der PCD1, der PCD2.M487 und der PCD3.Mxxx7 **nicht** zur Verfügung. Auf der PCD2.M487 und der PCD3.Mxxx7 kann alternativ die Funktion SFB256 BOARDCNT benutzt werden.

Schaltungsbeispiel und Anschlussbelegung:

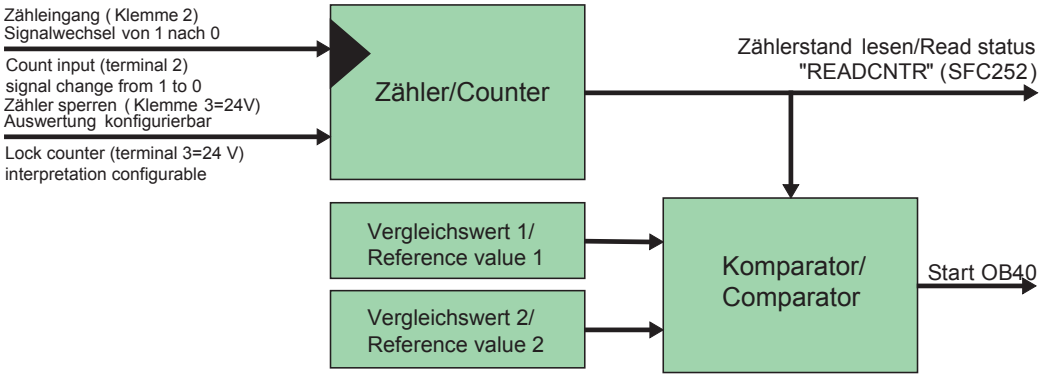


Zuordnung PCD2.M1x7

Name	Klemme	Signal	Beschreibung
Zählereingang	2	IN A2	Zählimpuls durch Signalwechsel 1 nach 0.
Zählerfreigabe	3	IN B2	Auswertung, wenn im Programm konfiguriert

Funktionsweise:

Ist der unidirektionale Zähler gestartet und konfiguriert, so ist seine prinzipielle Funktionsweise wie folgt:



Mit dem SFC251 (INTCNTR) wird der unidirektionale Zähler konfiguriert und gestartet oder gestoppt. Für das Konfigurieren, bzw. Starten des Zählers genügt ein einmaliger Aufruf des SFC251 im Anwenderprogramm.

Parameter des SFC251:

Parameter	Deklaration	Typ	Bereich	Beschreibung
START	Eingang	BOOL	TRUE/ FALSE	TRUE: Zähler starten FALSE: Zähler stoppen
ENABLE	Eingang	BOOL	TRUE/ FALSE	Signal an Klemme 3: TRUE: auswerten FALSE: nicht auswerten
INT2	Eingang	BOOL	TRUE/ FALSE	Beim Erreichen von VALUE2 OB40: TRUE: aufrufen FALSE: nicht aufrufen
INT1	Eingang	BOOL	TRUE/ FALSE	Beim Erreichen von VALUE1 OB40: TRUE: aufrufen FALSE: nicht aufrufen
VALUE2	Eingang	WORD	0x0002... 0xFFFF	Vergleichswert 2: VALUE2 muss grösser VALUE1 sein
VALUE1	Eingang	WORD	0x0002... 0xFFFF	Vergleichswert 1: VALUE2 muss grösser VALUE1 sein
RET_VAL	Ausgang	WORD	YYYY[1)]	Fehlermeldung: 0x0000: Kein Fehler 0x00FE: ungültiger Vergleichswert

[1)] 2Byte Bereiche von 0x0000 bis 0xFFFF.

Programmierbeispiel:

Aufruf des SFC 251

```
Netzwerk 1: Zaehler konfigurieren

      U     E      0.7          // Konfigurationsuebernahme
      FP    M      1.7
      SPBN  NOKO                // Impuls konfiguriere Zaehler
      CALL  "INTCNTR"           // Aufruf SFC 251
       START  :=E0.0            // 1 -> Start, 0 -> Stop Zaehler
       ENABLE :=E0.1            // 1 -> an Klemme3 24V = Sperren, 0 -> egal
       INT2   :=E0.2            // 1 -> OB 40 starten bei Vergleichswert 2
       INT1   :=E0.3            // 1 -> OB 40 starten bei Vergleichswert 1
       VALUE2 :=W#16#A          // Vergleichswert 2 (muss > VALUE1 sein
       VALUE1 :=W#16#5          // Vergleichswert 1
       RET_VAL:=MW240           // Fehlercode
NOKO: NOP   0                   // Sprunglabel
```

Nach der Initialisierung mit dem SFC251 steht der Zähler auf VALUE2. Beim ersten Zählimpuls zählt er auf 1.

Parallel zum Zählen kann der Zählerstand mit Hilfe des SFC252 (READCNTR) im Anwenderprogramm ausgelesen werden.

Parameter des SFC252:

Parameter	Deklaration	Typ	Bereich	Beschreibung
RET_VAL	Ausgang	WORD	YYYY[1)]	Aktueller Zählerstand

[1)] 2Byte Bereiche von 0x0000 bis 0xFFFF.

Programmierbeispiel:

Aufruf des SFC 252

```
Netzwerk 1: Zaehlerstand lesen

      CALL  "READCNTR"              // Aufruf SFC 252
       RET_VAL:=MW252               // Zaehlerstand lesen
```

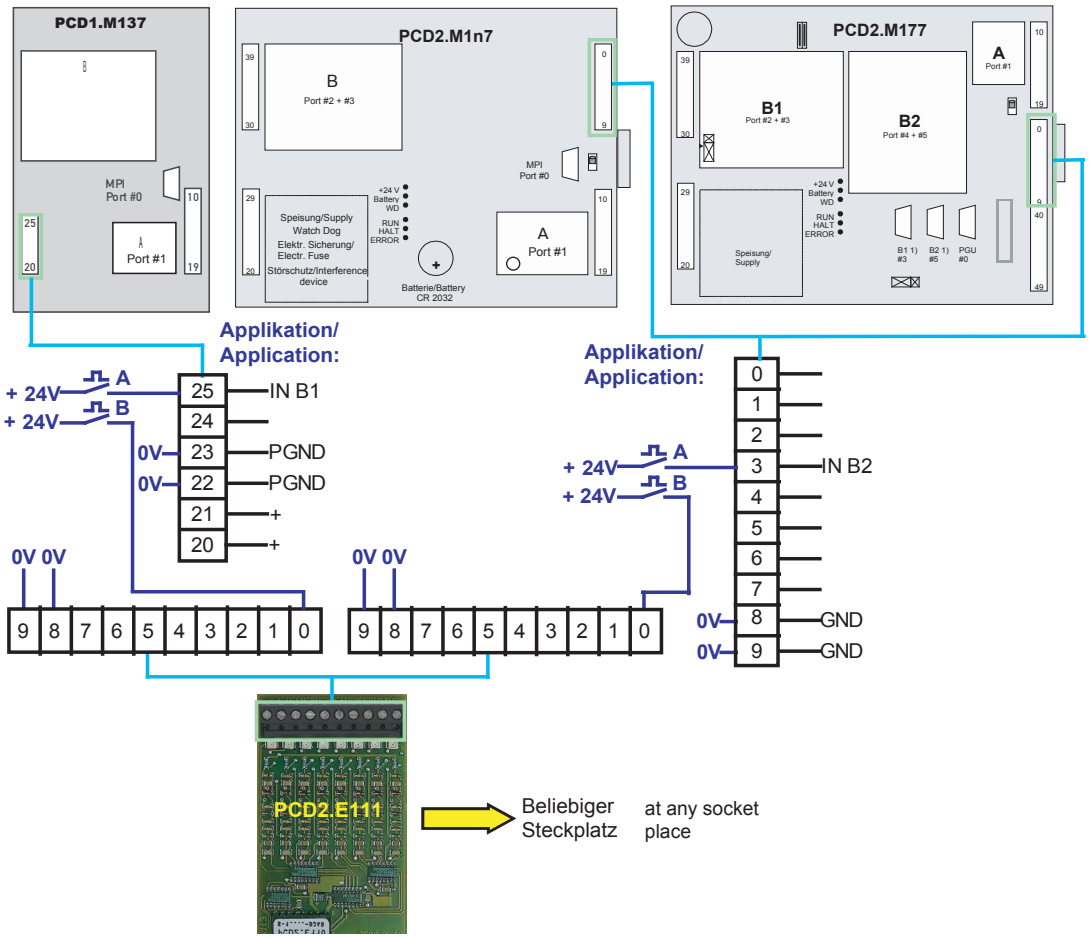
3.3.2 Bidirektionaler Zähler (SFC248 INTDIR)

Mit Hilfe des integrierten Interrupteingangs 1 und dem Bit 0 eines digitalen Eingangsmoduls kann ein bidirektionaler Zähler realisiert werden. Der Drehrichtungs-wechsel wird durch das Bit 0 des digitalen Eingangsmoduls (z.B. PCDx.E110, PCDx.E111) detektiert. Die Zählfrequenz ist abhängig vom Eingangsfilter des Eingangsmoduls. Wird das PCDx.E111 eingesetzt, so beträgt der Eingangsfilter 0.2ms, d.h. die maximale Zählfrequenz liegt bei ca. 5 kHz. Es kann kontinuierlich oder bis zu einem Maximalwert gezählt werden. Das Overflow Handling ist einstellbar. Bei Erreichen des Maximalwertes kann ein Interrupt ausgelöst werden. Nach Auslösen des Interrupts wird der Prozessalarm OB41 aufgerufen.

3

Die Funktionalität des bidirektionalen Zählers (SFC 248) ist auf der PCD2.M487 und der PCD3.Mxxx7 **nicht** vorhanden. Alternativ kann die Funktion SFB256 BOARDCNT benutzt werden.

Schaltungsbeispiel und Anschlussbelegung



Zuordnung PCD1:

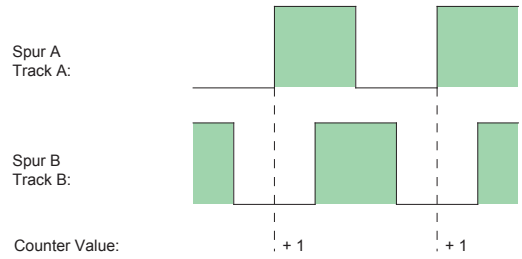
Name	Klemme	Signal	Beschreibung
Zähleingang	25	IN B1	Zählimpuls durch positive Flanke
Richtung	0	Bit 0	Digitales Eingangsmodul

Zuordnung PCD2.M1x7

Name	Klemme	Signal	Beschreibung
Zähleingang	3	IN B2	Zählimpuls durch positive Flanke
Richtung	0	Bit 0	Digitales Eingangsmodul

Einschränkung: Es werden nur die positiven Flanken am Zähleingang ausgewertet (1-fach Modus).

Ein Inkremantalgeber kann direkt an die Steuerung angeschlossen werden:
Spur A = Zähleingang auf Klemme 3 (25),
Spur B = Richtungseingang auf Klemme 0 (Bit 0).
Durch die Phasenverschiebung zwischen Spur A und Spur B wird dabei die Richtung detektiert:

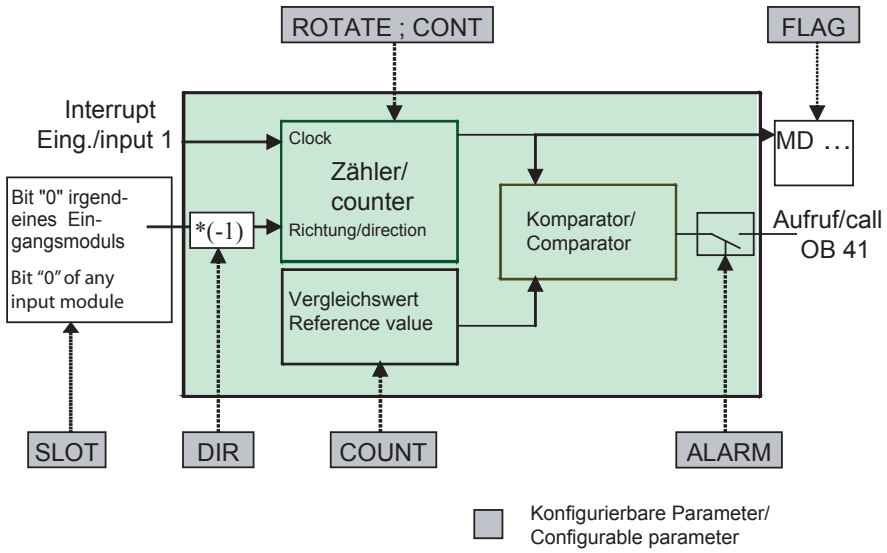


Dabei wird nur die positive Flanke der Spur A als Zählimpuls ausgewertet.

Funktionsweise

Mit dem SFC248 (INTCNTR) wird der bidirektionale Zähler konfiguriert und gestartet oder gestoppt. Für das Konfigurieren, bzw. Starten des Zählers genügt ein einmaliger Aufruf des SFC248 im Anwenderprogramm.

Ist der bidirektionale Zähler gestartet und konfiguriert, so ist seine prinzipielle Funktionsweise wie folgt:



Parameter des SFC248

Parameter	Deklaration	Typ	Bereich	Beschreibung
START	Eingang	BOOL	TRUE/ FALSE	TRUE: Zähler starten FALSE: Zähler stoppen
CONT	Eingang	BOOL	TRUE/ FALSE	TRUE: Kontinuierlich zählen FALSE: Zählen bis Maximalwert erreicht ist

ALARM	Eingang	BOOL	TRUE/ FALSE	TRUE: OB41 wird beim Erreichen vom Maximalwert aufgerufen. FALSE: OB41 wird nicht aufgerufen
ROTATE	Eingang	BOOL	TRUE/ FALSE	TRUE: Overflow Handling (siehe unten) FALSE: Kein Overflow Handling
DIR	Eingang	BOOL	TRUE/ FALSE	TRUE: Richtungssignal wird invertiert FALSE: Richtungssignal bleibt
SLOT	Eingang	INT	1...8	Modulsteckplatz des Eingangsmoduls für das Richtungsbit.
COUNT	Eingang	DWORD	XXXX[1)]	Maximalwert und Vergleichswert des Zählers.
FLAG	Eingang	INT	YYYY[2)]	Das erste Byte des Doppelwortes, das den Zählwert enthält.
RET_VAL	Ausgang	INT	YYYY[2)]	Fehlermeldung: 0: Kein Fehler -1: Falscher Zählwert (=0?) -2: Falsche FLAG-Adresse -3: Falsche SLOT-Adresse

1) 4Byte Bereiche von 0x0000`0000 bis 0xFFFF`FFFF.

2) Integer Bereiche von -32'768 bis +32'767.

ROTATE: Wird dieser Wert auf 1 gesetzt, wird fortwährend weiter gezählt. D.h. beim Vorwärtszählen bis zum Maximalwert (Parameter COUNT) und dann wieder mit 0 beginnend und beim Rückwärtszählen bis 0 und dann wieder beim Maximalwert (Parameter COUNT) beginnend weiter gezählt.

Wird dieser Wert auf 0 gesetzt, wird zwischen 0 und 0xFFFF`FFFF gezählt.

Parameter des OB41

Beim Aufruf des OB41 werden 2 Informationen in den Lokaldatenbytes zur Verfügung gestellt:

#OB41_RESERVED_1[BYTE]:

Wert=1 Vergleichswert wurde beim Aufwärtszählen erreicht,
Wert=0 bedeuted beim Abwärtszählen.

#OB41_POINT_ADDR[DWORD]:

Enthält den Vergleichswert (Maximalwert) des Zählers zum Zeitpunkt des Aufrufes von OB41.

Programmierbeispiele:
Aufruf des SFC 248

```
Netzwerk 1: Bidirektionalen Zaehler Starten / konfigurieren

      U     E      0.7            // Konfigurationsuebernahme
      FP    M      0.7
      SPBN  CNT0                  // Impuls Konfiguriere Zaehler
      CALL  "INT_DIR"
       START  :=E0.2              // 1->Start, 0->Stop
       CONT   :=E0.3              // 1->Kontinuierlich, 0->Stop bei Max
       ALARM  :=E0.4              // 1->Aufruf OB41, 0->kein Aufruf
       ROTATE :=E0.5              // 1->OverflowHandling, 0->keins
       DIR    :=E0.6              // 1->Invertierung, 0->Keine Invertierung
       SLOT   :=1                 // Modulsteckplatz vom dig. E-Modul
       COUNT  :=DW#16#A           // Vergleichswert (Maximalwert = 10)
       FLAG   :=10                // ->MD10 enthaelt Zaehlwert
       RET_VAL:=MW248             // Rueckgabewert, Fehler
CNT0: NOP   0
```

Auswertung des Interrupts im OB41

```
OB 41:  „Hardware Interrupt"
Netzwerk 1: Zaehlrichtung ermitteln

      L     0                          // Wird abwaerts gezaehlt?
      L     #OB41_RESERVED_1           // Zaehlrichtung
      ==I
      SPB   Down
      L     1                          // Wird aufwaerts gezaehlt?
      ==I
      SPB   UP
      BEA

Netzwerk 2: Abwaerts zaehlen

Down: L     #OB41_POINT_ADDR           // Lese Vergleichswert
// ... weiteres Programm

      BEA

Netzwerk 3: Aufwaerts zaehlen

UP:   L     #OB41_POINT_ADDR           // Lese Vergleichswert
// ... weiteres Programm

      BEA
```

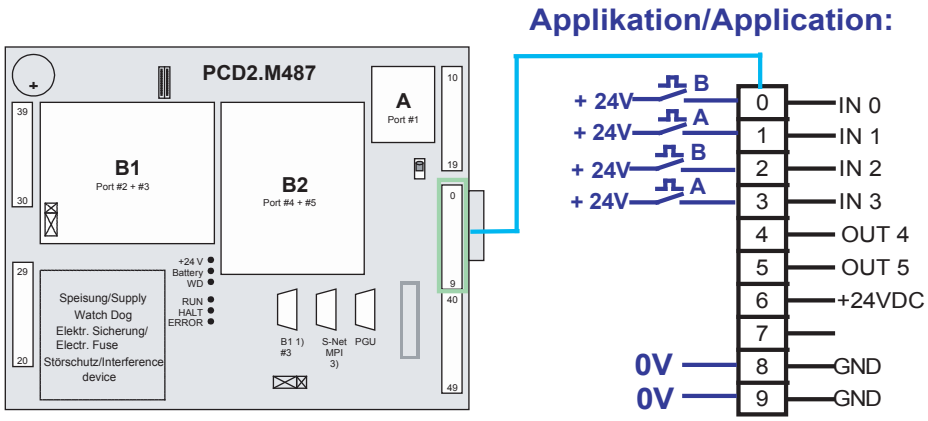
3

3.3.3 OnBoard Zähler auf PCD2.M487/PCD3.Mxxx7 (SFB256 BOARDCNT)

Mit Hilfe der vier OnBoard Interrupt–Eingängen (IN0 bis IN3) können zwei unabhängige bidirektionale Zähler realisiert werden. Dabei werden am jeweiligen Zähleingang die Impulse (positive Flanken) ausgewertet. Das Signal am zweite Eingang legt die Zählrichtung fest. Die maximale Zählfrequenz liegt bei ca. 1 kHz. Das Overflow-Handling ist einstellbar. Bei Erreichen des Vergleichswertes kann sowohl ein Interrupt ausgelöst, als auch ein OnBoard-Ausgang (Out4,Out5) gesetzt werden. Nach Auslösen des Interrupts wird der projektierte Alarm OB (40...47) aufgerufen.

Die Funktionalität des SFB 256 (OnBoard Zähler) wird auf der PCD1 und den PCD2.M1x7 **nicht** unterstützt. Auf diesen Systemen können alternativ die Funktionen SFC 251 (INTCNTR) oder SFC 248 INTDIR benutzt werden.

Schaltungsbeispiel und Anschlussbelegung PCD2.M487



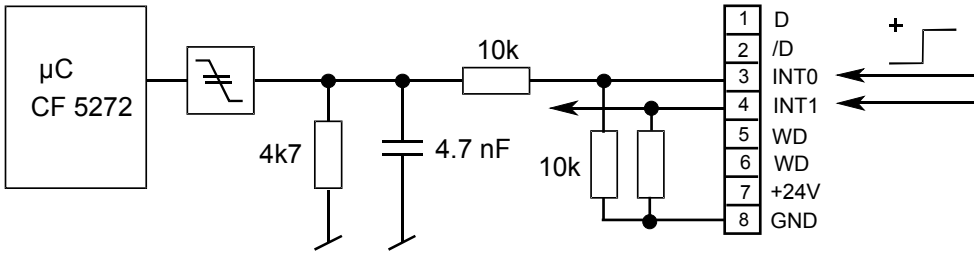
Aus technischen Gründen ist es notwendig am Eingang 0(2) das Richtungssignal (Signal B) und am Eingang 1(3) das Zählsignal (Signal A) anzuschliessen.

Zuordnung PCD2.M487

Name	Klemme	Signal	Beschreibung
Zähleingang 1	1	IN 1	Zählimpuls durch positive Flanke Zähler 1
Richtung 1	0	IN 0	Richtungseingang Zähler 1
Zähleingang 2	3	IN 3	Zählimpuls durch positive Flanke Zähler 2
Richtung 2	2	IN 2	Richtungseingang Zähler 2
Max Wert erreicht	4	OUT 4	Zähler 1 hat den REF Wert erreicht.
Max Wert erreicht	5	OUT 5	Zähler 2 hat den REF Wert erreicht.

Einschränkung: Es werden nur die positiven Flanken am Zähleingang ausgewertet (1-fach Modus).

Schaltungsbeispiel PCD3.Mxxx7

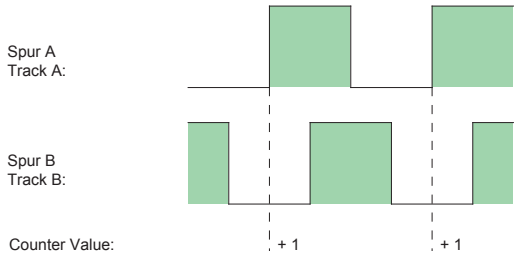


3

Zuordnung PCD3.Mxxx7

Name	Klemme	Signal	Interrupt wird ausgelöst durch:
Interrupteingang 0	3	INT0	positive Flanke
Interrupteingang 1	4	INT1	positive Flanke

Die zwei Interrupteingänge können unabhängig voneinander konfiguriert und freigegeben, bzw. gesperrt werden.



Ein Inkremantalgeber kann direkt an die Steuerung angeschlossen werden:
Spur A = Zähleingang auf Klemme 1 (3),
Spur B = Richtungseingang auf Klemme 0 (2).
Durch die Phasenverschiebung zwischen Spur A und Spur B wird dabei die Richtung detektiert:
Dabei wird nur die positive Flanke der Spur A als Zählimpuls ausgewertet.

Funktionsweise

Mit dem SFB256 (BOARDCNT) werden die beiden bidirektionalen Zähler auf der M487 bzw. der bidirektionale Zähler auf der PCD3.Mxxx7 konfiguriert und gestartet oder gestoppt. Für das Konfigurieren, bzw. Starten des Zählers genügt ein einmaliger Aufruf des SFB256 im Anwenderprogramm.
Der aktuelle Zählerstand kann jederzeit über die folgenden Peripherieadressen gelesen werden:

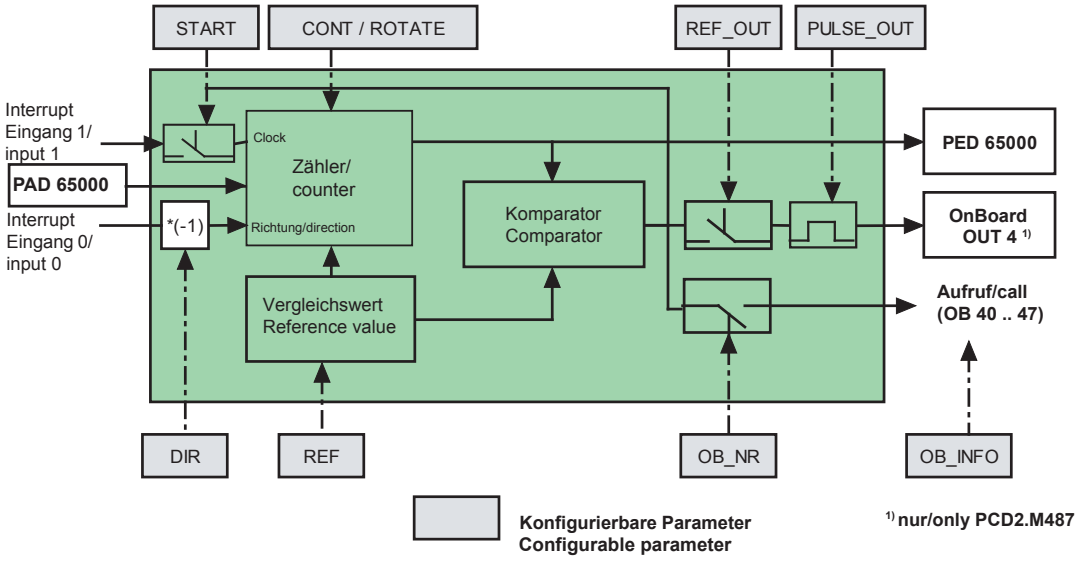
- Zähler 1: PED 65’000
- Zähler 2: PED 65’004 (nur PCD2.M487)

Der Zählerwert kann jederzeit über die folgenden Peripherieadressen vorbelegt werden:

- Zähler 1: PAD 65’000
- Zähler 2: PAD 65’004 (nur PCD2.M487)

Beim Aufstarten der Saia PCD®, werden die beiden Zähler auf 0 gesetzt. Geht die Saia PCD® in Stop, so werden die Zähler gestoppt.

Die Parameter können nur bei gestopptem Zähler geändert werden. Ausnahme: Der Zähler kann jederzeit mit einem neuen Wert belegt werden.
Nachfolgende Abbildung zeigt die prinzipielle Funktionsweise des OnBoard–Zählers auf der PCD2.M487 bzw. der PCD3.Mxxx7 anhand des Zählers 1:



Parameter des SFC248

Parameter	Deklaration	Typ	Bereich	Beschreibung
COUNT_NUM	Eingang	INT	1...2	1: Zähler 1 (IN0 , IN1, OUT4) 2: Zähler 2 (nur PCD2.M487) (IN2, IN3, OUT5)
START	Eingang	BOOL	TRUE/ FALSE	TRUE: Zähler starten FALSE: Zähler stoppen
CONT	Eingang	BOOL	TRUE/ FALSE	TRUE: Kontinuierlich zählen FALSE: Zählen bis Vergleichswert erreicht ist, dann Zähler stoppen.
ROTATE	Eingang	BOOL	TRUE/ FALSE	TRUE: Overflow-Handling (Siehe Tabellenende) FALSE: Kein Overflow-Handling
DIR	Eingang	BOOL	TRUE/ FALSE	TRUE: Richtungssignal wird invertiert FALSE: Richtungssignal bleibt unverändert.
REF_OUT (nur PCD2. M487)	Eingang	BOOL	TRUE/ FALSE	TRUE: Erreicht der Zähler den Vergleichswert, so werden die OnBoard-Ausgänge gesetzt. Zähler 1: OUT4 Zähler 2: OUT 5 FALSE: Das Erreichen des Vergleichswertes hat keinen Einfluss auf den OnBoard-Ausgang.
PULSE_OUT	Eingang	BOOL	TRUE/ FALSE	TRUE: Der OnBoard-Ausgang wird mit dem nächsten Zählimpuls zurück auf 0 gesetzt. FALSE: Der OnBoard-Ausgang bleibt solange gesetzt, bis er vom Anwender zurückgesetzt wird. (Siehe Tabellenende).
REF	Eingang	DWORD	XXXX[1)]	Maximalwert und Vergleichswert des Zählers.
OB_NR	Eingang	INT	0 40...47	0: Bei Erreichen des Vergleichswertes wird kein OB aufgerufen. 40...47: Bei Erreichen des Vergleichswertes wird der hier parametrierte OB aufgerufen.
OB_INFO	Eingang	WORD	YYYY[2)]	Der Wert wird beim Aufruf des Counter–OBs in das Lokaldatenwort OB_4x_MDL_ADDR kopiert.

RET_VAL	Ausgang	INT	ZZZZ[3]	Fehler- und Statusmeldung: 0: Der Zähler wurde gestartet. 1: Der Zähler läuft bereits. (Aufruf mit START=TRUE) 2: Der Zähler wurde gestoppt -2: Parameter COUNT_NUM ist ungültig (1...2) (nur PCD3.Mxxx7 [1]) -3: Parameter OB_NR ist ungültig (40...47 oder 0) -4: Die Counter Eingänge wurden bereits mit dem SFC256 als Interrupteingänge konfiguriert -5: Der Zähler wurde mit COUNT = TRUE und ROTATE = TRUE, und REF=0 initialisiert.
RESERVED	Ausgang	BOOL	TRUE/ FALSE	Reserviert

[1] 4Byte Bereiche von 0x0000`0000 bis 0xFFFF`FFFF.
[2] 2Byte Bereiche von 0x0000 bis 0xFFFF.
[3] Integer Bereiche von -32768 bis +32767.

Konnte der SFB256 ohne Fehler beendet werden, so wird das BIE-Bit zurückgesetzt. Im Fehlerfall wird das BIE-Bit gesetzt und RET_VAL enthält die Fehlerinformation.

ROTATE: Wird dieser Wert auf 1 gesetzt, wird fortwährend weiter gezählt. D.h. beim Vorwärtszählen bis zum Vergleichswert (Parameter REF) und dann wieder mit 0 beginnend und beim Rückwärtszählen bis 0 und dann wieder beim Vergleichswert (Parameter REF) beginnend weiter gezählt.

Wird dieser Wert auf 0 gesetzt, wird zwischen 0 und 0xFFFF`FFFF gezählt.

Rücksetzen des OnBoard–Ausgangs: Mit dem Befehl "T PAB 65’533" wird das Bit 4 vom Akku 1 auf den OnBoard-Ausgang OUT4 geschrieben. Damit kann der Ausgang des Zähler 1 zurückgesetzt werden (nur auf PCD2.M487).

Mit dem Befehl "T PAB 65’534" wird das Bit 5 vom Akku 1 auf den OnBoard-Ausgang OUT5 geschrieben. Damit kann der Ausgang des Zähler 2 zurückgesetzt werden (nur auf PCD2.M487).

Mit dem Befehl "T PAB 65’535" werden die Bits 4 und 5 vom Akku 1 auf die On-Board-Ausgänge OUT4 und OUT5 geschrieben. Damit kann der Ausgang des Zähler 1 und 2 gleichzeitig zurückgesetzt werden (nur auf PCD2.M487).

Programmierbeispiele:
Aufruf des SFB 256

```
Netzwerk 1: Zähler 1 konfigurieren und starten

      U     E      0.7               // Konfigurationsübernahme und Start
      FP    M      0.7
      SPBN  CNT0                     // Impuls Konfiguriere Zähler
      CALL  "BOARDCNT" , "BOARDCNT1"  // Aufruf SFB256 mit InstanzDB
       IN0    :=1                    // COUNT_NUM:Zähler 1 konfigurieren
       IN1    :=E0.1                 // START: 1->Start, 0->Stop
       IN2    :=E0.2                 // CONT: 1->Kontinuierlich, 0->Stop bei REF
       IN3    :=E0.3                 // ROTATE: 1->Overflow Handling, 0->keins
       IN4    :=E0.4                 // DIR: 1->Invertierung, 0->keine Invertierung
       IN5    :=E0.5                 // REF_OUT: 1: Set OUT4 bei REF,nicht gesetzt
       IN6    :=E0.6                 // PULSE_OUT: 1:OUT4 bleibt 1 bis zum nächsten Zählimpuls
       IN7    :=DW#16#A              // REF: Vergleichswert (Maximalwert=10)
       IN8    :=41                   // OB_NR: Wenn REF erreicht, call OB41
       IN9    :=W#16#C001            // OB_INFO: Startinformation im OB41
      RET_VAL:=MW240                 // RET_VAL: allfällige Fehlermeldung oder Statusinformation
       OUT10  :=M250.0               // RESERVED: Reserviert.
CNT0: NOP   0
```

Auswertung des Interrupts im OB41

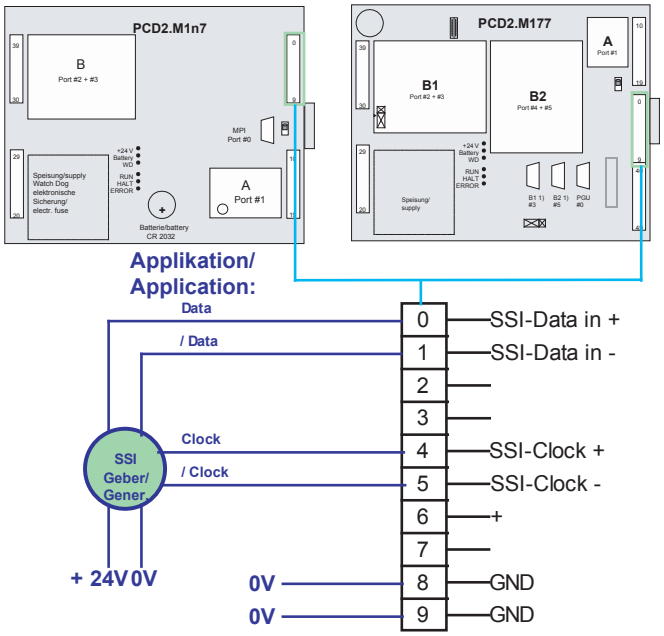
```
OB 41: „Zähler 1 Interrupt"
Netzwerk 1:

      L     #OB41_MDL_ADDR          // Startinformation
      L     W#16#C001               // Projektierte Startinformation
      ==I
      SPB   Int0                    // Springe zu Interrupt 0
      BEA

Netzwerk 2: Zähler 1 hat REF erreicht

Int0: NOP   0
// Interrupt Funktion ausführen

// ...
```

3.4 Integrierte SSI-Schnittstelle (SFC253 READ_SSI), (SFC254 GRAY2BIN)

Die integrierte SSI-Schnittstelle erlaubt das Auslesen von Absolutwertgebern. Bitformat und Wertcodierung können individuell angepasst werden.

Die Funktionalität der SSI–Schnittstelle steht in der PCD1, der PCD2.M487 und der PCD3.Mxxx7 **nicht** zur Verfügung.

Schaltungsbeispiel und Anschlussbelegung:



Zuordnung PCD2.M1x7

Name	Klemme	Signal	Beschreibung
SSI-Data in +	0	D	
SSI-Data in -	1	D/	
SSI-Clock Out +	4	C	
SSI-Clock Out -	5	C/	

Funktionsweise:

Die SSI-Schnittstelle ist immer aktiv. Sobald im Anwenderprogramm der SFC 253 (READ_SSI) aufgerufen wird, kann der aktuell anliegende Positionswert ausgelesen werden. Für den Fall, dass der Wert im Gray-Code vorliegt, kann dieser auf einfache Weise in ein für das Anwenderprogramm brauchbares Binärformat gewandelt werden.

Parameter des SFC253 (READ_SSI):

Parameter	Deklaration	Typ	Bereich	Beschreibung
BIT_CNT	Eingang	BYTE	1...32	Anzahl Bits, die eingelesen werden sollen.
RET_VAL	Ausgang	DWORD	YYYY[1)]	Gelesener Wert.

Parameter des SFC254 (GRAY2BIN):

Parameter	Deklaration	Typ	Bereich	Beschreibung
GRAY	Eingang	DWORD	YYYY[1)]	Wert in Gray Code
RET_VAL	Ausgang	DWORD	YYYY[1)]	Binärer Wert

[1)] 4Byte Bereiche von 0x0000`0000 bis 0xFFFF`FFFF.

Programmierbeispiele:

```
Netzwerk 1: SSI-Schnittstelle auslesen und Wert wandeln

// Schnittstelle auslesen
      CALL  "READ_SSI"              // Aufruf SFC 253
       BIT_CNT:=B#16#24             // 24 Bit werden gelesen
       RET_VAL:=MD250               // Temporaere Variable
// Wandlung Grey-Code in Binaer-Code
      CALL  "GRAY2BIN"              // Aufruf SFC 254
       GRAY   :=MD250               // Temporaere Variable
       RET_VAL:=MD100               // Aktuelle Position im Binaerformat
```

4 Kommunikations Funktionen

Die Saia PCD® Steuerungen der Serie xx7 bestechen durch eine breite Palette an Kommunikationsmöglichkeiten. Von der einfachen seriellen Datenschnittstelle über MPI-Vernetzung bis hin zu Feldbusanschaltungen und Telekommunikation via Modem bieten die Saia PCD® Steuerungen eine enorme Vielfalt von Kommunikationslösungen. In den folgenden Kapiteln werden nur beispielhaft die erweiterten Systemfunktionen für die Kommunikationsfunktionen beschrieben. Weitere detailliertere Informationen befinden sich in den funktionsspezifischen Handbüchern.

4.1 MPI-Protokoll

Die Steuerungen PCD1, PCD2 und PCD3 Serie xx7 können neben der üblichen MPI-Schnittstelle auch über eine serielle Anwenderschnittstelle programmiert werden. Dadurch kann zur Programmierung die serielle Schnittstelle direkt ohne MPI-Adapter mit dem PC verbunden werden. Für die PCD1 und PCD2.M1x7 benötigt man ein serielles Verbindungskabel (Siehe Handbuch “26/794 Serielle Kommunikation”) sowie ein RS-232 Schnittstellenmodul um sie mit der COM-Schnittstelle Ihres PCs zu verbinden. Falls auf der PCD2.M487 oder der PCD3.Mxxx7 das Port 0 (Default-Port) benützt wird, so ist kein Schnittstellenmodul notwendig. Weiterhin wird das Verbindungskabel PCD8.K111 PGU benötigt. Im SIMATIC-Manager muss die Schnittstelle auf “MPI-Adapter” mit 19‘200 Baud eingestellt sein.

Mit dem Konfigurationsdatenblock (CDB) kann eine beliebige Schnittstelle auf das MPI-Protokoll eingestellt werden. Ist kein CDB vorhanden, so wird immer auf Port 1, (auf der PCD2.M487 bzw den PCD3 auf Port 0) das MPI-Protokoll aktiviert.

Nach Netz Ein ist der MPI-Treiber auf der seriellen Schnittstelle aktiv. Sie können also mit Ihrer Programmiersoftware sofort online gehen.

Auf der PCD1 kann das MPI-Protokoll auf Port 1 nur mit Modem benutzt werden. Für Port 2 und 3 wird ein Sondermodul mit Spezialkabel benötigt.

Über die serielle Schnittstelle können Sie nur auf die direkt angeschlossene PCD1 / PCD2.M1x7 zugreifen, andere über MPI verbundene Steuerungen können nicht erreicht werden.

Auf der PCD2.M487 und der PCD3.Mxxx7 ist eine Gateway-Funktionalität vorhanden. D.h. über die serielle Schnittstelle können Steuerungen, die über das MPI Netzwerk mit einer PCD2.M487 oder einer PCD3.Mxxx7 verbunden sind, erreicht werden.

4.1.1 Aus- / Einschalten des MPI-Protokolls (SFC 200 Control)

Mit dem SFC 200 kann der Programmierer das MPI-Protokoll auf der seriellen Schnittstelle Ein- / Ausschalten. Der SFC 200 aktiviert / deaktiviert den MPI-Treiber auf der bei Netz Ein definierten Schnittstelle.

Auf der PCD2.M487 bzw der PCD3.Mxxx7 aktiviert / deaktiviert der SFC 200 den MPI-Treiber auf der zuletzt aktivierten Schnittstelle. Zum aktivieren / deaktivieren des MPI-Treibers auf einer beliebigen Schnittstelle kann der SFC300 benutzt werden.

Durch Aufruf der SFC 200 mit VKE = 0 wird das MPI-Protokoll deaktiviert. Durch Aufruf der SFC 200 mit VKE = 1 wird das MPI-Protokoll wieder aktiviert.

Programmierbeispiel:

```
Netzwerk 1: Ein- / Ausschalten MPI- Protokoll

       U      E 0.7                 // Ein-/Ausschalter
       FP     M 10.0                // Hilfsmerker Flanke Pos.
       SPBN   Aus
       SET                          // VKE Ein
       CALL   SFC 200               // MPI Ein

Aus:   U      E 0.7                 // Ein-/Ausschalter
       FN     M 10.1                // Hilfsmerker Flanke Neg.
       SPBN   End
       CLR                          // VKE Aus
       CALL   SFC 200               // MPI Aus

End:   NOP    0
```

4.1.2 Erweitertes Aus- / Einschalten des MPI-Protokolls (SFC 300 XControl)

Mit dem SFC 300 kann der Programmierer das MPI-Protokoll auf einer beliebigen seriellen Schnittstelle Ein- / Ausschalten. Der SFC 300 aktiviert / deaktiviert den MPI-Treiber auf der definierten Schnittstelle.

Die Funktion des SFC 300 steht auf der PCD1 und den PCD2.M1x7 **nicht** zur Verfügung. Alternativ kann die Funktion des SFC 200 Control benutzt werden.

Parameter des SFC 300:

Parameter	Typ	Art	Bereich	Beschreibung
PORT	IN 0	INT	0...6	Schnittstellen Nummer
DoStart	IN 1	BOOL	TRUE / FALSE	TRUE: Startet den MPI-Treiber FALSE: Stoppt den MPI-Treiber
Ret_Val	Ausgang	INT	XXXX[2)]	Fehlermeldung: 0:Funktion ohne Fehler ausgeführt 1:Funktion wird ausgeführt -1:Ungültige Schnittstellennummer -2:Interner Fehler -3:Treiber Fehler

[1)] PCD3..Mxxx7 Schnittstellen Nummer 0 bis 2
[2)] Integer Bereiche von -32’768 bis +32’767

Bevor das MPI-Protokoll auf einer neuen Schnittstelle aktiviert werden kann, muss es auf der alten Schnittstelle deaktiviert werden.

Mit dem SFC 300 kann das MPI-Protokoll nur aktiviert werden, wenn vorgängig die Schnittstelle mit einem CDB-Eintrag oder mit dem SFC 245 mit folgenden Par-ame-tern initialisiert wurde:

Mode:	1 = DK3964R
Baudrate:	19’200 oder 38’400
Data Bit:	8
Stop Bit:	1
Parity:	2 = Ungerade
ZVZ:	0 = Default Wert
QVZ:	0 = Default Wert
Send Buffer:	256
RCV Buffer:	256

Programmierbeispiel:

```
Netzwerk 1: Ein- / Ausschalten MPI-Protokoll

       U      E 0.7                 // Ein-/Ausschalter
       FP     M 10.0                // Hilfsmerker Flanke Pos.
       SPBN   next
       CALL   SFC 300
         IN0      :=1               // Schnittstelle 1
         IN1      :=TRUE            // MPI Ein
         RET_VAL  :=MW102           // Fehlermeldung
next:  U      „Switch 7“            // Ein-/Ausschalter
       FN     M 10.1                // Hilfsmerker Flanke Neg.
       SPBN   nex2
       CALL   SFC 300
         IN0      :=1               // Schnittstelle 1
         IN1      :=FALSE           // MPI Aus
         RET_VAL  :=MW102           // Fehlermeldung
nex2:  NOP 0
```

4.1.3 MPI-Protokoll Einschalten mit Konfigurationsdatenblock (CDB)

Um eine andere als die Default-Schnittstelle auf MPI-Protokoll zu konfigurieren, muss der Configuration Data Block (CDB, Ident = “SBC xx7 CDB“) programmiert werden.

Folgende Schnittstellen werden unterstüzt:

PCD1:	Schnittstelle 1...3 (default: 1), nur 19’200 Baud Schnittstelle 1 kann nur mit Modem benutzt werden (kein F120 Modul).
PCD2.M127, PCD2.M157:	Schnittstelle 1...3 (default: 1), 19’200 oder 38’400 Baud
PCD2.M177:	Schnittstelle 1...5 (default: 1), 19’200 oder 38’400 Baud
PCD2.M487:	Schnittstelle 0...5 (default: 0), 19’200 oder 38’400 Baud
PCD3.Mxxx7:	Schnittstelle 0..1 (default: 0), 19’200 oder 38’400 Baud

Auf den PCD2.M1x7 unterstützen die Schnittstellen auf Slot B1 und B2 per Default nur 19’200 Baud. Mit einem CDB Eintrag können die Schnittstellen 38’400 Baud anstelle von 19’200 Baud unterstützen. Diese Einstellung gilt für alle Schnittstellen auf dem entsprechenden Steckplatz.

Beispiel Schnittstelle 2:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	ParaCOM2	STRING[30]	´COM2:PTP_MPI,RS-232,19200,8,O,1´
=46.0		END_STRUCT	

Durch das Aktivieren einer Schnittstelle auf MPI wird das MPI-Protokoll auf der Default-Schnittstelle ausgeschaltet. Der CDB kann grundsätzlich auch für die Default-Schnittstelle benutzt werden.

Der Konfigurationsdatenblock (CDB) wird nur nach einem Kaltstart der CPU (Netz Ein) ausgewertet.

Weitere Informationen findet man im Kapitel Systemkonfiguration (CDB).

4.2 LON Kommunikation

Die Steuerungen der Serie PCD2.M1x7 können mit einer LON-Anschaltung (**L**ocal **O**perating **N**etwork) ausgerüstet werden. Die LON-Anschaltung wird mit SFC-Aufrufen programmiert. Netzwerkvariable SNVT und Explicit Messages werden in Datenbausteinen abgelegt. Es stehen 3 SFCs zur Verfügung:

1. Initialisierung des LON-Interface, SFC 220, LON_INIT
2. Senden einer Netzwerkvariablen SNVT, SFC 221, NV_SEND
3. Senden einer Message, SFC 223, MSG_SEND

Die Bausteine SFC 220, SFC 221 und SFC 223 sind nur im Betriebssystem der CPU vorhanden, wenn das LON-Modul gesteckt ist. Alternativ können sie aus der SBC Bibliothek in das Offline-Projekt kopiert werden.

Die LON-Funktionalitäten werden von der PCD1, der PCD2.M487 und der PCD3.Mxxx7 **nicht** unterstützt.

Näheres kann im Handbuch 26/767 LON nachgelesen werden. Der LON Netzwerk-Konfigurator SNET32 für xx7 steht kostenlos unter www.sbc-support.com zur Verfügung.

4.2.1 Initialisierung des LON-Interface (SFC 220 LON_INIT)

Der SFC 220 wird zur Initialisierung der LON-Schnittstelle und zur Definition der LON Datenbausteine benutzt.

Parameter des SFC 220:

Parameter	Typ	Art	Bereich	Beschreibung
REQ	Eingang	BOOL	TRUE / FALSE	TRUE: Initialisierung einschliesslich Reset
DB_NO	Eingang	WORD	1...1023	DB-Nr. Mit LON Konfiguration-Daten
Ret_Val	Ausgang	WORD	0	Fehlerinformation: siehe Kapitel Fehlerinformationen der LON SFCs

Nach der Initialisierung ruft die CPU den OB86 auf. Falls der OB86 nicht programmiert wurde, geht die CPU in Stop.

Im OB86 kann ausgewertet werden, ob während der Initialisierung ein Fehler aufgetreten ist, oder die Initialisierung korrekt ausgeführt werden konnte.

Programmierbeispiel:

LON Initialisierung im OB100:

```
Netzwerk 1: Initialisierung LON-Interface

       SET
       R      M 100.0                 // LON ist nicht initialisiert
       L      512                     // Config-DB Nummer = 512
       T      MW 200
       CALL   „LON_INIT"
         REQ      :=TRUE              // REQ: Start Initialisierung
         DB_NO    :=MW200             // DB_NO: Config-DB = 512
         RET_VAL  :=MW204
```

Auswertung im OB86:

```
Netzwerk 1: Testen ob LON Initialisierung OK.

       L      #OB86_EV_CLASS
       L      B#16#39                 // Fehler
       <>I
       SPB    ok                      // nein -> LON Initialisierung OK
       R      M 100.0
       L      512
       T      MW 200
       CALL   „LON_INIT"              // RE- Initialisierung LON
         REQ      :=TRUE              // REQ: Start Initialisierung
         DB_NO    :=MW200             // DB_NO: Config-DB = 512
         RET_VAL  :=MW204
       BEA
ok:    S      M 100.0                 // LON Initialisierung ist OK
```

Die möglichen Fehlermeldungen des OB86 sind im Handbuch 26/767 LON beschrieben.

4.2.2 Senden einer Netzwerkvariablen SNVT (SFC 221 NV_SEND)

Der SFC 221 sendet eine Netzwerk-Variable (NV). Diese wird an einen bestimmten Knoten gesendet.

Parameter des SFC 221:

Parameter	Typ	Art	Bereich	Beschreibung
REQ	Eingang	BOOL	TRUE / FALSE	TRUE: SEND
VAR_NAME	Eingang	ANY		Symbolischer Name der Netzwerk-Variablen
Ret_Val	Ausgang	WORD	0	Fehlerinformation: siehe Kapitel Fehlerinformationen der LON SFCs
BUSY	Ausgang	BOOL	TRUE / FALSE	TRUE: Sende Auftrag läuft.
DONE	Ausgang	BOOL	TRUE / FALSE	TRUE: Sende Auftrag fertig ohne Fehler
ERROR	Ausgang	BOOL	TRUE / FALSE	TRUE: Sende Auftrag fertig mit Fehler

Programmierbeispiel:

```
Netzwerk 1: LON Netzwerkvariable Senden

       CALL  "NV_SEND"
         REQ      :=M100.1
         VAR_NAME :=DB120.DBD1      // Symbolischer Name der NV
         RET_VAL  :=MW222
         BUSY     :=M200.0
         DONE     :=M200.1
         ERROR    :=M200.2
```

4.2.3 Senden einer Message (SFC 223 MSG_SEND)

Der SFC 223 sendet eine explizite Mitteilung. Diese explizite Mitteilung wird an den bestimmten Knoten gesendet.

Parameter des SFC 223:

Parameter	Typ	Art	Bereich	Beschreibung
REQ	Eingang	BOOL	TRUE / FALSE	TRUE: SEND
MSG	Eingang	ANY		Mitteilungs-Name
LEN	Eingang	INT	1... 32'767	Länge der Mitteilung
Ret_Val	Ausgang	WORD	0	Fehlerinformation: siehe Kapitel Fehlerinformationen der LON SFCs
BUSY	Ausgang	BOOL	TRUE / FALSE	TRUE: Sende Auftrag läuft.
DONE	Ausgang	BOOL	TRUE / FALSE	TRUE: Sende Auftrag fertig ohne Fehler
ERROR	Ausgang	BOOL	TRUE / FALSE	TRUE: Sende Auftrag fertig mit Fehler

Programmierbeispiel:

```
Netzwerk 1: LON-Mitteilung senden

       CALL  "MSG_SEND"
         REQ      :=M100.2
         MSG      :=DB120.DBD4          // Mitteilungsname
         LEN      :=20
         RET_VAL  :=MW224
         BUSY     :=M201.0
         DONE     :=M201.1
         ERROR    :=M202.2
```

4.2.4 Fehlerinformationen der LON SFCs

Die SFCs 220, 221 und 223 setzen das Binärergebnisbit (Bit 8) des Status Registers der CPU zurück, wenn sie ohne Fehler ausgeführt werden konnten. Im Falle eines Fehlers wird das Binärergebnisbit (BIE) gesetzt und die Variable RET_VAL enthält die weitere Fehlerinformation. Die Fehlercodes sind teilweise nach Standard S7 ausgeführt. Einige Fehlercodes wurden speziell für xx7-LON hinzugefügt.

BUSY	DONE	ERROR	BIE	RET_VAL	Beschreibung
0	0	0	0	0x7000	Erster Aufruf mit REQ = 0: kein Daten Transfer aktiv
1	0	0	0	0x7001	Erster Aufruf mit REQ = 1: Daten Transfer gestartet
1	0	0	0	0x7002	Zwischenaufruf (REQ irrelevant): Daten Transfer ist bereits aktiv
0	1	0	0	0x0000	Daten Transfer erfolgreich abgeschlossen
0	0	1	0	0x0001	LON interface Fehlfunktion
0	0	1	0	0x0002	Der Job kann innerhalb der Zeit nicht angenommen werden
0	0	1	0	0x4000	Datenübertragung war nicht erfogreich
0	0	1	1	0xFFFF	Der Job konnte nicht angenommen werden, weil zuwenig (interner) Speicher vorhanden ist. Später versuchen!
0	0	1	1	0xFFFE	Die spezifizierte Netzwerkvariable konnte nicht gefunden werden
0	0	1	1	0xFFFD	Fehler im LON Konfiguration Datenbaustein
0	0	1	1	0xFFFC	Der LON-Treiber ist noch nicht initialisiert
0	0	1	1	0xFFFB	Die Datenbaustein Nummer passt nicht zu der Initialisierung
0	0	1	1	0xFFFA	Nachricht ist zu lang
0	0	1	1	0x803A	Der spezifizierte DB (z.B der SNVT-DB) wurde nicht gefunden
0	0	1	1	0x8022	Der Datenbaustein enthält nicht die SNVT oder message (DB Längen Fehler)

4.3 Serielle Kommunikation

Die Steuerungen der Serie PCD1/PCD2.M1x7 unterstützen 2 Möglichkeiten der seriellen Kommunikation:

1. SBC-Modus
2. CP 441

Die Steuerungen der Serie PCD2.M487 und PCD3.Mxxx7 unterstützen nur 1 Möglichkeit der seriellen Kommunikation:

1. CP 441

Konfiguration
Mit dem SFC 245 kann das COM Port konfiguriert werden.

SBC-Modus
Der seit Beginn der Serie PCD1/PCD2.M1x7 zur Verfügung stehende Modus beinhaltet 4 SFCs (240…243). Mit diesen kann auf den Sende- und Empfangspuffer transparent zugegriffen werden. Die Möglichkeiten sind völlig offen, aber durch Zykluszeit und Puffergrösse beschränkt.

CP 441
In diesem Modus kann mit der SIMATIC® Umgebung kommuniziert werden. So werden die meist genutzten Protokolle, wie ASCII, DK3964R und RK512 unterstützt.

Mit den SFB 12, 13 und 14, die die gleichen SFB darstellen wie bei der SIMATIC®, wird die Kommunikation abgehandelt.

Ausserdem sind 2 Protokolle vorhanden, die nicht kompatibel zur SIMATIC® sind, da sie mehr Funktionalität beinhalten.

1. RK512MP: Wie RK512, aber es unterstützt ein Master-Slave Netzwerk (Multi-point MP). Auf der PCD2.M487 und PCD3.Mxxx7 wird das RK512MP Protokoll nicht unterstüzt.
2. Transparent: Wie Saia Modus, kann aber mehr Daten verarbeiten.

Modem Signale
Ausserdem gibt es für den direkten Zugang zu Modemsignalen den SFC 244. Für die Steuerungen der Serie PCD2.M487 und PCD3.Mxxx7 steht ein erweiterter Zugang zu den Modemsignalen den SFC344 zur Verfügung.

In den folgenden Kapiteln der seriellen Kommunikation werden nur beispielhaft die erweiterten Systemfunktionen des SBC-Modus und der Initialisierungsbaustein SFC 245 für den CP441 Modus beschrieben. Weitere detailliertere Informationen befinden sich im Handbuch 26/794 Serielle Kommunikation.

Die SFCs 240 bis 243, sowie das RK512MP Protokoll werden auf der PCD2.M487 und PCD3.Mxxx7 **nicht** unterstützt.

4.3.1 Daten von der seriellen Schnittstelle lesen (SFC 240 COM_RCV)

Nachdem eine serielle Schnittstelle initialisiert wurde, beginnt das Betriebssystem selbständig Zeichen über die serielle Schnittstelle zu empfangen und diese im Em-pfangspuffer abzulegen. Durch Aufruf des SFC 240 “COM_RCV” wird die spezifizierte Anzahl Zeichen (Bytes) aus dem Empfangspuffer in einen Datenbereich übertragen, der durch einen ANY-Pointer selektiert wird. Der ANY-Pointer spezifiziert nicht nur die Startadresse sondern auch die Länge des Datenbereiches. Der Datenbereich kann beginnend bei einem Byte bis zu 128 Bytes (Grösse des Empfangspuffers) gross sein.

Der SFC 240 wird auf der PCD2.M487 und PCD3.Mxxx7 **nicht** unterstützt.

Parameter des SFC 240:

Parameter	Typ	Art	Bereich	Beschreibung
COM_NR	Eingang	BYTE	1...5 / 1...3	Gibt an, welche Schnittstelle initialisiert werden soll. Erlaubte Werte: PCD2.M177 Port 1...5 PCD1 und PCD2.M127/157/257 Port 1...3
BUFFER	Eingang	ANY		Spezifiziert die Startadresse und Länge des Datenbereichs, in den die empfangenen Zeichen (Bytes) übertragen werden sollen. (Zeiger auf Datenbereich)
Ret_Val	Ausgang	INT	-1...1	Fehlermeldung: 0: Kein Fehler 1: Nicht genügend Bytes im Empfangspuffer -1: Ungültige Schnittstellennummer

Programmierbeispiel:

```
Netzwerk 1: Daten aus Schnittstelle lesen

       CALL   "COM_RCV"
         COM_NR   :=B#16#1                    // Schnittstellen Nr. 1
         BUFFER   :=P#M 1000.0 BYTE 20        // Zielbereich Empfangsdaten
         RET_VAL  :=MW240                     // Fehlermeldung
```

4.3.2 Daten an die serielle Schnittstelle senden (SFC 241 COM_SEND)

Nach Aufruf des SFC 241 “COM_SEND” wird ein Datenbereich, der durch einen ANY-Pointer selektiert wird, in den Sendepuffer übertragen. Der ANY-Pointer spezifiziert nicht nur die Startadresse sondern auch die Länge des Datenbereiches. Der Datenbereich kann beginnend bei einem Byte bis zu 128 Bytes (Grösse des Sende-puffers) gross sein. Das eigentliche Senden erfolgt durch das Betriebssystem im Hintergrund.

Der SFC 241 wird auf der PCD2.M487 und PCD3.Mxxx7 **nicht** unterstützt.

4

Parameter des SFC 241:

Parameter	Typ	Art	Bereich	Beschreibung
COM_NR	Eingang	BYTE	1...5 / 1...3	Gibt an, welche Schnittstelle initialisiert werden soll. Erlaubte Werte: PCD2.M177 Port 1...5 PCD1 und PCD2.M127/157/257 Port 1...3
BUFFER	Eingang	ANY		Spezifiziert die Startadresse und Länge des Datenbereichs (Byte), der gesendet werden soll. (Zeiger auf Datenbereich)
Ret_Val	Ausgang	INT	-1...1	Fehlermeldung: 0: Kein Fehler 1: Nicht genügend Bytes im Sendepuffer -1: Ungültige Schnittstellennummer

Programmierbeispiel:

```
Netzwerk 1: Daten an die Schnittstelle senden

       U      E 0.0                    // Sendeanstoss
       FP     M 0.0
       SPBN   NSND                     // Impuls
       CALL   "COM_SEND"
         COM_NR   :=B#16#1             // Schnittstellen Nr. 1
         BUFFER   :=P#M 2000.0 BYTE 20 // Quellbereich Sendedaten
         RET_VAL  :=MW240              // Fehlermeldung
NSND:  NOP    0
```

4.3.3 Status der seriellen Schnittstelle lesen (SFC 242 COM_STAT)

Nach Aufruf des SFC 242 “COM_STAT” wird der Status der spezifizierten Schnittstelle zurückgegeben. Der SFC 242 liefert folgende Informationen:

- Die Anzahl der empfangenen Zeichen, die sich im Empfangspuffer befinden.
- Die Anzahl der Zeichen die sich im Sendepuffer befinden und noch nicht gesendet wurden.
- Empfangspuffer-Überlauf, d.h. es wurden mehr Zeichen empfangen, als in den Empfangspuffer passen. Ein Empfangspuffer-Überlauf tritt auf, wenn nicht schnell genug Daten aus dem Empfangspuffer mit dem SFC 240 “COM_RCV” ausgelesen werden.
- Schnittstellen-Fehler, d.h. es wurden nicht korrekte Zeichen empfangen (z.B. falsche Baudrate, falsche Parität, EMV-Störungen, etc.).

Der SFC 242 wird auf der PCD2.M487 und PCD3.Mxxx7 **nicht** unterstützt.

Parameter des SFC 242:

Parameter	Typ	Art	Bereich	Beschreibung
COM_NR	Eingang	BYTE	1...5 / 1...3	Gibt an, welche Schnittstelle initialisiert werden soll. Erlaubte Werte: PCD2.M177 Port 1...5 PCD1 und PCD2.M127/157/257 Port 1...3
Ret_Val	Ausgang	INT	-1...0	Fehlermeldung: 0: Kein Fehler -1: Ungültige Schnittstellennummer
RCV_CNT	Ausgang	INT	0...128	Anzahl Bytes im Empfangspuffer
SND_CNT	Ausgang	INT	0...128	Anzahl Bytes im Sendepuffer
STATUSL	Ausgang	INT	0...1	Fehlermeldung: Bit 0 = 1 Empfangspufferüberlauf Bit 1 = 1 Schnittstellenfehler

Programmierbeispiel:

```
Netzwerk 1: Status der Schnittstelle lesen

      CALL   "COM_STAT"
        COM_NR   :=B#16#1          // Schnittstellen Nr. 1
        RET_VAL  :=MW240           // Fehlermeldung
        RCV_CNT  :=MW242           // Anzahl Bytes im Empfangsbuffer
        SND_CNT  :=MW244           // Anzahl Bytes im Sendepuffer
        STATUS   :=MW246           // Ueberlauf / Schnittstellenfehler
```

4.3.4 Serielle Schnittstelle initialisieren (SFC 243 COM_INIT)

Mit dem SFC 243 können die Schnittstellen 1 bis 5 initialisiert werden (nicht MPI). In der Regel wird der SFC nur einmal vor Beginn der seriellen Kommunikation aufgerufen, z.B. im OB100.

Der SFC 243 wird auf der PCD2.M487 und PCD3.Mxxx7 **nicht** unterstützt.

Parameter des SFC 243:

Parameter	Typ	Art	Bereich	Beschreibung
COM_NR	IN 0	BYTE	1...5 / 1...3	Gibt an, welche Schnittstelle initialisiert werden soll. Erlaubte Werte: PCD2.M177 Port 1...5 PCD1 und PCD2.M127/157/257 Port 1...3
SELECT	IN 1	BYTE	0...3	Schnittstellen Modus: 0: RS-232 1: RS-485 2: RS-422 3: TTY 20mA
BAUDRATE	IN 2	DINT	300... 38'400	Baudrate: erlaubte Werte sind 300, 600, 1'200, 2'400, 4'800, 9'600, 19'200, 38'400 (nur COM1)
COM_PAR	IN 3	WORD		Schnittstellen Parameter: Bit 1...0: Anzahl Datenbits: 00=5, 01=6, 10=7, 11=8 Bit 4...2: Parität: 000=gerade, 001=ungerade, 010=forced low, 011=forced high, 10x=keine Bit 5: Stopbits: 0=1 Stopbit, 1=2 Stopbits
Ret_Val	Ret_Val	INT	-1...0	Fehlermeldung: 0: Kein Fehler -1: Ungültige Schnittstellennummer

Programmierbeispiel:

```
Netzwerk 1: Serielle Schnittstelle initialisieren

      CALL   "COM_INIT"
        COM_NR   :=B#16#1       // Schnittstelle Nr. 1
        SELECT   :=B#16#0       // RS-232
        BAUDRATE :=L#9600       // 9600 Baud
        COM_PAR  :=W#16#3       // 8 Datenbits, gerade Parität, 1 Stopbit
        RET_VAL  :=MW240        // Fehler ?
```

4.3.5 Zugang zu den Modem Signalen (SFC 244 COM_SIG)

Durch den Aufruf der SFC 244 kann das

- **DTR** (Data Transmit Receive) Signal geändert, sowie der Status der Signale
- **DCD** (Data Carrier Detect) und
- **DSR** (Data Set Ready)

der COM1- Schnittstelle abgefragt werden.

Dieser Baustein ist nur für die Schnittstelle 1. Das DTR-Signal ist invertiert. Somit ist es per Default im Signalzustand 1. Um es zu aktivieren, wird der SFC 244 mit einer 0 am DTR- Eingang aufgerufen. Um es zu deaktivieren, wird der SFC 244 mit einer 1 am DTR- Eingang aufgerufen.

Parameter des SFC 244:

Parameter	Typ	Art	Bereich	Beschreibung
DTR	IN 0	BOOL	TRUE / FALSE	Data Transmit Receive Signal steuern
DCD	OUT 1	BOOL	TRUE / FALSE	Data Carrier Detect Signal abfragen
DSR	OUT 2	BOOL	TRUE / FALSE	Data Set Ready Signal abfragen

Programmierbeispiel:

```
Netzwerk 1: Zugang zu den Modemsignalen

       CALL   "COM_SIG"
         DTR:=M1.0                  // DTR steuern
         DCD:=M1.1                  // Zustand des DCD Signal
         DSR:=M1.2                  // Zustand des DSR Signals
```

4.3.6 Erweiterter Zugang zu den Modem Signalen (SFC 344 XCOM_SIG)

Durch den Aufruf der SFC 344 können die Signale

- **DTR** geändert, sowie der Status der Signale
- **DCD** und
- **DSR**

einer COM- Schnittstelle auf der **PCD2.M487 und PCD3.Mxxx7** abgefragt werden.

Das DTR- Signal ist invertiert. Somit ist es per Default im Signalzustand 1. Um es zu aktivieren, wird der SFC 344 mit einer 0 am DTR- Eingang aufgerufen. Um es zu deaktivieren, wird der SFC 344 mit einer 1 am DTR- Eingang aufgerufen.

Diese Funktion wird auf der PCD1 und den PCD2.M1x7 nicht unterstützt. Alternativ kann die Funktion SFC 244 COM_SIG benutzt werden.

Parameter des SFC 344:

Parameter	Typ	Art	Bereich	Beschreibung
Port	IN 0	INT	Gültige Port Nr.	Schnittstellennummer
DTR	IN 1	BOOL	TRUE / FALSE	Data Transmit Receive Signal steuern
DCD	OUT 2	BOOL	TRUE / FALSE	Data Carrier Detect Signal abfragen
DSR	OUT 3	BOOL	TRUE / FALSE	Data Set Ready Signal abfragen
RetVal	RET_ VAL	INT	-2...0	Fehlercode: 0: Kein Fehler -1: Ungültige Schnittstellennummer -2: Gewählte Schnittstelle unterstützt keine Modemsignale

Programmierbeispiel:

```
Netzwerk 1: Erweiterter Zugang zu den Modemsignalen

      CALL  SFC  344
       IN0    :=2                      // Schnittstelle 2
       IN1    :=M1.0                   // DTR steuern
       OUT2   :=M1.1                   // Zustand des DCD Signal
       OUT3   :=M1.2                   // Zustand des DSR Signals
       RET_VAL:=MW240                  // Fehlermeldung
```

4.3.7 Serielle Schnittstelle initialisieren (CP441) (SFC 245 B_INIT)

Mit dem SFC 245 können die Schnittstellen 0 bis 6 initialisiert werden (CP441). Die Initialisierung steht im Zusammenhang mit den Kommunikationsbausteinen SFB 12, SFB 13 und SFB 14, die auch in der Standard-Bibliothek von der Step7®-Software zu finden sind. Die weiteren Zusammenhänge können dem Handbuch 26/794 Serielle Kommunikation entnommen werden. In der Regel wird der SFC nur einmal vor Beginn der seriellen Kommunikation aufgerufen, z.B. im OB100.

Es ist möglich den Schnittstellentreiber oder das Protokoll während der Ausführung umzuschalten, dabei muss folgendes beachtet werden:

- Die Puffergrösse kann nicht verändert werden
- Die Sende und Empfangsbausteine benötigen eine positive Flanke am Enable-Eingang, um den neuen Modus zu erkennen.

Die folgende Tabellen bezieht sich beispielhaft auf den Transparent Modus. Näheres zur weiteren Beschreibung der Parameter und Werte können dem Handbuch 26/794 Serielle Kommunikation entnommen werden.

Parameter des SFC 245:

Parameter	Typ	Art	Bereich	Beschreibung
COM_NR	IN 0	INT	Gültige Port Nr.	Gibt an, welche Schnittstelle initialisiert werden soll. Erlaubte Werte: PCD2.M487 Port 0 bis 6 PCD2.M177 Port 1 bis 5. PCD1 und PCD2.M127/157/257 Port 1 bis 3 PCD3.Mxxx7 Port 0 bis 2.
MODE	IN 1	INT	0...8	0=Transparent mode 1=DK3964 2=DK3964R 3=RK512 - DK3964 4=RK512 - DK3964R (Multi-Point wird auf der PCD2.M487 und der PCD3.Mxxx7 nicht unterstützt) 5=ASCII - feste Länge 6=ASCII - 1 Endzeichen 7=ASCII - 2 Endzeichen 8=ASCII - ZVZ (0...8 siehe Handbuch 26/794 Serielle Kommunikation!)
BAUDRATE	IN 2	DINT	Gültige Baudrate	Erlaubte Baudraten: 300, 600, 1'200, 2'400, 4'800, 9'600, 19'200, 38'400 nur COM1, **PCD2.M487**: Baudraten <1'200 Baud werden nicht unterstützt. Port 0 und 6 unterstützen 115'000 Baud. **PCD3.Mxxx7**: Baudraten <1'200 Baud werden nicht unterstützt. Port 0 bis 2 unterstützen 115'000 Baud.
DATA_BIT	IN 3	INT	7…8	Anzahl Datenbits
STOP_BIT	IN 4	INT	1…2	Anzahl Stopbits
PARITY	IN 5	INT	0…4	0=Keine, 1=Gerade, 2=Ungerade, 3=Force Low, 4=Force High 5=Multi-Point (Nur bei Mode=4). Wird auf der PCD2.M487 und PCD3.Mxxx7 nicht unterstützt
CONTROL	IN 6	INT	0…3	Schnittstellentyp: 0=RS-232, 1=RS-485, 2=RS-422, 3=TTY
XON	IN 7	BYTE	0…FFh	Zeichen XON (Default 0x11)
XOFF	IN 8	BYTE	0…FFh	Zeichen XOFF (Default 0x13)
WAIT_ SEND	IN 9	WORD	0… FFFEh	Zeit vom Aktivieren des Kontrollsignals bis gesendet wird (10 ms Schritte)
WAIT_IN-ACTIVITY	IN 10	WORD	0… FFFEh	Zeit vom Ende des Sendens, bis RTS weggenommen wird (10 ms Schritte)
TEL_ COUNT	IN 11	INT	1	Anzahl Telegramme im Puffer

OVER_ WRITE	IN 12	BOOL	FALSE / TRUE	FALSE: Telegramm nicht überschreibbar TRUE: Telegramm überschreibbar, aber nur wenn TEL_COUNT = 1
DEL_RX_ BUFFER	IN 13	BOOL	FALSE / TRUE	Der Empfangspuffer wird im Anlauf gelöscht
DK_PRI- ORITY	IN 14	BOOL	FALSE / TRUE	Nur DK3964R/RK512: Priorität des DK-Treibers (TRUE: hohe Priorität)
ZVZ	IN 15	WORD	0… FFFEh	Zeichenverzugszeit nach der ein Telegrammende erkannt wird: Die ZVZ sollte mindestens 4-fache Übertragungs-zeit eines Zeichens betragen. 0=Default 220 ms Bei Mode 1 bis 4: Zeichenverzugszeit in 10 ms Schritten Bei Mode 5 bis 8: Zeichenverzugszeit in 1 ms Schritten
QVZ	IN 16	WORD	0… FFFEh	Quittungsverzugszeit in 10 ms Schritten 0 = Default Bei Mode 1 und 3 Default 550 ms Bei Mode 2 und 4 Default 2000 ms
TRY_CON- NECT	IN 17	INT	0…255	Anzahl Versuche für den Verbindungsaufbau 0 = 6 Versuche (Default)
TRY_SEND	IN 18	INT	0…255	Anzahl Sendversuche bei Fehler 0 = 6 Versuche (Default)
FIXED_ LENGTH	IN 19	INT	1…1024	Telegrammlänge
END_ CHAR1	IN 20	BYTE	0…255	Erstes Ende Zeichen eines Telegrammes (Nur MODE 6)
END_ CHAR2	IN 21	BYTE	0…255	Zweites Ende Zeichen eines Telegrammes (Nur MODE 6 und 7)
SEND_ Buffer	IN 22	INT	0…4000	Grösse des Sendepuffers. Abhängig von der Grösse des Sendetelegramms.
RCV_Buffer	IN 23	INT	0…4000	Grösse des Empfangspuffers. Abhängig von der Grösse des Empfangstelegramms.
Dummy_I1	IN 24	INT	0	Nicht verwendet
Dummy_ W1	IN 25	WORD	0	Nicht verwendet (Nur RK 512MP)
Dummy_ W2	IN 26	WORD	0	Nicht verwendet (Nur RK 512MP)
Dummy_ DW1	IN 27	DWORD	0	Nicht verwendet (Nur RK 512MP)

RetVal	OUT	INT	-13...0	Fehlercode: 0: Initialisierung war erfolgreich -1: Ungültige Schnittstellennummer -2: Nicht genügend S7-Speicher, um die Puffer zu erzeugen. Evtl. Komprimierung durchführen. -3: Nicht genügend S7-Speicher, um die Puffer zu erzeugen nach durchgeführter Komprimierung -4: Ungültiger Wert für den Parameter MODE -5: Ungültige Schnittstellenparameter (Baudrate, Datenbit, Stopbit oder Parität) -6: Ungültiger Wert für Parameter WAIT_SEND oder WAIT_INACTIVITY -7: Ungültiger Wert für Parameter TEL_COUNT -8: Ungültiger Wert für Parameter ZVZ oder QVZ -9: Ungültiger Wert für Parameter TRY_CONNECT oder TRY_SEND -10: ASCII - Fixed length: Die Telegrammlänge ist länger als die Empfangspuffergrösse -11: Ungültiger Wert für Parameter in SEND_BUFFER oder RCV_BUFFER -12: Die Gesamtspeichergrösse des Sende- und Empfangspuffer ist grösser als die maximal erlaubten 64kB. -13: Der SFC wurde mit einer unterschiedlichen Summe aus Empfangs- und Sendepuffer aufgerufen als beim letzten Aufruf.

Nicht erforderlich Erforderlich

Programmierbeispiel:

```
Netzwerk 1: Initialisierung Serielle Schnittstelle 1, Transparent Modus

      CALL  "B_INIT"
       COM_NR        :=1                 // Schnittstelle Nr. 1
       MODE          :=0                 // Transparent Modus
       BAUDRATE      :=L#9600            // 9600 Baud
       DATA_BIT      :=8                 // 8 Databit
       STOP_BIT      :=1                 // 1 Stopbit
       PARITY        :=0                 // Keine Parität
       CONTROL       :=0                 // RS-232
       XON           :=B#16#0            // Nicht verwendet
       XOFF          :=B#16#0            // Nicht verwendet
       WAIT_SEND     :=W#16#0            // Nicht verwendet
       WAIT_INACTIV :=W#16#0             // Nicht verwendet
       TEL_COUNT     :=1                 // Anzahl Telegramme im Puffer
       OVER_WRITE    :=FALSE             // Telegramm nicht überschreibbar
       DEL_RX_BUFFER:=FALSE              // Nicht verwendet
       DK_PRIORITY   :=FALSE             // Nicht verwendet
       ZVZ           :=W#16#0            // Nicht verwendet
       QVZ           :=W#16#0            // Nicht verwendet
       TRY_CONNECT   :=0                 // Nicht verwendet
       TRY_SEND      :=0                 // Nicht verwendet
       FIXED_LENGTH :=0                  // Nicht verwendet
       END_CHAR1     :=B#16#0            // Nicht verwendet
       END_CHAR2     :=B#16#0            // Nicht verwendet
       SEND_BUFFER   :=300               // Grösse des Sendepuffers
       RCV_BUFFER    :=300               // Grösse des Empfangspuffers
       Dummy_I1      :=0                 // Nicht verwendet
       Dummy_W1      :=W#16#0            // Nicht verwendet
       Dummy_W2      :=W#16#0            // Nicht verwendet
       Dummy_DW1     :=DW#16#0           // Nicht verwendet
       RET_VAL       :=MW246             // Rückgabewert
```

4.4 CAN Kommunikation

Die Steuerungen der Serie PCD3.M6347 besitzen eine CAN-Anschaltung. Die CAN-Anschaltung wird mit SFB-Aufrufen programmiert.

Die CAN Funktionalität ist eine Layer-2 Schnittstelle. Der Anwender kann unter mehreren Betriebsmodi auswählen, je nach gewünschter Komfort- oder Funktionsstufe. Der Anwender kann auf alle Funktionen über die System-Funktionen zugreifen.

Die CAN-Schnittstelle stellt keine CANopen Funktionen zur Verfügung, jedoch kann über die vorhandene Layer-2 Schnittstelle einfach auf CANopen Slaves zugegriffen werden. Dazu muss der Anwender alle relevanten CANopen Nachrichten selbst in das Anwenderprogramm über die Layer-2 Schnittstelle eingeben.

4.4.1 Übersicht

CAN Layer-2
Die CAN Steuerung auf der SPS Erweiterung enthält den CAN Layer-2 Zugriff auf den Bus. Die gesamte Layer-2 Protokoll Abarbeitung erfolgt durch die Steuerung. Der Anwender schickt an die Steuerung eine Nachricht mit dem festgelegten Sendezeitpunkt. Ausserdem kann er festlegen, welche Nachrichten von der Steuerung empfangen und dann nach dem Empfang aus dem Speicher ausgelesen werden sollen.

Betriebsmodi

Die CAN Steuerung enthält mehrere Puffer (Message Objects), die als Übertragungs- oder Empfangspuffer für jegliche Nachrichten-ID konfiguriert werden können. Der Anwender kann mit drei verschiedenen Zugriffstypen die Steuerungsfunktionen ansprechen:

- **CAN Direkt Zugriff** (FullCAN): Direkter Zugriff zu allen 32 Puffern. Der Anwender hat zu allen 32 Puffern unabhängigen Zugriff. Die Schnittstelle erlaubt alle Funktionen, die die CAN Steuerung enthält. Dieser Modus ist analog dem FullCAN Prinzip, das die Steuerung enthält.
- **CAN Basisdienste** (BasicCAN): eine Übertragungs- und eine Empfangswarteschlange erlauben eine einfache Behandlung der CAN Kommunikation im Anwenderprogramm. Es können mehrere Nachrichten, ohne die Bestätigung für jede Nachricht abzuwarten, an die Warteschlange gesendet werden. Die Empfangswarteschlange stellt sicher, dass keine Nachricht verloren geht (vorausgesetzt, die Tiefe der Warteschlange ist entsprechend festgelegt). Die Anwenderschnittstelle ist gegenüber der CAN Direkt Zugriffs-Schnittstelle beträchtlich vereinfacht. Dieser Modus ist analog dem BasicCAN Prinzip, das in verschiedenen einfachen CAN Steuerungen mit nur einem Empfangs- und einem Übertragungspfad angewandt wird.
- **CAN Daten Abbildung**: Die Daten Abbildung vereinfacht und automatisiert den zyklischen Austausch der Prozessdaten. Dieses Abbilden wird zu Beginn mit den entsprechenden Nachrichten IDs konfiguriert. Die Nachrichtendaten werden direkt als Prozessdaten in der SPS abgebildet. Alle abgehenden Nachrichten werden durch den Datenmanager während eines speziellen Intervals automatisch gesendet. Die empfangenen Nachrichten sammelt der Manager für die Prozessdaten.

Diese drei Modi können gleichzeitig, allerdings mit gewissen Einschränkungen bezüglich der Nachrichten ID-Bereiche, die den einzelnen Modi zugeordnet sind, angewandt werden.

Die Bausteine sind hier nur der Vollständigkeit halber erwähnt. Weitere Informationen bekommen Sie über mail an:support@saia-pcd.com, oder im Handbuch “CAN Kommunikation xx7, 26/840”.

4.4.2 CAN-SFBs

Die Step7 CAN Schnittstelle beinhaltet mehrere SFBs für die Konfiguration, die Datenübertragung und den Status.

Generelle Funktionen
SFB403: CAN_CFG, CAN-Driver konfigurieren

SFB404: CAN_STAT, Abfrage des gegenwärtigen Status des CAN-Drivers

Diese Funktionen stellen die Basis-Konfiguration und die Status-Einrichtung, die in jedem Betriebsmodus benötigt werden bereit.

CAN Direkt Zugriff
SFB402: CAN_CFGO, Nachrichten-Objekte konfigurieren

SFB401: CAN_TX, Nachrichten senden

SFB400: CAN_RX, Nachrichten empfangen

Diese Funktionen erlauben direkten und flexiblen Zugriff auf die Hardware der CAN Steuerung. Dieser Modus entspricht dem in der Steuerung enthaltenen FullCAN Prinzip.

CAN Basisdienste
SFB412: CAN_CFGQ, Warteschlangen konfigurieren

SFB411: CAN_TXQ, Nachrichten an die Warteschlange senden

SFB410: CAN_RXQ, Nachrichten aus der Warteschlange empfangen

Diese Funktionen erlauben zusätzliche Hardware Eingriffe: z. B. wenn der direkte Hardware Zugriff mittels Warteschlangen bearbeitet wird und der Anwender die low-level CAN Abhandlung nicht weiter beachten muss. Dieses Konzept entspricht dem BasicCAN Prinzip, das einige CAN Steuerungen enthalten.

CAN Daten Abbildung
SFB413: CAN_DMAP, Daten Abbildung konfigurieren

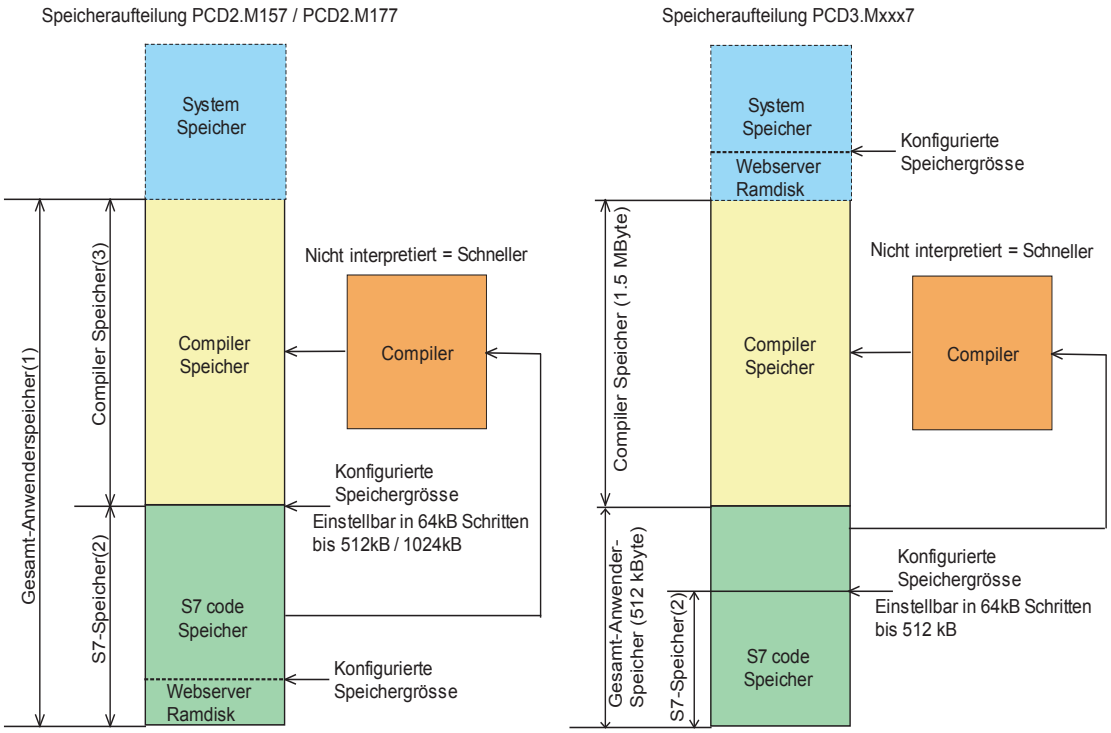
5 Speicher

5.1 Allgemeines

Die Grösse des S7 Code Speichers ist vom Saia PCD® Typ abhängig: Während bei einigen PCD-Typen diese Grösse fix ist, kann die Grösse des S7 Code Speichers bei anderen Typen in Schritten von 64 kB bis zu einem Maximalwert mittels CDB (Configuration Data Block) frei konfiguriert werden. Die nachfolgende Tabelle gibt einen Überblick.

PCD-Typ	S7 Code Speichergrösse
PCD1.M137	Default: 64 kByte. Von 64 kB bis 128 kB konfigurierbar.
PCD2.M127	Fix 132 kByte
PCD2.Mx57	Default: 256 kByte. Von 64 kB bis 512 kB konfigurierbar.
PCD2.M177	Default: 512 kByte. Von 64 kB bis 1024 kByte konfigurierbar.
PCD2.M487	Default: 1 MByte. Von 64 kB bis 1024 kByte konfigurierbar.
PCD3.Mxxx7	Default: 512 kByte Von 64 kB bis 512 kByte konfigurierbar.

Wird bei den Steuerungen des Typs PCD2.Mx57 und PCD2.M177 der S7 Code Speicher nicht auf die maximal einstellbare Grösse konfiguriert, so wird der nicht beanspruchte Bereich dem Compiler zur Verfügung gestellt. Bei den Steuerungen des Typs PCD1 und PCD2.M487 und PCD3.Mxxx7 bleibt der nicht beanspruchte Speicher unbenutzt.



Die Steuerungen PCD2.Mx57, M177 und PCD3.Mxxx7 besitzen die Fähigkeit, das Anwenderprogramm zu kompilieren. Im Gegensatz zum Interpretieren, bei dem jede Befehlszeile während der Laufzeit beim Durchlaufen von neuem übersetzt wird,

findet beim Kompilieren vor dem zyklischen Abarbeiten des Programms dessen Übersetzung in “CPU-Sprache” statt. Dadurch, dass das Übersetzen jeder einzelnen Programmzeile nun wegfällt, wird dasselbe Programm je nach Programmstruktur und verwendeten Befehlen im Schnitt 2 bis 3 mal schneller abgearbeitet. Die Übersetzung benötigt ihren eigenen, den Compiler-Speicherbereich, welcher sich im oben erwähn-ten Bereich des RAM‘s befindet.

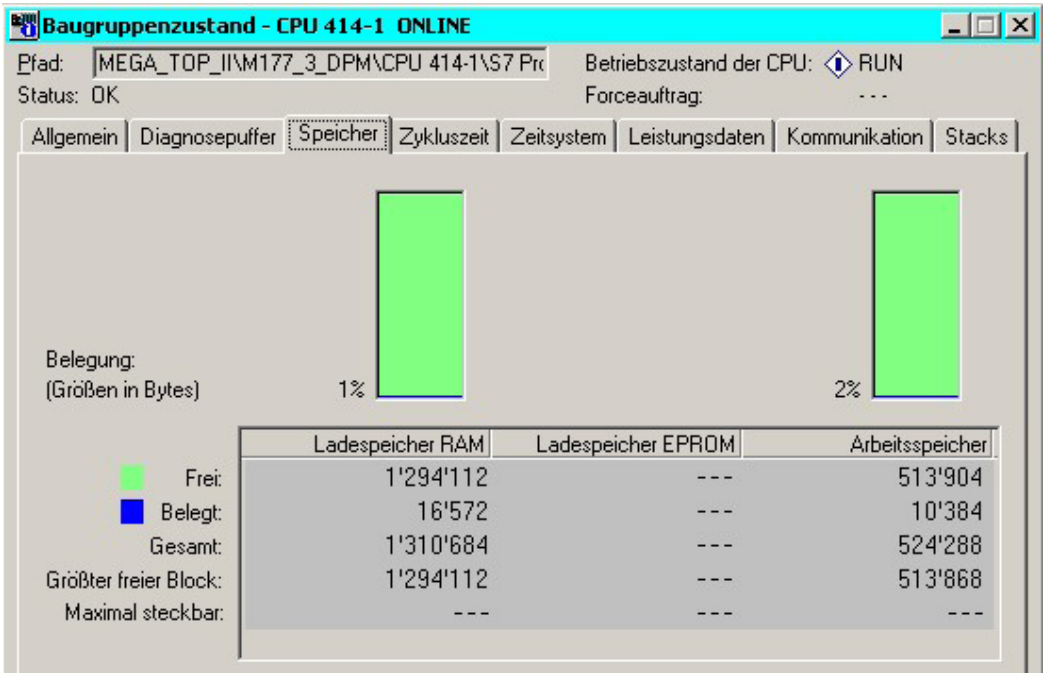
Im Kapitel Systemkonfiguration wird beschrieben, wie die Grenze zwischen Anwender- und Compiler-Bereich mit Hilfe eines Eintrages im CDB definiert wird.

Eine Step7-Anweisung kann zu bis zu 10 mal grösserem Compilercode führen. Insbesondere bei grossen Programmen kann es daher vorkommen, dass nicht alle Bausteine kompiliert werden können. Nicht kompilierte Bausteine werden in diesem Fall interpretiert.

Mittels Priorisierung der Bausteine kann man die Zykluszeit der Saia PCD® weiter optimieren. Man erreicht dies damit, indem man vor allem oft aufgerufene Bausteine kompiliert. Die Priorisierung wird ebenfalls im CDB parametriert.

Wird der Web-Server aktiviert, so wird der notwendige Speicher für die RAM Disk bei der PCD2.M157 und M177 aus dem S7-Code Speicher benutzt. D.h, der verfügbare Speicher für die Step7-Bausteine wird um die Web-Server RamDisk Grösse kleiner. Auf der PCD2.M487 und der PCD3.Mxxx7 ist für die RamDisk ein eigener Speicherbereich reserviert. Die Grösse der RamDisk kann mittels CDB-Eintrag festgelegt werden. Weitere Informationen findet man im Kapitel Web-Server.

Anzeige des verfügbaren und belegten Speichers im online Baugruppenzustand:
Im nachfolgenden Screen Shot wird die Speicherbelegung am Beispiel einer M177 dargestellt.



Die Einträge haben für die PCD2.M157 / M177 / PCD3 folgende Bedeutung:

Ladespeicher	Frei	Gesamter Anwenderspeicher - belegter Kompilerspeicher. Der freie verfügbare Speicher für den Kompiler = Dieser Wert - konfigurierter S7-Code Speicher
	Belegt	Durch den Kompiler belegten Speicher
	Gesamt	gesamter zur Verfügung stehender Anwenderspeicher
Arbeitsspeicher	Frei	Noch freier S7-Code Speicher
	Belegt	Durch S7-Bausteine belegter Speicher
	Gesamt	Projektierter S7-Code Speicher - WebServer RamDiskgrösse

Die % Anzeige des Ladespeichers kann bei einer PCD2.M157/M177/PCD3 nie 100% erreichen.

5.2 Flash Backup Funktionen

5.2.1 Anwenderprogramm Backup

Flash EPROMs auf einer xx7-Steuerung können mit der Funktion “RAM nach ROM kopieren...“ im SIMATIC®-Manager-Menu “Zielsystem PLC“ als Backup-Speicher für das Anwenderprogramm verwendet werden. Die Grösse des Backup-Speicher-bereichs im Flash EPROM entspricht derjenigen des Anwender-Speicherbereichs im RAM.

5.2.2 Datenbaustein Speicher

Den ungenutzten Speicherbereich des Flash-EPROM kann man zusätzlich als Da-tenbausteinspeicher verwenden. Dieser ist in zwei gleich grosse Speicherbere-iche aufgeteilt. Die Grösse des Datenbausteinspeichers ist von der Grösse des eingestellten Anwenderspeichers abhängig. Die Formel zur Berechnung eines Daten-baustein-speicherbereichs lautet folgendermassen:

DB-Speicherbereich = (Grösse des Flash - Grösse des Anwenderspeichers) / 2

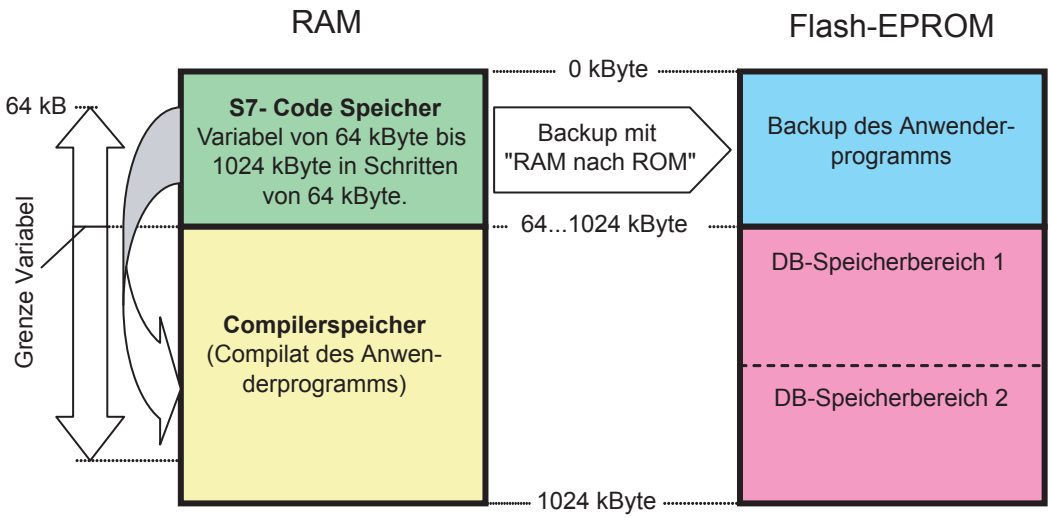
Nachfolgende Tabelle zeigt abhängig vom Saia PCD® Typ die Grösse der Daten-baustein-speicherbereich im Flash-Eprom.

PCD-Typ / Flash-Typ	Flash-Eprom Grösse	Grösse DB-Speicher
PCD1.M137	128 kByte	0 bis 32 kByte, in 32 kByte Schritten
PCD2.M127	512 kByte	190 kByte fix
PCD2.Mx57	512 kByte	0 bis 224 kByte, in 32 kByte Schritten
PCD2.M177 / PCD7.R400	1 MByte	0 bis 480 kByte, in 32 kByte Schritten
PCD2.M487 / PCD7.R400	1 MByte	0 bis 480 kByte, in 32 kByte Schritten
PCD3.Mxxx7[1)] / PCD7.R500	1 MByte	512 bis 992 kbyte, in 32 kByte Schritten

[1)] Der Backup Speicher sowie die DB Flash Funktionalität steht nur auf dem Slot M1 zur Verfügung.

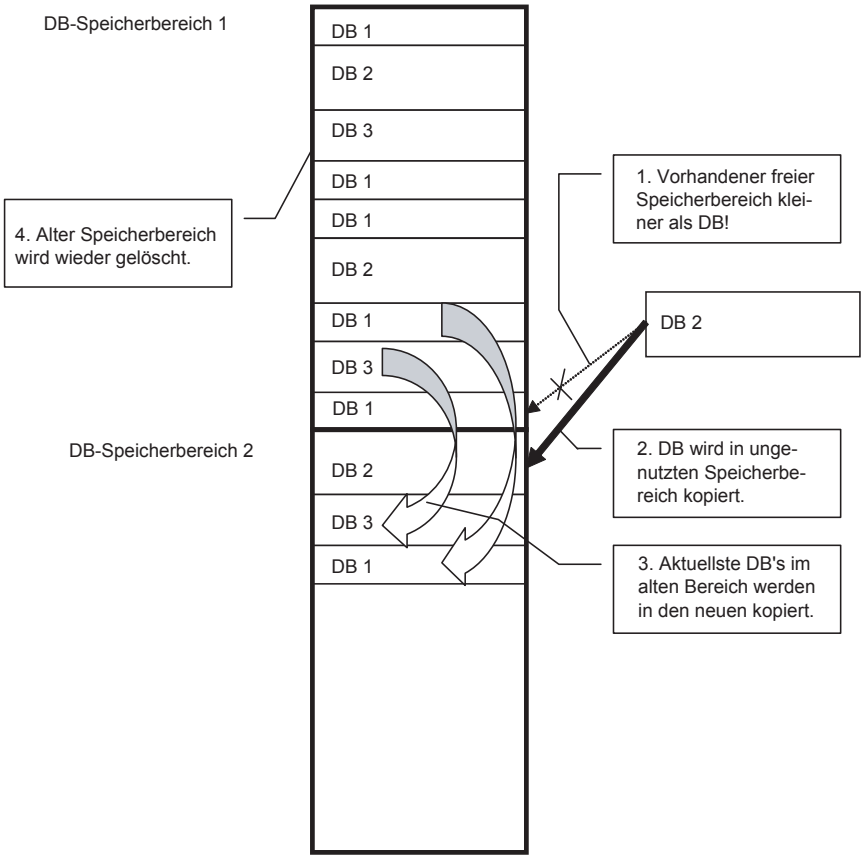
Wird der S7-Code Speicher auf die maximal einstellbare Grösse konfiguriert, so steht kein DB-Speicher im Flash zur Verfügung. (Gilt nicht für die PCD3!)

Speicheraufteilung am Beispiel einer PCD2.M177:



Der DB-Speicherbereich muss aufgeteilt werden, um ein “Überfüllen“ des Flash-EPROMs zu verhindern. Dies würde relativ schnell geschehen, weil alte Versionen der DBs im Flash-EPROM nicht gelöscht, sondern als ungültig markiert werden.

Die DBs werden bei jedem Schreibbefehl kontinuierlich in einen DB-Bereich geschrieben. Dies gilt auch dann, wenn der gleiche DB mehrere Male ins Flash geschrieben wird. In diesem Fall ist der aktuellste DB der gültige.



Ist ein DB grösser als der freie zur Verfügung stehende Speicherbereich, wird der DB zusammen mit den aktuellen DBs in den zweiten DB-Bereich kopiert. Anschlies-

send wird der volle DB-Bereich mit den alten DBs gelöscht, und somit wird Speicher freigegeben.

Natürlich ist es immer noch möglich, das Flash-EPROM so zu füllen, dass kein Speicher mehr freigegeben werden kann. Für diesen Fall wurde eine Funktion vorgesehen, um beide DB-Speicherbereiche löschen zu können.

Die Funktionen “Speichern“, “Lesen“ und “Löschen“ sind im SFB 240 implementiert:

Parameter des SFB 240:

Parameter	Typ	Art	Bereich	Beschreibung
REQ	IN 0	BOOL		Start der Bearbeitung bei positiver Flanke
FUNCTION	IN 1	BYTE	1...3	Funktionsnummer: B#16#1 : DB in Flash sichern B#16#2 : DB wiederherstellen B#16#3 : DB-Bereiche löschen
DB_NO	IN 2	INT		Nummer des Datenbausteines zum Sichern und Wiederherstellen
DONE	OUT 3	BOOL		Bearbeitung erfolgreich beendet
ERROR	OUT 4	BOOL		Fehler während der Bearbeitung
STATUS	OUT 5	WORD		Status-/ Fehlerinformation 1: Eine Flash-Funktion ist schon im Gange 2: Unbekannte Funktion angefordert. Funktionsnummer liegt ausserhalb des gültigen Bereichs 1...3 3: DB kann nicht gefunden werden. Zu sichernder DB ist nicht im Anwenderspeicher des RAM, bzw. der wiederherzustellende DB ist nicht im Datenbausteinspeicherbereich des Flash-EPROM vorhanden. 5: Flash-EPROM ist voll. Keine zusätzliche DB-Sicherung mehr möglich, solange die Datenbausteinspeicherbereiche nicht gelöscht (Funktion 3) werden. 2F: Fehler beim Abspeichern (z.B. nicht genügend Speicher→komprimieren) FF: Flash-EPROM nicht vorhanden oder defekt AA: Flash-EPROM-Daten korrupt. DB-Struktur ist inkorrekt (ungültige Grösse, falscher Header-Code, etc.) F1: Schreiben. Kein Fehler→Zustandsanzeige des SFB 240 Prozesses F2: Lesen. Kein Fehler→Zustandsanzeige des SFB 240 Prozesses F3: Löschen. Kein Fehler→Zustandsanzeige des SFB 240 Prozesses

5.3 Dateisystem

5.3.1 Allgemein

Die PCD3 Familie stellt verschiedene Dateisysteme zur Verfügung. Auf diese Datei-systeme kann man mit Hilfe der System Funktionen (SFB 450 bis SFB463) vom Anwenderprogramm aus zugreifen. Weiter ist es möglich über FTP / Web-Server auf die Dateisysteme zuzugreifen.

Die Funktionalität des FTP Servers wird im Handbuch “26-791_Ethernet_xx7” beschrieben.

Mit den zusätzlichen steckbaren Flash Modulen steht ein Flash Filesystem zur Verfügung. Zur Zeit existieren folgende Flash Module:

Flash-Typ	Flash Grösse	Steckplatz	Flash Filesystem Name
PCD7.R550M04	4 MByte	M1 und M2	M1_Flash M2_Flash
PCD3.R550M04	4 MByte	IO-Steckplatz 0 bis 3	SL0Flash SL1Flash SL2Flash SL3Flash
PCD3.R600	SD Device 128MB to 1GB	IO-Steckplatz 0 bis 3	SL0Flash SL1Flash SL2Flash SL3Flash
PCD3 mit 4 MB internem Flash *)	1MByte	intern	INTFLASH
PCD3 mit 4MB SRAM *)	1MByte	intern (RAM)	USERSRAM

*) Diese Versionen sind nur als OEM-Plattformen erhältlich.

Standardmässig werden die PCD7.R550M04 formatiert vertrieben. Sobald sie im M1 oder M2 Slot des CPU-Gehäuses eingesteckt werden, sind sie betriebsbereit.

Sie sind organisiert nach Seiten (256 bytes), Blöcken (2KB) und Sektoren (64KB). Eine Seite ist die Speichergrösse fürs Schreiben, ein Block bildet eine Datei und ein Sektor ist die Speichergrösse, die gelöscht werden kann.

Das PCD7.R550M04 Flash selbst ist 4MB gross. Allerdings sind 64 KB reserviert und einige Blöcke werden für die Speicherung des internen Dateisystem benötigt. Das heisst, dass dem Anwender geringfügig weniger als der gesamte Flash Speicher zur Verfügung steht.

Wird eine Datei angelegt, wird dafür ein Block (2KB) reserviert. Auch wenn die Datei kleiner ist, werden diese 2KB reserviert. Sind die 2KB gefüllt, wird ein neuer Block hinzugefügt und die Grösse dieser Dateieinheit ist nun 4KB. Der physikalische Platz, den eine Datei beansprucht ist 2KB * ((mod 2KB der Dateigrösse) + 1).

Wird eine Datei gelöscht, werden die durch sie benutzten Blöcke als frei markiert. Es ist jedoch nicht möglich, sofort neue Daten einzuschreiben. Zuerst müssen alle Bits des Blocks auf “1” gesetzt werden. Diese Operation ist komplex und weder automatisch noch durch das Anwenderprogramm durchführbar. Während dieser Opera-

tion ist das Flash beschäftigt und kein Zugriff (lesen / schreiben / auflisten) möglich. Während dieser “Kompression” wird der Sektor zuerst in einen reservierten Sektor kopiert, der aktuell komprimierte Sektor wird gelöscht und nur die gerade benutzten Blöcke werden vom reservierten Sektor zurückkopiert. Am Ende der Operation wird auch der reservierte Sektor gelöscht. Diese Abfolge von Operationen wird für alle Sektoren des Flashs durchgeführt. Bei einem 4 MB Flash sind dabei 63 Sektoren betroffen.

Zieht man in Betracht, dass ein Sektor in 2 Sekunden gelöscht wird (Herstelle-ran-gabe), benötigt die Kompression aller Sektoren eines Flashs im Maximum etwa 400 Sekunden (mehr als 6 Minuten). Während dieser Zeit ist das Flash beschäftigt. Manche Sektoren müssen nicht komprimiert werden, weil in diesen keine Blöcke ausgelöst wurden. Dies reduziert die Kompressionszeit beträchtlich.

Die PCD7.R550M04 ist während des Betriebs steckbar. Das heisst, sie kann jederzeit während der PLC Operationen eingesteckt werden. Unmittelbar nach dem Einstecken wird das Dateisystem aufgebaut und falls notwendig die Kompression durchgeführt. Dies bedeutet, dass während des Einsteckens das Flash für den Anwender sichtbar ist, es ist jedoch genausogut möglich, dass das Flash beschäftigt meldet.

Obwohl das PCD7.R550M04 prinzipiell im Betrieb ausgesteckt werden kann, ist dies mit Vorsicht zu betrachten. Das Flash hat keine LED, die anzeigt, ob auf das Flash gerade zugegriffen wird oder nicht.

- Zwei Fälle sollten beim Ausstecken des Flashs betrachtet werden: Einige Schreib-Operationen laufen ständig. Diese Operationen werden entweder durch das Anwenderprogramm oder durch FTP ausgelöst. Vor dem Ausstecken des Flashs, die PLC auf STOP setzen, falls das Anwenderprogramm in das Flash schreibt und sicherstellen, dass keine FTP Verbindung besteht.

- Während des Ablaufs der Kompression: Diese Operation wird entweder durch das Anwenderprogramm angestossen (direkt oder indirekt durch Löschen von Dateien) oder auch indirekt durch FTP nach dem Löschen von Dateien. Diese Aktionen sind durch den Anwender schwer zu erkennen.

Aus diesen Gründen, und weil das Ausstecken der Flashs nicht ausdrücklich untersagt ist, wird das Ausstecken der Flashs während des Betriebs ***nicht*** empfohlen.

Vorsicht ist ebenso geboten, wenn die PLC ausgeschaltet wird. Führt das Anwenderprogramm gerade Aufrufe zum Flash Dateisystem aus (Schreiben oder Kompri-mieren) oder ein FTP-Client greift gerade auf das Flash zu, während die PLC abgeschaltet wird, reicht die Datenrettungszeit eventuell nicht aus. Zu diesem Zeitpunkt wird ein spezieller Algorithmus angestossen, um so viele Daten wie möglich zu retten, es ist aber möglich, dass die zuletzt geschriebenen Daten / Dateien verloren gehen.

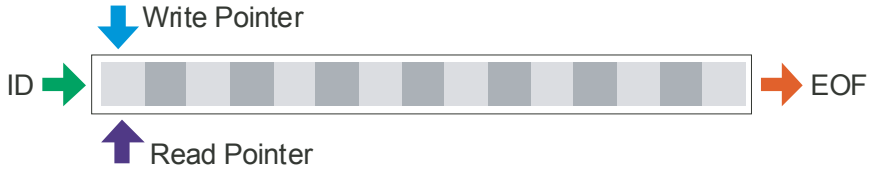
Für die Benützung der Dateisystemfunktionen sind folgende Schritte notwendig:

1) Erzeugen und Öffnen einer Datei.
2) Optional: den Lese/Schreibzeiger in der Datei positionieren
3) Schreiben der Daten von den PLC Medien in die Datei
4) Lesen der Daten von der Datei in die PLC Medien
5) Schliessen der Datei

Zusätzlich stehen noch Funktionen für das Löschen von Dateien, sowie Funktionen

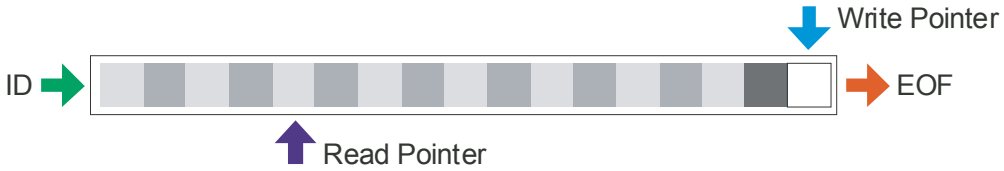
für das Bearbeiten von Verzeichnissen zur Verfügung.
Beim Öffnen (Erzeugen) einer Datei, wird dem Anwender eine FileId (DateiId) zurückgegeben. Dieser FileId sind zusätzlich zwei Dateizeiger zugeordnet:

1) Lesezeiger (Write Pointer)
2) Schreibzeiger (Read Pointer)



Diese Zeiger werden durch die Lese-, Schreib- und Such-Funktionen modifiziert.

Beispiel: Nach dem Öffnen einer Datei werden 4 Byte gelesen und 1 Byte am Ende der Datei geschrieben. Die Dateizeiger haben dann folgende Positionen:



In den nachfolgenden Kapiteln werden die System Funktionsbausteine für die Dateizugriffe beschrieben.

Für die nachfolgenden Systemfunktionen existiert eine Bibliothek. Diese können von der Support-Hompage www.sbc-support.com heruntergeladen werden.

5.3.2 “FCREATE” SFB450

Mit dem SFB FCREATE wird eine neue Datei erzeugt. Über den Parameter FileName wird festgelegt, in welchem Flash Filesystem sowie in welchem Verzeichnis die Datei abgelegt wird. Der Dateiname muss immer als absoluter Dateiname angegeben werden

Parameter des SFB 450:

Parameter	Typ	Art	Beschreibung
FileName	IN 0	STRING[64]	Der Parameter besteht aus dem Flash Filesystem Namen, dem Verzeichnisnamen sowie aus dem eigentlichen Dateinamen. Für den Dateinamen können nur alphanumerische Zeichen (ohne Space) sowie “.” verwendet werden. Ein Dateiname kann maximal 24 Zeichen lang sein. Die maximale Länge des absoluten Dateinamens darf maximal 64 Zeichen lang sein. Beispiel: `M1_Flash:/Report.txt`
GroupID	IN 1	BYTE	Definiert zu welcher Gruppe die erzeugte Datei gehört. Folgende 4 Gruppen sind definiert: 0x10 USER Gruppe 1 0x20 USER Gruppe 2 0x40 USER Gruppe 3 0x80 USER Gruppe 4
GroupAccess	IN 2	BYTE	Mit diesem Parameter können Zugriffsgruppen definiert werden. Für die Definition kann eine Kombination der GroupIds verwendet werden. Das Erzeugen oder Löschen einer Datei oder eines Unterverzeichnisses ist nur in einem Verzeichnis möglich, welches zu einer entsprechenden Gruppe gehört. Mit dem Wert “0” ist der Zugriff auf alle Gruppen erlaubt.
Handle	OUT 3	DINT	Wenn die Datei erzeugt werden konnte, steht hier die FileId. Diese FileId ist gültig bis die Datei mit FCLOSE wieder geschlossen wird. Steht hier eine negative Zahl, konnte die Datei nicht erzeugt werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Erzeugen eines Files

      CALL  #FCREATE, DB450
       FileName   :="FileNames".M1Flash_DataFile
       GroupID    :=#GroupID
       GroupAccess:=#AccessLevel
       Handle     :=#FileHandle
      L     #FileHandle
      L     0
      >=D
      SPB   next
      SPA   erro
```


5.3.3 "DCREATE" SFB451

Mit dem SFB DCREATE wird ein neues Verzeichnis erzeugt. Über den Parameter FileName wird festgelegt in welchem Flash Filesystem das Verzeichnis angelegt wird. Der Verzeichnisname muss immer als absoluter Name angegeben werden.

Parameter des SFB 451:

Parameter	Typ	Art	Beschreibung
FileName	IN 0	STRING[64]	Der Parameter besteht aus dem Flash Filesystem Namen, und dem Verzeichnisnamen. Für den Verzeichnisnamen können nur alphanumerische Zeichen (ohne Space) verwendet werden. Ein Verzeichnisname kann maximal 24 Zeichen lang sein. Beispiel: `M1_FLASH:/Config`
GroupID	IN 1	BYTE	Definiert zu welcher Gruppe die erzeugte Datei gehört. Folgende 4 Gruppen sind definiert: 0x10 USER Gruppe 1 0x20 USER Gruppe 2 0x40 USER Gruppe 3 0x80 USER Gruppe 4
GroupAccess	IN 2	BYTE	Mit diesem Parameter können Zugriffsgruppen definiert werden. Für die Definition kann eine Kombination der GroupIds verwendet werden. Das Erzeugen oder Löschen einer Datei oder eines Unterverzeichnisses ist nur in einem Verzeichnis möglich, welches zu einer entsprechenden Gruppe gehört. Mit dem Wert "0" ist der Zugriff auf alle Gruppen erlaubt.
RetVal	OUT 3	DINT	Wenn diese Funktion erfolgreich ausgeführt wurde, ist RetVal gleich "0". Steht hier eine negative Zahl, konnte die Funktion nicht korrekt ausgeführt werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Erzeugen eines Verzeichnisses

      CALL  #DCREATE, DB451
       FileName   :="FileNames".M1Flash_Dir3
       GroupID    :=#GroupID
       GroupAccess:=#AccessLevel
       RetVal     :=#ReturnValue
      L     #ReturnValue
      L     L#0
      ==D
      SPB   next
      SPA   erro
```


5.3.4 "FOPEN" SFB452

Mit dem SFB "FOPEN" wird eine Datei geöffnet. Über den Parameter FileName wird festgelegt in welchem Flash Filesystem sowie in welchem Verzeichnis die Datei geöffnet wird. Der Dateinamen muss immer als absoluter Dateiname angegeben werden

Parameter des SFB 452:

Parameter	Typ	Art	Beschreibung
FileName	IN 0	STRING[64]	Der Parameter besteht aus dem Flash Filesystem Namen, dem Verzeichnisnamen sowie aus dem eigentlichen Dateinamen. Für den Dateinamen können nur alphanumerische Zeichen (ohne Space) sowie "." verwendet werden. Ein Dateiname kann maximal 24 Zeichen lang sein. Die maximale Länge des absoluten Dateinamens darf maximal 64 Zeichen lang sein. Beispiel: `M1_FLASH:/Config/Maschine1.txt`
AccessTYPE	IN 1	BYTE	Definiert die Zugriffsrechte auf die zu erzeugende Datei. 0x01 Read only. Die Datei kann nur gelesen werden. Einen Schreibzugriff auf die Datei erzeugt einen Fehler. 0x02 Write only. Die Datei kann nur beschrieben werden. Einen Lesezugriff auf die Datei erzeugt einen Fehler. 0x03 Read/Write. Die Datei kann sowohl lesend wie schreibend bearbeitet werden.
Handle	OUT 2	DINT	Wenn die Datei geöffnet werden konnte, steht hier die FileId. Diese FileId ist gültig bis die Datei mit FCLOSE wieder geschlossen wird. Steht hier eine negative Zahl, konnte die Datei nicht geöffnet werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Oeffnen eines Files

      CALL  #FOPEN, DB452
       FileName  :="FileNames".M1Flash_DataFile
       AccessType:=#AccessType
       Handle    :=#FileHandle
      L     #FileHandle
      L     0
      >=D
      SPB   next
      SPA   erro
```


5.3.5 "FSEEK" SFB453

Mit der Funktion SFB "FSEEK" wird in einer geöffneten Datei der Schreib- oder Lesezeiger manipuliert. Über Handle wird festgelegt, in welcher Datei die Zeiger modifiziert werden. Mit dem Parameter SeekPos wird die neue Position des Zeigers in der Datei angegeben.

Parameter des SFB 453:

Parameter	Typ	Art	Beschreibung
Handle	IN 0	DINT	FileId die beim Öffnen / Erzeugen einer Datei im Parameter Handle zurückgegeben wird. Mit Handle wird die Datei eindeutig identifiziert, die bearbeitet werden soll.
SeekPos	IN 1	DINT	Neue Position der Dateizeiger. Die neue Position wird relativ angegeben. Der Wert "0" setzt beide Dateizeiger (Lesen / Schreiben) an den Anfang des Datei.
AccessType	IN 2	BYTE	Definiert welcher Dateizeiger modifiziert wird: 0x01 Lesezeiger wird modifiziert. 0x02 Schreibzeiger wird modifiziert. 0x03 Lese-/Schreibzeiger werden modifiziert.
RetVal	OUT 3	DINT	Wenn diese Funktion erfolgreich ausgeführt wurde, ist RetVal gleich "0". Steht hier eine negative Zahl, konnte die Funktion nicht korrekt ausgeführt werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Manipulieren der Filezeiger

      L     1                              // read pointer
      T     #AccessType
      L     0                              // Set readpointer to start of File
      T     #SeekPos

      CALL  #FSeek, DB453
       Handle     :=#FileHandle
       SeekPos    :=#SeekPos
       AccessType:=#AccessType
       RetVal     :=#ReturnValue
      L     #ReturnValue
      L     L#0
      ==D
      SPB   next
      SPA   erro
```


5.3.6 “FWRITE” SFB454

Mit der Funktion SFB “FWRITE” werden Daten von einem PLC- Media in eine geöffnete Datei übertragen. Über Handle wird festgelegt, in welche Datei die Daten übertragen werden. Mit dem Parameter Buffer wird ein Any-Pointer auf die zu übertragende Daten der Funktion übergeben.

Parameter des SFB 454:

Parameter	Typ	Art	Beschreibung
Handle	IN 0	DINT	FileId die beim Öffnen / Erzeugen einer Datei im Parameter Handle zurückgegeben wird. Mit Handle wird die Datei eindeutig identifiziert, die bearbeitet werden soll.
WrAttr	IN 1	BYTE	Abhängig von diesem Parameter werden die Daten entweder ans Ende der Datei geschrieben, oder an die Stelle des Write Zeigers. 16 Daten werden an die aktuelle Stelle des Writezeigers geschrieben. D.h. die Daten in der Datei werden überschrieben. Diese Operation ist nur für Dateien im RAM Bereich gültig. Auf Dateien in Flash Einheiten wird eine Fehlermeldung zurückgegeben. 17 Die Daten werden am Ende der Datei angehängt.
Buffer	IN 2	ANY	Die Daten der PLC-Medien werden mit einem ANY-Pointer übergeben. Als PLC- Medien können DBs, Merker, Eingangs- und Ausgangsprozessabbild übergeben werden werden. Die maximale Länge ist auf 256 Bytes begrenzt.
RetVal	OUT 3	DINT	Wenn diese Funktion erfolgreich ausgeführt wurde, ist RetVal gleich “0”. Steht hier eine negative Zahl, konnte die Funktion nicht korrekt ausgeführt werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Daten in ein File schreiben

      L     17                          // Append
      T     #WriteAttribute

      CALL  #FWrite, DB454
       Handle:=#FileHandle
       WrAttr:=#WriteAttribute
       Buffer:="Bloecks".TempBlock[0]
       RetVal:=#ReturnValue
      L     #ReturnValue
      L     L#0
      ==D
      SPB   next
      SPA   erro
```


5.3.7 “FREAD” SFB455

Mit der Funktion SFB “FREAD” werden Daten von einer geöffneten Datei in die PLC-Media übertragen. Über Handle wird festgelegt, von welcher Datei die Daten übertragen werden. Mit dem Parameter Buffer wird ein Any-Pointer auf die zu übertragende Daten der Funktion übergeben.

Parameter des SFB 455:

Parameter	Typ	Art	Beschreibung
Handle	IN 0	DINT	FileId die beim Öffnen / Erzeugen einer Datei im Parameter Handle zurückgegeben wird. Mit Handle wird die Datei eindeutig identifiziert, die bearbeitet werden soll.
Buffer	IN 1	ANY	Die Daten der PLC-Medien werden mit einem ANY-Pointer übergeben. Als PLC- Medien können DBs, Merker, Eingangs- und Ausgangsprozessabbild übergeben werden werden. Die maximale Länge ist auf 256 Bytes begrenzt.
RetVal	OUT 2	DINT	Wenn diese Funktion erfolgreich ausgeführt wurde, ist RetVal gleich der Anzahl Bytes, die in den Puffer übertragen wurden. Steht hier eine negative Zahl, konnte die Funktion nicht korrekt ausgeführt werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Daten von einem File lesen

      CALL  #FRead, DB455
       Handle:=#FileHandle
       Buffer:="Bloecks".TempBlock    // Zeiger auf 256 Byte
       RETVal:=#ReturnValue
      L     #ReturnValue
      L     L#256
      ==D
      SPB   next
      SPA   erro
```

5.3.8 “FLENGTH” SFB456

Mit der Funktion SFB “FLENGTH” wird die Grösse einer geöffneten Datei gelesen.

Parameter des SFB 456:

Parameter	Typ	Art	Beschreibung
Handle	IN 0	DINT	FileId die beim Öffnen / Erzeugen einer Datei im Parameter Handle zurückgegeben wird. Mit Handle wird die Datei eindeutig identifiziert, die bearbeitet werden soll.
RetVal	OUT 1	DINT	Wenn diese Funktion erfolgreich ausgeführt wurde, ist RetVal gleich der Dateigrösse. Steht hier eine negative Zahl, konnte die Funktion nicht korrekt ausgeführt werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Filegroesse lesen

      CALL  #FLENGTH, DB456
       Handle:=#FileHandle
       RetVal:=#ReturnValue
      L     #ReturnValue
      T     #FileGroesse
      L     0
      >D                          // File ist groesser 0
      SPB   next
      SPA   erro
```

5.3.9 “FCLOSE” SFB457

Mit der Funktion SFB “FCLOSE” wird eine geöffnete Datei geschlossen. Konnte die Funktion korrekt ausgeführt werden, so ist die FileId nicht mehr gültig.

Parameter des SFB 457:

Parameter	Typ	Art	Beschreibung
Handle	IN 0	DINT	FileId die beim Öffnen / Erzeugen einer Datei im Parameter Handle zurückgegeben wird. Mit Handle wird die Datei eindeutig identifiziert, die geschlossen werden soll.
RetVal	OUT 1	DINT	Wenn diese Funktion erfolgreich ausgeführt wurde, ist RetVal gleich “0”. Steht hier eine negative Zahl, konnte die Funktion nicht korrekt ausgeführt werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: File schliessen

      CALL  #FClose, DB457
       Handle:=#FileHandle
       RetVal:=#ReturnValue
      L     #ReturnValue
      L     0
      ==D
      SPB   next
      SPA   erro
```


5.3.10 “Delete” SFB458

Mit dem SFB “Delete” wird eine Datei oder ein ganzes Verzeichnis gelöscht. Über den Parameter FileName wird festgelegt in welchem Flash Filesystem sowie in welchem Verzeichnis die Datei gelöscht wird. Der Dateinamen muss immer als absoluter Dateiname angegeben werden. Wenn im Parameter FileName nur ein Verzeichnis angegeben wird, so wird dieses Verzeichnis mit dem ganzen Inhalt gelöscht.

Parameter des SFB 458:

Parameter	Typ	Art	Beschreibung
FileName	IN 0	STRING[64]	Der Parameter besteht aus dem Flash Filesystem Namen, dem Verzeichnisnamen sowie aus dem eigentlichen Dateinamen, bzw Verzeichnisnamen.
GroupAccess	IN 1	BYTE	Mit diesem Parameter können Zugriffsgruppen definiert werden. Für die Definition kann eine Kombination der GroupIds verwendet werden. Das Erzeugen oder Löschen einer Datei oder eines Unterverzeichnisses ist nur in einem Verzeichnis möglich, welches zu einer entsprechenden Gruppe gehört. Mit dem Wert “0” ist der Zugriff auf alle Gruppen erlaubt.
RetVal	OUT 2	DINT	Wenn diese Funktion erfolgreich ausgeführt wurde, ist RetVal gleich “0”. Steht hier eine negative Zahl, konnte die Datei nicht geöffnet werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Loeschen eines Files

      CALL  #Delete, DB458
       FileName   :="FileNames".M1Flash_DataFile
       GroupAccess:=B#16#0
       RetVal     :=#ReturnValue
      L     #ReturnValue
      L     L#0
      ==D
      SPB   next
      SPA   erro
```

5.3.11 “SWRITE” SFB459

Mit der Funktion SFB “SWRITE” wird zuerst die Datei geöffnet (oder erzeugt), dann die Daten von einem PLC- Media in die entsprechende Datei übertragen. Am Ende des Schreibvorganges wird die Datei wieder geschlossen. Diese drei Funktionen werden in einem Aufruf durchgeführt. Mit dem Parameter Buffer wird ein Any-Pointer auf die zu übertragenden Daten der Funktion übergeben.

Parameter des SFB 459:

Parameter	Typ	Art	Beschreibung
FileName	IN 0	STRING[64]	Der Parameter besteht aus dem Flash Filesystem Namen, dem Verzeichnisnamen sowie aus dem eigentlichen Dateinamen. Für den Dateinamen können nur alphanumerische Zeichen (ohne Space) sowie “.” verwendet werden. Ein Dateiname kann maximal 24 Zeichen lang sein. Die maximale Länge des absoluten Dateinamens darf maximal 64 Zeichen lang sein. Beispiel: `M1_FLASH:/Config/Maschine1.txt`
GroupID	IN 1	BYTE	Definiert zu welcher Gruppe die erzeugte Datei gehört. Folgende 4 Gruppen sind definiert: 0x10 USER Gruppe 1 0x20 USER Gruppe 2 0x40 USER Gruppe 3 0x80 USER Gruppe 4
GroupAccess	IN 2	BYTE	Mit diesem Parameter können Zugriffsgruppen definiert werden. Für die Definition kann eine Kombination der GroupIds verwendet werden. Das Erzeugen oder Löschen einer Datei oder eines Unterverzeichnisses ist nur in einem Verzeichnis möglich, welches zu einer entsprechenden Gruppe gehört. Mit dem Wert “0” ist der Zugriff auf alle Gruppen erlaubt.
Buffer	IN 3	ANY	Die Daten der PLC-Medien werden mit einem ANY-Pointer übergeben. Als PLC- Medien können DBs, Merker, Eingangs- und Ausgangsprozessabbild übergeben werden werden. Die maximale Länge ist auf 256 Bytes begrenzt.
RetVal	OUT 4	DINT	Wenn diese Funktion erfolgreich ausgeführt wurde, ist RetVal gleich “0”. Steht hier eine negative Zahl, konnte die Funktion nicht korrekt ausgeführt werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Daten in ein File schreiben

      CALL  #SWrite, DB459
       FileName   :="FileNames".M1Flash_DataFile
       GroupID    :=#GroupID
       GroupAccess:=#Group
       Buffer:="Bloecks".TempBlock[0]
       RetVal:=#ReturnValue
      L     #ReturnValue
      L     L#0
      ==D
      SPB   next
      SPA   erro
```


5.3.12 “SREAD” SFB460

Mit der Funktion SFB “SREAD” wird zuerst die Datei geöffnet (oder erzeugt), der Lesezeiger auf die Position Offset gesetzt und die Daten der Datei in das entsprechende PLC- Media übertragen. Am Ende des Lesevorgangs wird die Datei wieder geschlossen. Diese vier Funktionen werden in einem Aufruf durchgeführt. Mit dem Parameter Buffer wird ein Any-Pointer auf das zu übertragende PLC Media der Funktion übergeben.

Parameter des SFB 460:

Parameter	Typ	Art	Beschreibung
FileName	IN 0	STRING[64]	Der Parameter besteht aus dem Flash Filesystem Namen, dem Verzeichnisnamen sowie aus dem eigentlichen Dateinamen. Für den Dateinamen können nur alphanumerische Zeichen (ohne Space) sowie “.” verwendet werden. Ein Dateiname kann maximal 24 Zeichen lang sein. Die maximale Länge des absoluten Dateinamens darf maximal 64 Zeichen lang sein. Beispiel: `M1_FLASH:/Config/Maschine1.txt`
Buffer	IN 1	ANY	Die Daten der PLC-Medien werden mit einem ANY-Pointer übergeben. Als PLC- Medien können DBs, Merker, Eingangs- und Ausgangsprozessabbild übergeben werden werden. Die maximale Länge ist auf 256 Bytes begrenzt.
Offset	IN 2	DINT	Setzt den Lesezeiger nach dem Öffnen der Datei auf die entsprechende Position. Der Wert “0” entspricht dem Anfang der Datei. Dieser Parameter muss grösser oder gleich “0” sein.
RetVal	OUT 3	DINT	Wenn diese Funktion erfolgreich ausgeführt wurde, ist RetVal gleich der Anzahl Bytes, die in den Puffer übertragen wurden. Steht hier eine negative Zahl, konnte die Funktion nicht korrekt ausgeführt werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Daten von einem File lesen

      CALL  #SRead, DB460
       FileName   :="FileNames".M1Flash_DataFile
       Buffer     :="Bloecks".TempBlock            // Zeiger auf 256 Byte
       Offset     :=#ReadOffset
       RETVal     :=#ReturnValue
      L     #ReturnValue
      L     L#256
      ==D
      SPB   next
      SPA   erro
```


5.3.13 “FSCREATE” SFB461

Dieser FB übernimmt die Formatierung / das Anlegen eines Dateisystem in einer bestimmten Einheit. Diese Operation läuft asynchron; um die Operation fertigzustellen, ist mehr als ein Aufruf nötig.

Die eigentliche Operation startet wenn am Req Parameters eine“1” anliegt. Der Wert des Req Parameters wird nach dem ersten Aufruf solange nicht mehr ausgewertet wie der Done Parameter auf “1” steht. Bei Erkennung des Starts werden die Parameter initialisiert und der erste Aufruf kehrt zurück zu FIRST_CALL (0x7001). Dann geht die Funktion bis zur Beendigung auf INTERIM_CALL (0x7002).

Wenn der Busy Parameter von “1” auf “0” zurückgeht, kann der RetVal Parameter ausgewertet werden. Die Funktion wurde erfolgreich beendet, wenn in RetVal “0” zurückgegeben wird. Im Fehlerfall wird ein negativer Wert zurückgegeben. (Siehe 5.3.16 Fehlerinformationen). Im Fehlerfall bleibt die Fehlermeldung solange stehen, bis der Req Parameter auf “0” gesetzt wird. Mit einem auf “0” gesetzten Req Parameter, kehrt die Funktion zurück zu CALL_WITHOUT_EXEC (0x7000).

Der nächste Aufruf nach dem Beenden dieser Funktion mit Req = “1” startet die FSCREATE Funktion neu.

Der Aufruf dieser Funktion mit dem auf “1” gesetzten Force Parameter löscht alle Daten in dieser Einheit. Wird diese Funktion mit dem auf “0” gesetzten Force Parameter vor dem erkennen der Einheit aufgerufen, so wird die Formatierung gestartet und allfällige Daten auf dieser Einheit werden gelöscht.

Parameter des SFB 461:

Parameter	Typ	Art	Beschreibung
Req	IN 0	BOOL	Die eigentliche Operation startet wenn der Req- Parameter „1“ gesetzt ist. Ist der Done Parameter „1“, so wird der Req Parameter nicht ausgewertet.
Force	IN 1	BOOL	Der Force Parameter kann auf “0” oder “1” gesetzt werden. Auf “0” gesetzt erlaubt er den Aufruf der Funktion ohne die Formatierung der Einheit zu erzwingen, falls sie bereits vorhanden ist. Auf “1” gesetzt erlaubt er den Aufruf der Funktion und erzwingt die Formatierung der Einheit, auch wenn darauf bereits ein Dateisystem vorhanden ist. Diese Operation löscht alle Daten auf der Einheit und löst eine Formatierung aus. Ist der Force Parameter “0” und die Einheit ist vorhanden, wird aber nicht als Dateisystem erkannt, erfolgt die Formatierung. Alle vorher auf der Einheit gespeicherten Daten gehen verloren, z. B. auf die Einheit geschriebene Backup-Daten.
FSName	IN 2	STRING[16]	Über diesen Parameter wird definiert, welches Flash Filesystem formatiert werden soll.
BlockSize	IN 3	WORD	Dieser Parameter wird zur Zeit nicht unterstüzt. Der Wert muss auf 2048 gesetzt werden.
BlkNbr	IN 4	WORD	Dieser Parameter wird zur Zeit nicht unterstüzt. Der Wert muss auf 256 gesetzt werden.
MNbr	IN 5	WORD	Dieser Parameter wird zur Zeit nicht unterstüzt. Der Wert muss auf 32 gesetzt werden.
RetVal	OUT 6	DINT	Der RetVal Parameter wird entweder auf CALL_WITHOUT_EXEC (0x7000), FIRST_CALL (0x7001) oder INTERIM_CALL (0x7002) während der Ausführung gesetzt. Der eigentliche Rückgabewert wird gesetzt, wenn der Busy Parameter von “1” auf “0” geht. Eine “0” als Rückgabe zu diesem Zeitpunkt heisst, dass alles erfolgreich abgearbeitet wurde und dass auf die Einheit mit anderen FBs zugegriffen werden kann.
Busy	OUT 7	BOOL	Der Busy Parameter wird auf “1” gesetzt, so lange die Funktion in Ausführung ist und auf “0” wenn die Funktion beendet wird.
Done	OUT 8	BOOL	Der Done Parameter wird auf “0” gesetzt, so lange die Funktion in Ausführung ist und auf “1” wenn die Funktion erfolgreich beendet wird. Im Fehlerfall wird Done **nicht** auf „1“ gesetzt.

Programmierbeispiel:

```
Netzwerk 1: Formatieren eines File Systems

      CALL  „FSCREATE“ , DB461
       Req       :=#Req
       Force     :=#Force
       FSName    :=#Device
       BlockSize:=W#16#800              // = 2048 Dez
       BlkNbr    :=W#16#100             // = 256 Dez
       MNbr      :=W#16#20              // = 32 Dez
       RetVal    :=#RetVal
       Busy      :=#Busy
       Done      :=#Done
```


5.3.14 “FSCPRESS” SFB462

Dieser FB übernimmt die Kompression / die Wiederherstellung von frei gewordenen Blöcken eines Dateisystem in einer bestimmten Einheit. Diese Operation läuft asynchron; um die Operation fertigzustellen ist mehr als ein Aufruf nötig.

Auf dem R55Mxx Flash Modul wird ein Block entweder als frei (nicht benutzt) oder als benutzt (gerade benutzt) und frei geworden (zu einem früheren Zeitpunkt benutzt) gekennzeichnet. Frei gewordene Blöcke können nicht benutzt werden bis der Sektor, der den Block enthält, gelöscht ist (alle Bits auf “1”). Nur zu diesem Zeitpunkt kann ein frei gewordener Block in die Liste der freien Blöcke aufgenommen werden.

Intern im Dateisystem können einige Blöcke als frei geworden markiert sein, wobei ein Block hauptsächlich dann als frei markiert wird, wenn eine Datei / ein Verzeichnis gelöscht wird, womit dann aber alle enthaltenen Blöcke gelöscht werden.

Intern im Dateisystem wird die Kompression (Wiederherstellung von frei gewordenen Blöcken) automatisch ausgelöst, wenn einige Kriterien zusammentreffen, z. B. die Anzahl von frei gewordenen Blöcken beträgt 80% des Totals der Blöcke oder die Anzahl von frei gewordenen Blöcken ist grösser als die Anzahl der freien Blöcke, falls diese Anzahl geringer ist als 1/4 des Totals der Blöcke. Dies kann zu jeder Zeit und auch während einer Operation geschehen, auch wenn das Dateisystem beschäftigt meldet. Alle Aufrufe der Dateisystem-FBs werden dann den Kode FS_DEVICE_ BUSY zurückgeben.

Dieser FB kann vom Anwender zum Erzwingen der Kompression der Einheit eingesetzt werden, auch wenn die vorherigen Kriterien nicht zusammentreffen, z. B. eine Datei wird gelöscht und der Anwender möchte unverzüglich alle Blöcke in dieser Datei wiederherstellen.

Die Funktion wird gestartet, wenn der Req Parameter auf „1“ gesetzt ist, Bei ersten Aufruf wird der Busy Parameter auf „1“ und der Done Parameter auf „0“ gesetzt. Der Parameter RetVal gibt FIRST_CALL (0x7001) zurück. Die weiteren Aufrufe dieser Funktion geben INTERIM_CALL (0x7002) zurück, bis sie abgearbeitet ist.

Mit dem Req Parameter auf “0” gesetzt, gibt die Funktion CALL_WITHOUT_EXEC (0x7000) zurück.

Wenn der Busy Parameter auf „0“ gesetzt wird, kann RetVal ausgewertet werden. Im Fehlerfall gibt RetVal einen negativen Wert zurück (sieh 5.3.16 Fehlerinformationen).

Der Fehler bleibt solange stehen, bis der Req Parameter auf „0“ gesetzt wird. Konnte die Funktion erfolgreich beendet werden steht in RetVal „0“.

Wird die Funktionen nach dem Beenden erneut mit Req=“1“ aufgerufen, so startet FSCPRESS neu.

Parameter des SFB 462:

Parameter	Typ	Art	Beschreibung
Req	IN 0	BOOL	Die eigentliche Operation startet mit einer “1” am Req Parameters. Der Wert des Req Parameters wird während der Ausführung nicht mehr ausgewertet (solange Busy auf “1” steht).
DRVName	IN 1	STRING[16]	Über diesen Parameter wird definiert, welches Flash Filesystem formatiert werden soll.
Busy	OUT 2	BOOL	Der Busy Parameter wird auf “1” gesetzt, so lange die Funktion in Ausführung ist und auf “0”, wenn die Funktion beendet wird.
Done	OUT 3	BOOL	Der Done Parameter wird auf “1” gesetzt, sobald die Funktion erfolgreich beendet wurde.
RetVal	OUT 4	DINT	Der RetVal Parameter wird entweder auf CALL_WITHOUT_EXEC (0x7000), FIRST_CALL (0x7001) oder INTERIM_CALL (0x7002) während der Ausführung gesetzt. Der eigentliche Rückgabewert wird gesetzt, wenn der Busy Parameter von “1” auf “0” geht. Im Fehlerfall zeigt der RetVal Parameter den aufgetauchten Fehler an. Wurde die Funktion erfolgreich abgearbeitet wird der RetVal Parameter auf “0” gesetzt.

Programmierbeispiel:

```
Netzwerk 1: komprimieren eines File Systems

      CALL  „FSCPRESS“ , DB462
       Req    :=#Req
       DRVName:=#Device
       Busy   :=#Busy
       Done   :=#Done
       RetVal :=#RetVal
```

Aufgrund der eingesetzten Flash-Technologie in den PCD7.R55x Modulen benötigt diese Funktion für die ausführung sehr viel Zeit. Abhängig von der Anzahl Blöcken und Sektoren die komprimiert werden müssen dauert die Ausführung zwischen 1Sek und 4 Minuten. Ein Ausschalten der Steuerung sollte während dieser Zeit vermieden werden.

Wird während der Ausführung der FSCPRESS Funktion die Steuerung ausgeschalten, so wird der Compress Algorithmus beim Aufstarten erneut ausgeführt. Dies kann zur Folge haben, dass diese Einheit nach dem Aufstarten während mehreren Minuten nicht zur Verfügung steht.

5.3.15 “FSGETSIZ” SFB463

Mit der Funktion FSGETSIZ können Informationen von einer spezifischen Einheit ausgelesen werden.

Parameter des SFB 463:

Parameter	Typ	Art	Beschreibung
DRVName	IN 0	STRING[16]	Über diesen Parameter wird definiert, von welchem Flash Filesystem Informationen ausgelesen werden sollen.
TSIZE	OUT 1	DINT	Grösse der Einheit.
FSIZE	OUT 2	DINT	Grösse des freien Speichers.
USIZE	OUT 3	DINT	Grösse des belegten Speichers.
RetVal	OUT 4	DINT	Wenn diese Funktion erfolgreich ausgeführt wurde, ist der RetVal Parameter gleich “0”. Steht hier eine negative Zahl, konnte die Funktion nicht korrekt ausgeführt werden. Siehe Kapitel Fehlerauswertung.

Programmierbeispiel:

```
Netzwerk 1: Deviceinformationen auslesen

      CALL  „FSGETSIZ“ , DB463
       DRVName:=#Device
       TSize  :=#TSize
       FSize  :=#Fsize
       USize  :=#Usize
       RETVal :=#RetVal
```

5.3.16 Fehlerinformationen der Dateisystem SFBs

Die SFBs 450 bis 463 enthalten im Falle eines Fehlers in der Variablen RET_VAL weitere Fehlerinformation. Die Fehlercodes sind teilweise nach Standard S7 ausgeführt. Einige Fehlercodes wurden speziell für Dateisystem-Funktionen hinzugefügt.

RETVAL	Beschreibung
0x7000	Erster Aufruf mit REQ = “0”: Funktion ist nicht aktiv
0x7001	Erster Aufruf mit REQ = “1”: Funktion gestartet
0x7002	Zwischenaufruf (REQ irrelevant): Funktion ist bereits aktiv
0x0000	Funktion erfolgreich abgeschlossen
0xFF9B	Interner Dateisystem-Fehler (PLC)
0xFF9C	Ungültiger Typ
0xFF9D	Einheit nicht gefunden
0xFF9E	Ungültiger Parameter
0xFF9F	Ungültiges Argument
0xFFA0	Datei nicht vorhanden
0xFFA1	Ungültiger Dateiname
0xFFA2	Ungültige Gruppe
0xFFA3	Ungültige Ebene
0xFFA4	Ungültiger Zugriffstyp
0xFFA5	Ungültiger Einheiten-Name
0xFFA6	Ungültiger Verzeichnisname
0xFFA7	Datei existiert bereits
0xFFA8	Nicht genügend Speicher (Aufruf der Funktion FSCPRESS notwendig)
0xFFA9	Die max. Anzahl offener Dateien ist überschritten.
0xFFAA	Datei ist nicht geöffnet
0xFFAB	Datei ist bereits geöffnet
0xFFAC	Ungültiger Zugriffstyp
0xFFAD	Ungültiger Dateityp
0xFFAE	Ungültiges Schreibattribut
0xFFAF	Ungültiger Parameter-Puffer
0xFFB0	Fehler während der Schreiboperation
0xFFB1	Fehler während der Leseoperation
0xFFB2	DAS-Zugriff nicht möglich
0xFFB3	Zugriff nicht möglich
0xFFB4	Ungültige File Id
0xFFB5	Ungültiger Benutzer
0xFFB6	Ungültige REGFLAGS
0xFFB7	REG_ENTRY_TABLE_FULL
0xFFB8	INVALID_REGID
0xFFB9	FILE_SYSTEM_CHECK_ERROR
0xFFBA	Ungültiger Name für die Einheit
0xFFBB	Einheit nicht geladen
0xFFBC	Name für Einheit existiert bereits
0xFFBD	Ungültige Operation
0xFFBE	Ungültiger Flash Wert
0xFFBF	Flash Operation konnte nicht ausgeführt werden
0xFFC0	Fehler während des Kompressionsvorgangs
0xFFC1	Einheit ist beschäftigt
0xFFC2	Operation wird zu einem späteren Zeitpunkt ausgeführt.
0xFFC3	Funktion nicht implementiert.
0xFFC4	Interner FS- Fehler

Wenn eine Funktion die Fehlermeldung “Einheit ist beschäftig” (0xFFC1) zurückgibt, so soll der Anwender diese Funktion zu einem späteren Zeitpunkt nochmals aufrufen. Dies ist Beispielsweise möglich, wenn eine Einheit Komprimiert wird, so können während dieser Zeit keine Daten auf diese Einheit geschrieben werden.

6 PC104 Funktionen für die PCD2.M257 Dual Port Ram

6.1 Allgemeines

Bei der Steuerung PCD2.M257, (PCD2.M157 mit aufgesetztem IPC) bestehen via Dual Port RAM (DPR) 2 Möglichkeiten des Datentransfers.

- S-Bus Modus
- Transparent Modus

Dabei stellt das DPR einen Speicherbereich zum Austausch der Daten zwischen der SPS und dem PC104-Board dar. Es erlaubt beiden Seiten Daten zu senden, bzw. zu empfangen.

Im **S-Bus Modus** ist auf der SPS-Seite keine weitere Programmierung notwendig, d.h. der Austausch der Daten via DPR ist von der Step®7-Seite nicht zugänglich (kein ereignisgesteuertes Senden von der SPS).

Im **Transparent Modus** ist auf der SPS-Seite Programmierung notwendig. Die SPS ist aktiv, d.h. sie kann ereignisgesteuerte Daten zum PC104-Teil senden. Dadurch, und durch jeweils 2 Empfangs- und Sendepuffer kann eine höhere Austauschrate der Daten erzielt werden.

Auf der SPS-Seite gibt es für den Transparent Modus 3 SFCs, die es der S7 Applikation erlauben, Daten vom PC104-Teil zu lesen, bzw. dahin zu schreiben.

- Lesen vom Dual-Port-RAM, SFC227, PC104_RD
- Schreiben ins Dual-Port-RAM, SFC228, PC104_WR
- Status vom Dual-Port-RAM abfragen, SFC 229, PC104_ST

Weitere Informationen zum Datenaustausch und den Software-Komponenten des PC-Teils kann im Handbuch 26/759 SPS mit integriertem IPC nachgelesen werden.

6.2 Lesen vom Dual-Port-RAM (SFC227 PC104_RD)

Die SFC 227 liest die Anzahl selektierter Bytes aus einem der 2 Empfangspuffer und legt sie im entsprechenden Datenbereich ab. Ausserdem ist eine ID im empfangenen Datenpaket, die diese identifiziert.

Parameter des SFC 227:

Parameter	Typ	Art	Bereich	Beschreibung
BUF_NO	IN 0	INT	0...2	Durch diesen Parameter kann der Anwender wählen, ob er die Daten in einen bestimmten Empfangspuffer lesen will oder ob die Saia PCD® die Daten vom “ältesten“ Puffer holen soll: 0: erster Verfügbarer 1: Empfangspuffer 1 2: Empfangspuffer 2
DEST	IN 1	ANY	gültiger ANY Zeiger	Spezifiziert den Zielbereich, wohin die Daten transferiert werden. Die Längenangabe wird nicht ausgewertet. Der Typ muss BYTE sein.
LEN	IN_OUT 0	INT	gültige Länge	Grösse des Anwenderzielbereichs. Sie muss gleich oder grösser der erwarteten max. Datenmenge sein. Nach der Übertragung enthält “LEN“ die Anzahl übertragener Bytes.
ID	OUT 0	WORD	gültige ID	Die ID des Headers wird zurückgegeben.
Ret_Val	OUT 1	INT	Fehlermeldungen	Fehlermeldung: Ausser den Step®7 - spezifischen Fehlernummern (z.B. ungültiger Pointer oder DB nicht gefunden u.s.w) sind folgende Fehlernummern möglich: 0xFFFF: kein Puffer frei 0xFFFE: ungültige Puffernummer 0xFFFD: Puffer zu klein 0xFFF0: Protokollfehler (d.h. der PC hat das Protokoll nicht umgeschaltet.

Programmierbeispiel:

```
Netzwerk 1: Daten vom Dual-Port-RAM lesen

      CALL  "PCD104_RD"
       BUF_NO :=2
       DEST   :=DB120.DBD4            // Zielbereich der zu lesenden Daten
       ID     :=MW120
       Ret_val:=MW122
       LEN    :=MW124
```

6.3 Schreiben ins Dual-Port-RAM (SFC228, PC104_WR)

Der SFC 228 schreibt die Anzahl selektierter Bytes aus dem entsprechenden Datenbereich in den DPR. Ausserdem wird eine ID zur Identifizierung des Datenpakets gesendet.

Parameter des SFC 228:

Parameter	Typ	Art	Bereich	Beschreibung
BUF_NO	IN 0	INT	0...2	Durch diesen Parameter kann der Anwender wählen, ob er die Daten in einen bestimmten Sendepuffer schreiben will, oder ob die Saia PCD® die Daten in einen verfügbaren Puffer schreiben soll. 0: erste verfügbare 1: Empfangspuffer 1 2: Empfangspuffer 2
SRC	IN 1	ANY	gültiger ANY Zeiger	Spezifiziert den Quellbereich, woher die Daten ins DPR transferiert werden. Die Längenangabe wird nicht ausgewertet. Der Typ muss BYTE sein.
LEN	IN 2	INT	gültige Länge	Grösse des Anwenderquellbereichs.
ID	IN 3	WORD	gültige ID	Diese ID wird in den Header eingetragen.
Ret_Val	OUT 0	INT	Fehlermeldungen	Fehlermeldung: Ausser den STEP®7 - spezifischen Fehlernummern (z.B. ungültiger Pointer oder DB nicht gefunden u.s.w) sind folgende Fehlernummern möglich: 0xFFFF: kein Puffer frei 0xFFFE: ungültige Puffernummer 0xFFFD: Puffer zu klein 0xFFF0: Protokollfehler (d.h. der PC hat das Protokoll nicht umgeschaltet.

Programmierbeispiel:

```
Netzwerk 1: Daten ins Dual-Port-RAM schreiben

      CALL  "PCD104_WR"
       BUF_NO :=2
       SRC    :=DB130.DBD4       // Quellbereich der zu schreibenden Daten
       LEN    :=20
       ID     :=MW150
       RET_VAL:=MW228
```

6.4 Status vom Dual-Port-RAM (SFC229 PC104_ST)

Der SFC 229 liefert den Status der Puffer des Dual-Port-RAMs. Als Rückgabewert gibt es die Synchro Bytes, ob ein Puffer benutzt wird und die Anzahl Bytes in den Empfangspuffern.

Parameter des SFC 229:

Parameter	Typ	Art	Bereich	Beschreibung
BUFBITS	OUT 0	WORD	Status des DUAL-Port-Rams	Dieses Byte ist das XOR-Ergebnis der beiden Semaphorbytes im DualPort-Ram Bit 0 = 1: Empfangspuffer 1 belegt Bit 1 = 1: Empfangspuffer 2 belegt Bit 4 = 1: Sendepuffer 1 belegt Bit 5 = 1: Sendepuffer 2 belegt
LEN_1	OUT 1	INT	Anzahl Bytes	Anzahl bytes im Empfangspuffer 1.
LEN_2	OUT 2	INT	Anzahl Bytes	Anzahl bytes im Empfangspuffer 2
Ret_Val	OUT 3	INT	Fehlermel-dung	Fehlermeldung: 0xFFF0: Protokollfehler (d.h. der PC hat das Protokoll nicht umge-schaltet.

Programmierbeispiel:

```
Netzwerk 1: Status vom Dual-Port-RAM

      CALL  "PCD104_ST"
       BUFBITS:=MW300                  // Puffer belegt?
       LEN_1  :=MW302                  // Anzahl Bytes im Empfangsbuffer 1
       LEN_2  :=MW304                  // Anzahl Bytes im Empfangsbuffer 2
       Ret_val:=MW230
```

7 Smart7 Funktionen

7.1 Allgemeines

Die Smart7 ist eine Baugruppe im Kreditkartenformat mit kompletter SPS-Funktionalität und bietet eine einfache Integration in anwenderspezifische Steuerungselektronik. Als Entwicklungsumgebung steht ein Evaluation-Board für die 2 CPUs PCD.Smart.M137 und PCD.Smart.M177 zur Verfügung. Der Zugang vom Anwenderprogramm in der CPU zu den Schnittstellen erfolgt bei bestehenden Schnittstellenmodulen wie bei herkömmlichen Steuerungen der Saia PCD® Serie xx7.

Auf Eigenentwicklungen für den Parallelbus wird mittels zweier SFBs zugegriffen. Diese sind:

- Schreibzugang zum Chip Select, SFB254, WriteCS
- Lesezugang zum Chip Select, SFB255, ReadCS

Die Bausteine SFB254 und SFB 255 sind nur in Smart7 CPUs im Betriebssystem vorhanden und auch nur dort ablauffähig.

Soll ein Baustein als Multiinstanz verwendet werden, muss zuerst die Step®7-Software durch Aufruf des SFBs den Instanz-DB erzeugen. Der Aufruf des SFBs muss im FB für die Multiinstanz erfolgen. Danach kann der SFB als statische Variable deklariert werden.

Die Bausteine sind hier nur der Vollständigkeit halber erwähnt. Weitere Informationen bekommen Sie von unseren Aussendienst Mitarbeitern oder über mail to: support@saia-pcd.com

7.2 Smart 7 Schreibzugriff auf Chip Select (SFB254, WriteCS)

Durch den Aufruf des SFB 254 kann schreibend auf die beiden Chip Selects (CSDRT und CS_F2), die den Parallelbus (SLOT B1 und B2) ansprechen, zugegriffen werden. Dieser SFB ist völlig transparent und bietet direkten Zugriff in Byte-, Wort,- oder Dop-pelwort Format. Durch einmaligen Aufruf wird der selektierte Quelldatenbereich auf den Parallelbus transferiert.

Parameter des SFC 254:

Parameter	Typ	Art	Bereich	Beschreibung
AREA	IN 0	INT	0...2	Auswahl des Chip Select (Slotnummer) 0: Nicht verwendet 1: Slot B1 (CSDRT) 2: Slot B2 (CS_F2)
ACCESS	IN 1	INT	0...2	Datenzugang: 0: Byte (Grösse=1) 1: Wort (Grösse=2) 2: Doppelwort (Grösse=4) Wort und Doppelwort müssen mit gerader Adresse beginnen!
OFFSET	IN 2	WORD		Byteoffset im Datenbereich (AREA)
LEN	IN 3	INT	0... 32767	Länge des zu schreibenden Blocks in Byte,- Wort,- oder Doppelwortform abhängig vom Parameter ACCESS. Wert = Länge in Bytes * Grösse (ACCESS)
P_SRC	IN 4	ANY		Zeiger auf die Anfangsadresse des Quellbereichs
RET_VAL	OUT 5	INT	-6...0	Rückgabewerte: 0: OK -1: Falsche Slot Nr.(AREA) -2: Falscher Zugriff (ACCESS) -3: Fehler im Zeiger Quellbereich (P_SRC) -4: Falsche Länge (LEN) -5: Überlauf Zeiger Quellbereich (P_SRC) -6: Wort,- oder Doppelwortzugriff auf ungerader Adresse

Programmierbeispiel:

```
Netzwerk 1: Schreibzugriff auf Parallelbus

      U     E      0.0                  // Schreibanforderung
      FP    M      0.0
      SPBN  nWRT                        // Impuls
      CALL  SFB  254 , DB254
       IN0 :=1                          // Slot B1, CsDRT
       IN1 :=0                          // Bytezugriff
       IN2 :=W#16#0                     // Byteoffset
       IN3 :=20                         // 20 Bytes
       IN4 :=DB11.DBX0.0                // Anfangsadresse Quellbereich
       OUT5:=MW200
nWRT: NOP   0
```

7.3 Smart 7 Lesezugriff auf Chip Select (SFB255, ReadCS)

Durch den Aufruf des SFB 255 kann lesend auf die beiden Chip Selects (CSDRT und CS_F2), die den Parallelbus (SLOT B1 und B2) ansprechen, zugegriffen werden. Dieser SFB ist völlig transparent und bietet direkten Zugriff in Byte-, Wort,- oder Doppelwortform. Durch einmaligen Aufruf wird der selektierte Zieldatenbereich vom Parallelbus in den Datenbereich des Anwenderprogramms transferiert.

Parameter des SFC 255:

Parameter	Typ	Art	Bereich	Beschreibung
AREA	IN 0	INT	0...2	Auswahl des Chip Select (Slotnummer) 0: Nicht verwendet 1: Slot B1 (CSDRT) 2: Slot B2 (CS_F2)

ACCESS	IN 1	INT	0...2	Datenzugang: 0: Byte (Grösse=1)0 1: Wort (Grösse=2) 2: Doppelwort (Grösse=4) Wort und Doppelwort müssen mit gerader Adresse beginnen!
OFFSET	IN 2	WORD		Byteoffset im Datenbereich (AREA)
LEN	IN 3	INT	0... 32767	Länge des zu schreibenden Blocks in Byte,- Wort,- oder Doppelwortform abhängig vom Parameter ACCESS. Wert = Länge in Bytes • Grösse (ACCESS)
P_SRC	IN 4	ANY		Zeiger auf die Anfangsadresse des Zielbereichs
RET_VAL	OUT 5	INT	-6...0	Rückgabewerte: 0: OK -1: Falsche Slot Nr.(AREA) -2: Falscher Zugriff (ACCESS) -3: Fehler im Zeiger Zielbereich (P_SRC) -4: Falsche Länge (LEN) -5: Überlauf Zeiger Zielbereich (P_SRC) -6: Wort,- oder Doppelwortzugriff auf ungerader Adresse

Programmierbeispiel:

```
Netzwerk 1: Lesezugriff auf Parallelbus

      U     E      0.0                  // Leseanforderung
      FP    M      0.0
      SPBN  nRD                         // Impuls
      CALL  SFB  255 , DB255
       IN0 :=1                          // Slot B1, CSDRT
       IN1 :=0                          // Bytezugriff
       IN2 :=W#16#0                     // Byteoffset
       IN3 :=20                         // 20 Byte Lesen
       IN4 :=DB11.DBX0.0                // Anfangsadresse Zielbereich
       OUT5:=MW200
nRD:  NOP   0
```

8 Systemkonfiguration (CDB)

8.1 Allgemeines

Die Saia PCD® Typen der Serie xx7 besitzen konfigurierbare Eigenschaften, die nicht mit dem SIMATIC Manager eingestellt werden können. Bis zur Einführung des Kompilers auf der PCD2.Mx57 konnte man nur die I/Os konfigurieren. Mit der Einführung eines weiterten Datenbausteins, dem Konfigurationsdatenblock (CDB), ist es nun möglich, neben den I/Os auch Systemeigenschaften zu konfigurieren. Der Aufbau des CDBs ist so angelegt, dass die Konfigurationen in ASCII Text eingetragen werden. Die meisten Einstellungen können über den I/O Builder vorgenommen werden. Dieser erzeugt automatisch die notwendigen CDB-Einträge. Der I/O-Builder kann kostenlos unter www.sbc-support.com heruntergeladen werden.

Folgende Konfigurationen sind möglich:

- Speicheraufteilung
- Prioritäten beim Kompilieren
- Schnittstellen- und Modem- Initialisierung
- Web-Server
- Peripheriezugriff im OB100
- Konfiguration Profi-S-IO-Master / MPI-Schnittstelle

Aufbau und Struktur des CDB
Der CDB wird vom Betriebssystem unter folgenden Bedingungen erkannt:

- Die Datenbaustein-Nr. muss DB1, DB511 oder DB1023 sein
- Die Kennung “SBC xx7 CDB” muss am Anfang des DBs stehen (case sensitiv)

Dabei ist die Kennung als STRING mit mindestens 12 Zeichen zu deklarieren. Wird ein längerer String deklariert, werden zur Kennung nur die ersten 12 Zeichen gelesen. Der Anwender kann in diesem String weitere Informationen wie z.B. Versionsnummer usw. anhängen.

Beispiel:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
=14.0		END_STRUCT	

Der Konfigurationsdatenbaustein besteht aus Strings, so dass die Informationen im Textformat zu lesen sind. Dabei ist folgende String-Syntax zu beachten:

- Alle Leerzeichen werden beim Auswerten ignoriert. Sie dienen nur der Leserlichkeit.
- Kleinbuchstaben werden in Grossbuchstaben konvertiert.

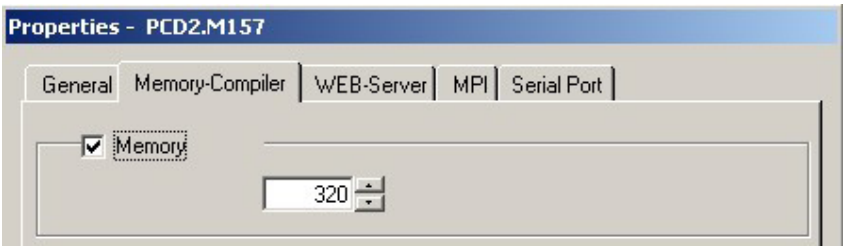
Generell besteht der String aus einem Schlüsselwort und Parametern. Das Schüsselwort endet mit einem Doppelpunkt ":". Die Parameter sind abhängig von der eigentlichen Funktionalität, die sie konfigurieren. Bei der Eingabe im String gelten folgende Regeln:

- Parameterwerte werden durch ein Komma "," getrennt
- Parameternamen werden durch ein Gleichheitszeichen "=" von den Werten getrennt
- Kommentare im CDB beginnen mit "//"

Auswertung des CDB
Der CDB wird nach Netz ein und nach einem Stop Run Übergang ausgewertet. Die Speicher- und die Kommunikationsparameter werden nur nach Netz Ein ausgewertet.

8.2 Speicherskalierung

Der S7-Code Speicher ist bei den meisten Steuerungen der Serie xx7 skalierbar. Durch Doppelklick der entsprechenden Steuerung im I/O-Builder kann der Speicher in 64kByte Schritten konfiguriert werden.



Die maximale Speichergrösse der verschiedenen Steuerungen ist aus der folgenden Tabelle ersichtlich:

PCD-Typ	S7 Code Speichergrösse
PCD1.M137	Default: 64 kByte. 64 kB oder 128 kB konfigurierbar.
PCD2.M127	Fix 132 kByte
PCD2.Mx57	Default: 256 kByte. Von 64 kB bis 512 kB konfigurierbar.
PCD2.M177	Default: 512 kByte. Von 64 kB bis 1024 kByte konfigurierbar.
PCD2.M487	Default: 1 MByte. Von 64 kB bis 1024 kByte konfigurierbar.
PCD3.Mxxx7	Default: 512 MByte. Von 64 kB bis 512 kByte konfigurierbar.

Bei den Steuerungen PCD2.Mx57 und PCD2.M177 wird der gesamte Anwender-speicher in S7-Code Speicher und Compilerspeicher aufgeteilt. Für gewisse Anwendungen kann es notwendig sein, mehr S7-Speicher zur Verfügung zu haben, z.B. wenn ein S7–Projekt einen grossen Anteil an DBs und wenige Programmbausteine enthält. Bei der Speicherzuordnung des S7-Codes muss darauf geachtet werden, wieviel Speicher für den Compiler Code verbleibt. Je kleiner der S7-Code Speicher ist, um so grösser ist der Compilerspeicher.
Diese Einschränkung gilt nicht für die PCD2.M487 und die PCD3.Mxxx7!
Genaueres kann aus dem Kapitel Speicher/Flash Funktionen entnommen werden.

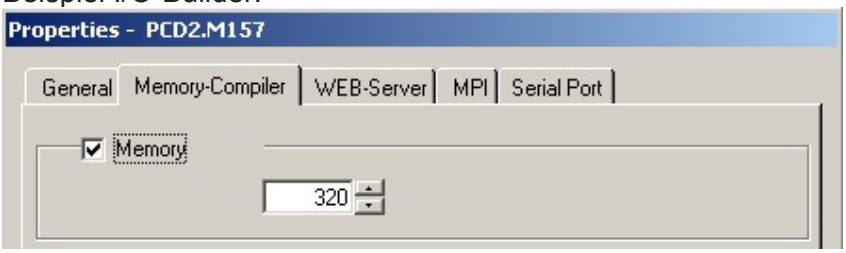
Wenn die Funktionalität des DB-Speichers benutzt wird, muss beachtet werden, dass der zur Verfügung stehende Platz des DB-Flash Speichers von der konfigurierten S7-Code Speichergrösse abhängig ist. Genaueres kann aus dem Kapitel Flash Funktionen entnommen werden.

Schlüsselwort: MEM7

Parameter: S7-Code Speichergrösse. Die Grösse des S7-Code Speichers muss mindestens 64 oder ein vielfaches von 64 sein. Werte die “0“ oder kein Vielfaches von 64 sind, werden ignoriert und habe keinen Einfluss auf die Steuerung. Setzt man einen zu grossen Wert ein, wird der Speicher auf den maximal möglichen Wert konfiguriert.

Die Speicherkonfiguration wird nur nach Netz Ein ausgewertet.

Beispiel I/O-Builder:



Die obige Einstellung im I/O-Builder erzeugt den folgenden CDB.

Beispiel Einstellung im CDB:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	Memory	STRING[8]	´MEM7:320´
=24.0		END_STRUCT	

Der für den S7-Code zur Verfügung stehende Speicher wird auf 320 kByte gesetzt.

8.3 Prioritäten beim Kompilieren

Bei jedem Stop-Run-Übergang wird versucht, das gesamte Step®7-Programm zu übersetzen. Dieser Vorgang kann einige Sekunden dauern. Während dieser Zeit blinkt die Led ‘RUN‘. Per Voreinstellung werden Bausteine in der folgenden Reihen-folge übersetzt:

- FC in absteigender Reihenfolge,
- FB in absteigender Reihenfolge,
- OB in absteigender Reihenfolge

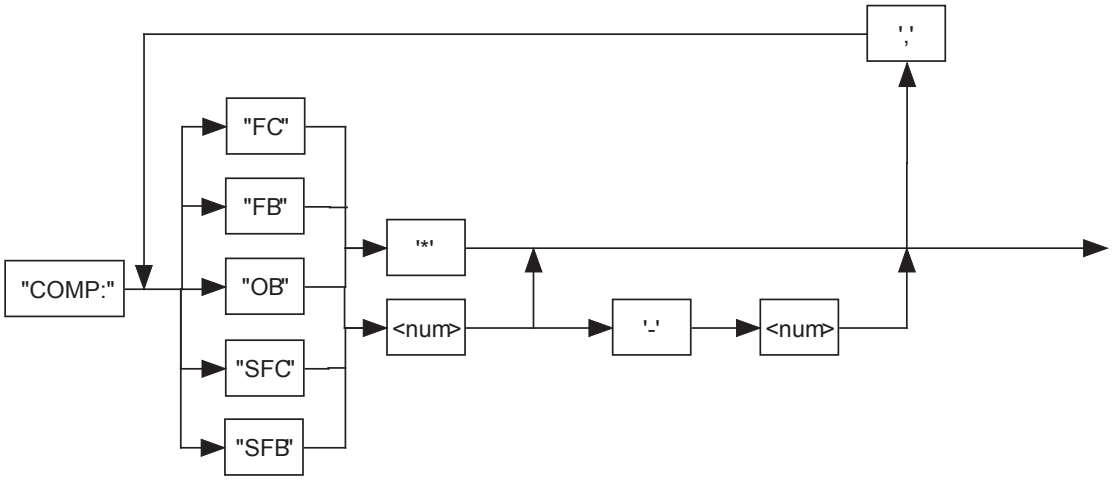
Können nicht alle Bausteine kompiliert werden, kann via CDB eine abweichende Reihenfolge definiert werden. Dies geschieht über das Schlüsselwort “COMP”: Wenn nach der priorisierten Kompilation noch Platz im Kompilatspeicher frei ist, werden weitere Bausteine nach der voreingestellten Reihenfolge übersetzt.

Werden Bausteine im Zustand Run übertragen so werden diese, falls möglich, übersetzt. Dabei kann jedoch keine Rücksicht auf evtl. Priorisierungen genommen werden. Es kann durchaus der Fall sein, dass ein Baustein der als Kompilat vorhanden gewesen ist, nach einer Änderung nicht mehr kompiliert werden kann. In diesem Fall kann ein Stop-Run-Übergang dazu führen, dass besagter Baustein wieder kompiliert wird.

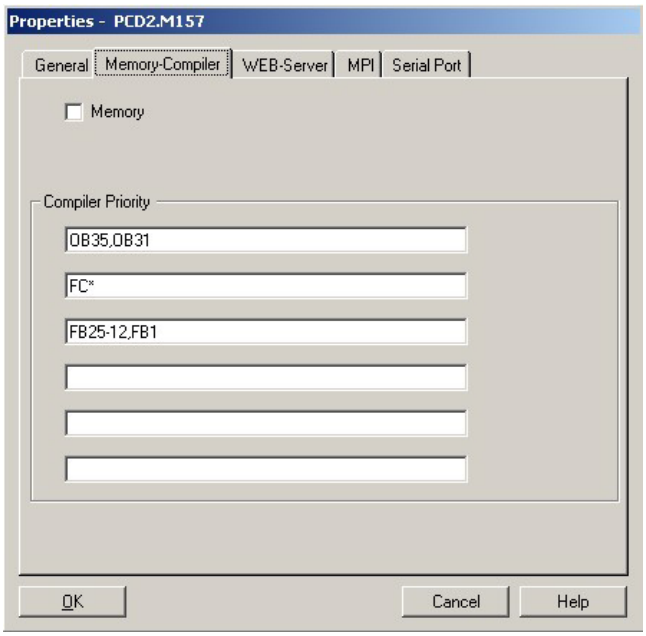
Schlüsselwort: COMP:

Parameter: Bausteintyp.
`*` Alle Bausteine des angegebenen Typs werden in aufsteigender Reihenfolge beim Kompilieren priorisiert. (Umkehrung der Voreinstellung)
`-` Bausteinbereich: Wird zuerst die höhere Bausteinnummer angegeben, so wird der Bereich in absteigender Reihenfolge kompiliert.

Syntaxdiagram:



Beispiel Einstellung im I/O-Builder:



Die obige Einstellung im I/O-Builder erzeugt den folgenden CDB.

Beispiel Einstellung im CDB:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	Compil_priority_0	STRING[14]	´COMP:OB35,OB31´
+30.0	Compil_priority_1	STRING[8]	´COMP:FC*´
+40.0	Compil_priority_2	STRING[16]	´COMP:FB25-12,FB1´
=58.0		END_STRUCT	

Im obigen Beispiel wird zuerst der OB35 und anschliessend der OB31 kompiliert. Weiter werden alle FC in aufsteigender Reihenfolge kompiliert. Falls noch genügend Compilerspeicher vorhanden ist, werden die FB12-25 in absteigender Reihenfolge (FB25-FB12) kompiliert. Anschliessend wird der FB1 kompiliert. Falls noch weiterer freier Compilerspeicher vorhanden ist, werden die restlichen FBs in absteigender Reihenfolge kompiliert. Zum Schluss werden noch die restlichen OBs in absteigender Reihenfolge kompiliert.

8.3.1 Lesen des Compiler Status (SFC230)

Mit dem SFC230 kann der Anwender testen, ob der aktuelle Baustein interpretiert wird oder im Compilat läuft. Mit dem SFC300 kann der Anwender zusätzlich testen, ob von einem bestimmten Baustein ein Compilat vorhanden ist. Der SFC230 ist parameterlos. Er interpretiert den Akku 1 als Eingangsparameter. Die Rückgabe erfolgt über das Statuswort (Bits A0, A1 und VKE).

Eingangsparameter im Akku 1:

Akku 1	
MSW	LSW
Typenkennung (0x08 = OB, 0x0C = FC, 0x0E = FB)	Bausteinnummer

Bedeutung der Rückgabe:

VKE	A0	A1	Beschreibung
0	x	x	aktueller Baustein wird interpretiert
1	x	x	aktueller Baustein läuft im Compilat
x	0	0	Der Baustein ist nicht geladen
x	1	0	Der Baustein ist nicht compiliert
x	0	1	Der Baustein ist als Compilat vorhanden
x	1	1	Ungültiger Parameter wurde in Akku 1 übergeben.

Beispiel:

```
Netzwerk 1: Überprüfen, ob aktueller Baustein als Compilat läuft

       UC  SFC 230                    // Aufruf
       SPB  Comp
// Der Baustein wird interpretiert
       .
       .
       BEA
Comp:  NOP 0
// Der Baustein läuft als Compilat
       .
       .
```

Beispiel:

```
Netzwerk 1: Überprüfen, ob FB 1 als Compilat vorhanden ist

       L   DW#16#000E0001             // Info über FB 1
       UC  SFC 230                    // Aufruf
       SPZ  m_01
       SPM  m_02
       SPP  m_03
       SPU  m_04
       BEA
m_01:  NOP 0      // Der Baustein ist nicht geladen
       .
       .
       BEA
m_02:  NOP 0      // Der Baustein ist nicht compiliert
       .
       .
       BEA
m_03:  NOP 0      // Der Baustein ist als Compilat vorhanden
       .
       .
       BEA
m_04:  NOP 0      // Ungültiger Parameter in Akku 1
       .
       .
       BEA
```

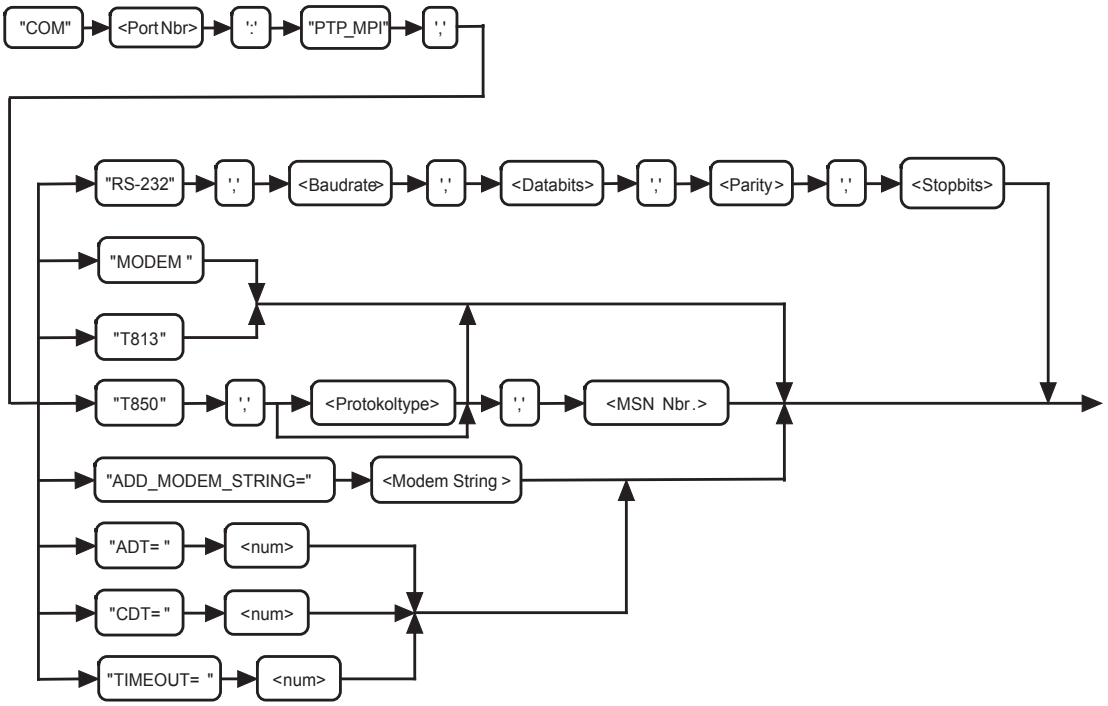
8.4 Initialisierung des seriellen Programmierinterfaces

Alle seriellen Schnittstellen der Steuerungen PCD2.xx7 können mittels Konfigurationsdatenblock als Programmierinterface initialisiert und konfiguriert werden. Weitere Möglichkeiten sind im Handbuch 26/794 Serielle Kommunikation beschrieben.

Folgende Einschränkungen sind zu beachten!

- Es kann immer nur eine Schnittstelle das MPI-Protokoll unterstützen!
- Wird kein Eintrag oder eine ungültige Parametereingabe vorgenommen, wird der Defaultwert genommen
- Die Schnittstellenkonfiguration wird nur nach Netz Ein ausgewertet.

Das nachfolgende Syntaxdiagramm gibt einen Überblick über die möglichen Einstellungen. In den nachfolgenden Kapiteln werden die einzelnen Parameter näher erklärt.



In der Grundeinstellung wird die serielle Schnittstelle COM1 (PCD2.M487 Port 0) mit folgenden Werten initialisiert:

19’200 Baud, 8 Datenbits, Odd Parity, 1 Stoppbit

Dabei ist das MPI-Protokoll eingeschaltet.

Folgende Portnummern werden unterstüzt:

PCD1.M137:	Port 1 bis 3, default Port =1 (Port 1 kann nur für Modem benutzt werden)
PCD2.M127/Mx57:	Port 1 bis 3, default Port =1
PCD2.M177:	Port 1 bis 5, default Port =1
PCD2.M487:	Port 0 bis 5, default Port =0
PCD3.Mxxx7:	Port 0 / 1, default Port =0

8.4.1 RS-232 Parameter

Schlüsselwort: COM<n>:PTP_MPI,RS-232,<baud>,<data>,<parity>,<stop>

Parameter:
<n>: Schnittstellen-Nummer (0...5), siehe Tabelle Baudraten
<baud>: folgende Baudraten werden unterstützt:

Saia PCD®	Port Nummer	unterstützte Baudraten
PCD1.M137	Port 1	Kann nur als Modem Port benützt werden.
	Port 2 / 3	300, 600, 1’200, 2’400, 4’800, 9’600, 19’200, 38’400
PCD2.M127/Mx57	Port 1-3	300, 600, 1’200, 2’400, 4’800, 9’600, 19’200, 38’400
PCD2.M177	Port 1-5	300, 600, 1’200, 2’400, 4’800,9’600, 19’200, 38’400
PCD2.M487	Port 0 / 1	1’200, 2’400, 4’800, 9’600, 19’200, 38’400, 57’600, 115’200
	Port 2-5	300, 600, 1’200, 2’400, 4’800, 9’600, 19’200, 38’400
PCD3.Mxxx7	Port 0 / 1	1’200, 2’400, 4’800, 9’600, 19’200, 38’400, 57’600, 115’200

Port 2 und 3 bzw. Port 4 und 5 unterstüzen entweder 19’200 oder 38’400 Baud. Welche Baudrate das F-Modul unterstüzen soll, wird mit einem CDB Eintrag festgelegt. Siehe Kapitel Einstellen der max. Baudrate auf Steckplatz B1 (B2).

<Data>: Anzahl Datenbits (7 oder 8)
<Parity>: Parität:
E = Even, O = Odd, N = None, L = Force Low, H = Force High
<Stop>: Anzahl Stoppbits (1 oder 2)

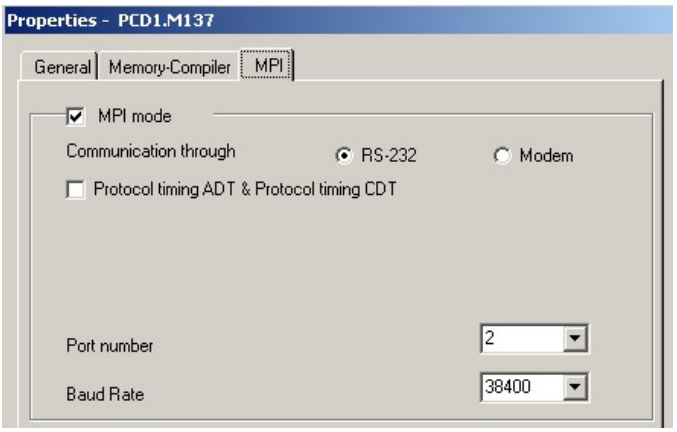
Wird die konfigurierte serielle Schnittstell als serielle MPI-Schnittstelle benutzt, so müssen die Einstellungen

COM<n>:PTP_MPI,RS-232,19’200,8,O,1 oder

COM<n>:PTP_MPI,RS-232,38’400,8,O,1

verwendet werden, andernsfalls kommt keine Verbindung mit der SIMATIC®- Software zustande.

Beispiel im I/O-Builder:



Die obige Einstellung im I/O-Builder erzeugt den folgenden CDB.

Beispiel Einstellung im CDB:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	COM	STRING[31]	´COM2:PTP_MPI,RS-232,38400,8,O,1´
=48.0		END_STRUCT	

8.4.2 Analog–Modem Parameter

Schlüsselwort: COM<n>:PTP_MPI,<Modem>

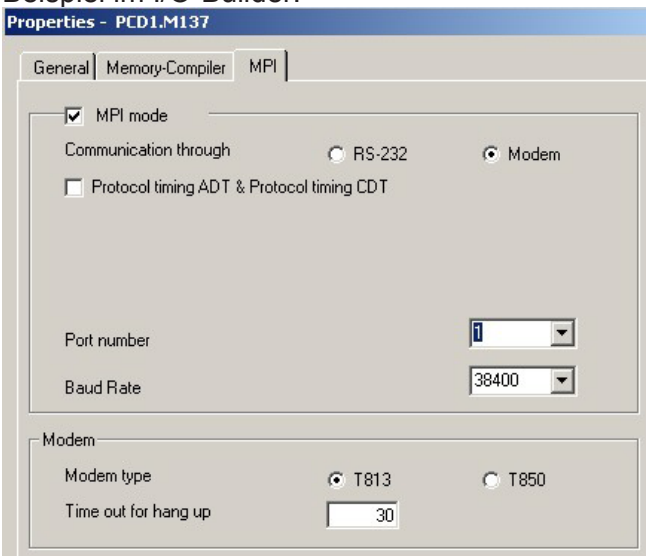
Parameter:
<n>: Schnittstellen-Nummer (1...5), siehe Tabelle Baudraten
<Modem>: Analog–Modem Typ (Modem oder T813)

Mit diesem CDB Eintrag wird folgender ModemInitString erzeugt:

“AT&FE0&C1&D3S0=1“

Die Initialisierung und Konfiguration der Schnittstelle als Analog-Modem ermöglicht nur, dass die Steuerung angerufen werden kann. Ausgehende Rufe sind nicht möglich. Als Standardeinstellung wird eine bestehende Verbindung nach 30 min automatisch unterbrochen, wenn kein Datenverkehr stattfindet. Dieser Wert kann durch den Parametertyp TIMEOUT verändert werden.

Beispiel im I/O-Builder:



Die obige Einstellung im I/O-Builder erzeugt den folgenden CDB.

Beispiel Einstellung im CDB:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	COM	STRING[18]	´COM1:PTP_MPI,T813´
=34.0		END_STRUCT	

8.4.3 ISDN–Modem Parameter

Schlüsselwort: COM<n>:PTP_MPI,T850,<Protokol>,<MSN>

Parameter:

<n>: Schnittstellen-Nummer (1...5), siehe Tabelle Baudraten

<Protokol>: Optional. ISDN- Protokoll. Folgende Protokollarten werden unterstützt:

- X.75-NL (default Einstellung)
- V.110
- V.120
- X.31B
- X.31D
- HDLC_ASYNC
- HDLC_TRANSPARENT
- BYTE_TRANSPARENT

Das T850 Modem unterstützt nur die ersten 3 Protokollarten. Für alle andern Protokollarten, muss ein externes ISDN-Modem verwendet werden.

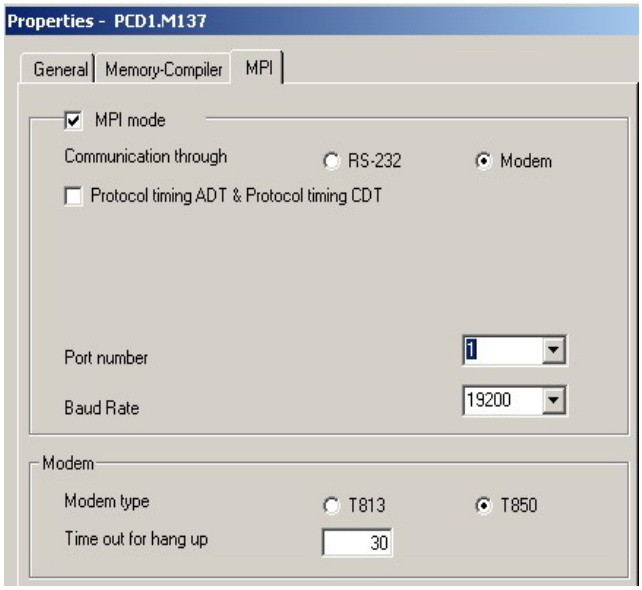
<MSN>: Optional. MSN Teilnehmernummer. Defaulteintrag = *.
Diese Nummer darf maximal 22 Ziffern lang sein. Werden mehr als 22 Ziffern eingetragen, so wird dieser Eintrag ignoriert.

Der CDB Eintrag „COM1:PTP_MPI,T850“ erzeugt folgenden ModemInitString:
“AT&FE0S0=1&D2B10#Z=*“

Die Initialisierung und Konfiguration der Schnittstelle als ISDN Modem ermöglicht nur, dass die Steuerung angerufen werden kann. Ausgehende Rufe sind nicht möglich. Als Standardeinstellung wird eine bestehende Verbindung nach 30 min automatisch unterbrochen, wenn kein Datenverkehr stattfindet. Dieser Wert kann durch den Parametertyp TIMEOUT verändert werden.

Mit dem I/O Builder kann nur der CDB Eintrag für die Default–Einstellung gewählt werden. Sollen andere Kommunikationsparameter für das ISDN-Modem eingestellt werden, so muss der CDB von Hand abgeändert und ergänzt werden. (Siehe nachfolgendes Beispiel.)

Beispiel im I/O-Builder:



Beispiel CDB-Eintrag:

Der CDB-Eintrag wurde abgeändert damit das V.110 Protokoll benutzt und die MSN-Nummer auf 21 gesetzt wird.

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	COM	STRING[26]	´COM1:PTP_MPI,T850,V.110,21´
=40.0		END_STRUCT	

8.4.4 Zusätzlicher Modemstring

Schlüsselwort: COM<n>:PTP_MPI,ADD_MODEM_STRING=<InitString>

Parameter:
<n>: Schnittstellen-Nummer (1...5), siehe Tabelle Baudraten
<InitString>: Zusätzlicher Modem String.

Der Zusätzliche Modem String wird an den ModemInitString angehängt.

Dieser CDB-Eintrag wird nur ausgewertet, wenn vorgängig ein Modem konfiguriert wurde.

Dieser CDB-Eintrag kann nicht mit dem I/O-Builder vorgenommen werden.

Beispiel:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	COM	STRING[23]	´COM1:PTP_MPI,T850,V.110´
+40.0	ADD_MODEM	STRING[37]	´COM1:PTP_MPI,ADD_MODEM_ STRING=AT&FB10´
=80.0		END_STRUCT	

Obiges Beispiel erzeugt den folgenden ModemInitString:

“AT&FE0S0=1&D2B0#Z=*AT&FB10“

8.4.5 Timeout Parameter

Als Standardeinstellung wird eine bestehende Modem Verbindung nach 30 min automatisch unterbrochen, wenn kein Datenverkehr stattfindet. Dieser Wert kann durch den Parametertyp TIMEOUT verändert werden. Das Timeout kann mit dem Wert 0 ausgeschalten werden.

Weiter können die Werte für Character Delay Time (CDT, Default =220 ms/1 s) und Answer Delay Time (ADT, Default = 2000 ms/10 s) für das Kommunikationsprotokoll eingestellt werden. Die Defaultwerte CDT und ADT sind bei serieller Programmierung 220 ms und 2000 ms. Bei Programmierung via Modem sind CDT und ADT auf 1 s und 10 s eingestellt.

TIMEOUT
Schlüsselwort: COM<n>:PTP_MPI,TIMEOUT=<timeout>

Parameter:
<n>: Schnittstellen-Nummer (1...5), siehe Tabelle Baudraten
<timeout>: Abbruch nach (0 = kein, 1...65'535 in min)

Character Delay Time
Schlüsselwort: COM<n>:PTP_MPI,CDT=<cdt>

Parameter:
<n>: Schnittstellen-Nummer (1...5), siehe Tabelle Baudraten
<cdt>: Character Delay Time (0...65'535 in ms)
Default = 220 ms/1 s (seriell/Modem)

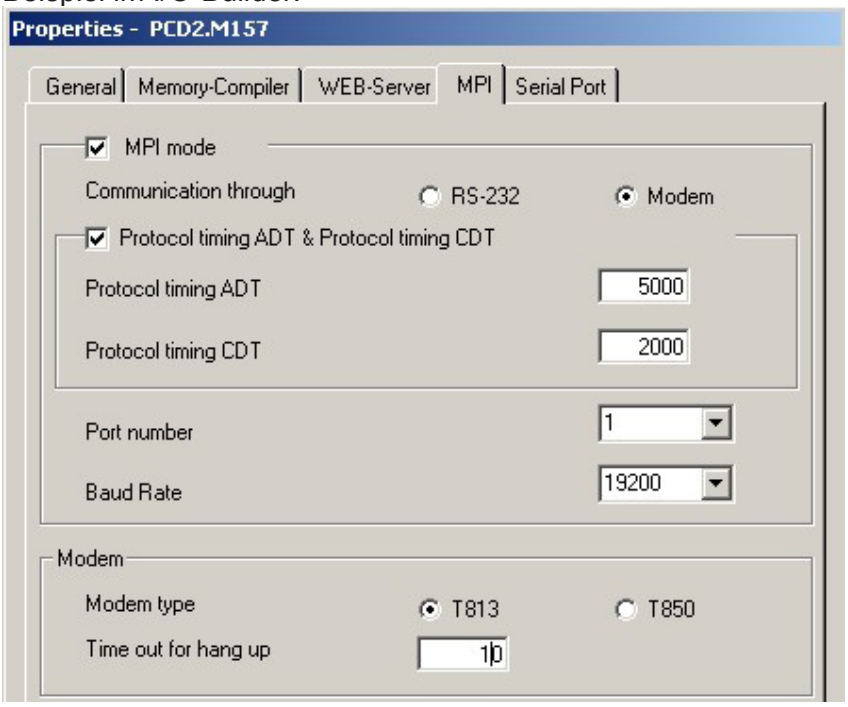
Beide Kommunikationspartner sollten die gleiche Einstellung haben. Die Einstellung wird intern in 10 ms Schritten umgerechnet.

Answer Delay Time
Schlüsselwort: COM<n>:PTP_MPI,ADT=<adt>

Parameter:
<n>: Schnittstellen-Nummer (1...5), siehe Tabelle Baudraten
<adt>: Answer Delay Time (0 ... 65'535 in ms)
Default = 2000 ms/10 s (seriell/Modem)

Beide Kommunikationspartner sollten die gleiche Einstellung haben. Die Einstellung wird intern in 10 ms Schritten umgerechnet.

Beispiel im I/O-Builder:



Die obige Einstellung im I/O-Builder erzeugt den folgenden CDB.

Beispiel Einstellung im CDB:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	COM	STRING[18]	´COM1:PTP_MPI,T813´
+34.0	TIMEOUT	STRING[24]	´COM1:PTP_MPI,TIMEOUT=10´
+60.0	ADT	STRING[22]	´COM1:PTP_MPI,ADT=5000´
+84.0	CDT	STRING[22]	´COM1:PTP_MPI,CDT=2000´
=108.0		END_STRUCT	

8.4.6 Einstellen der max. Baudrate auf Steckplatz B1 (B2)

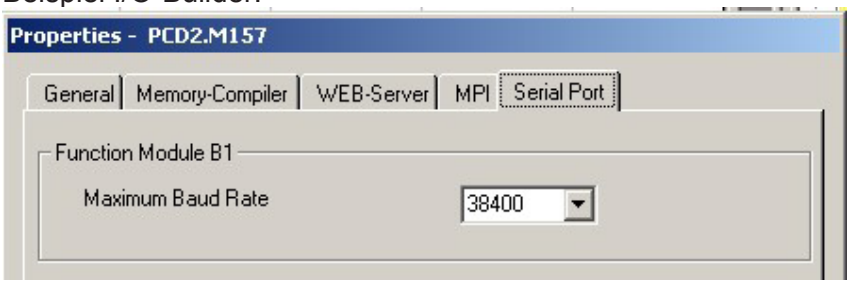
Die seriellen Schnittstellen auf den F5xx Modulen unterstüzen entweder 19’200 oder 38’000 Baud. Diese Einstellung gilt für beide Schnittstellen auf dem Modul.

Schlüsselwort: SLOT_B<n>:ENABLE_38400

Parameter:
<n>: Slot-Nummer (1...2)

Dieser CDB-Eintrag wird auf der PCD1.M137 nicht ausgewertet.

Beispiel I/O-Builder:



Die obige Einstellung im I/O-Builder erzeugt den folgenden CDB.

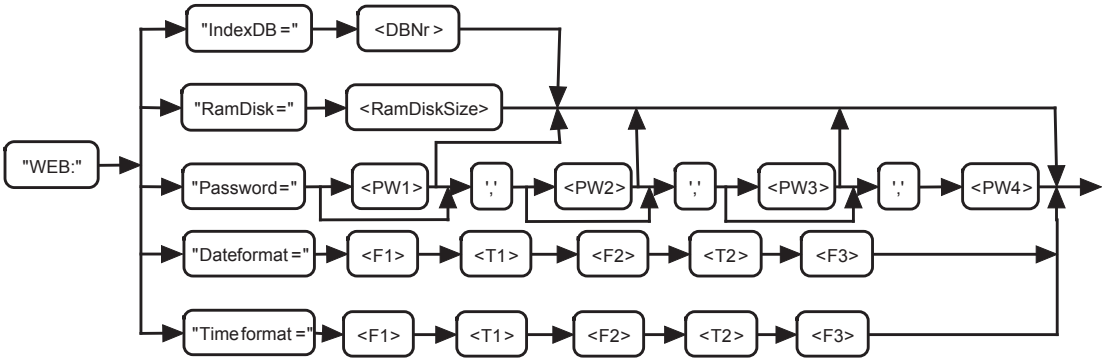
Beispiel Einstellung im CDB:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	SLOT_B1	STRING[20]	´SLOT_B1:ENABLE_38400´
=36.0		END_STRUCT	

8.5 Web-Server

Die Steuerungen PCD2.M157, PCD2.M177, PCD2.M487 und PCD3.Mxxx7 können mittels CDB zur Nutzung des Web-Servers initialisiert und konfiguriert werden. Die Einstellung der seriellen Schnittstelle wurden im vorherigen Kapitel beschrieben. Weiteres zum Thema Web-Server ist im Handbuch Web-Server 26/775 beschrieben.

Das nachfolgende Syntaxdiagramm gibt einen Überblick über die Konfigurationsmöglichkeiten des Web-Servers. In den folgenden Kapiteln werden die einzelnen Parameter näher erklärt.



Die Web-Server Konfiguration wird nach jedem Stop zu Run Wechsel ausgewertet. Ausnahme: der Parameter RamDisk benötigt ein “Netz Ein” damit er ausgewertet wird.

Parameter IndexDB=:
<DBNr> = Nummer des IndexDB: Dieser Datenbaustein enhält das Verzeichnis für alle Dateien, die als Datenbausteine (DBNr + n) in die xx7-Steuerung geladen wurden.

Parameter RamDisk=:
<RamdiskSize> = Grösse der internen Ramdisk. Die Standardeinstellung ist 2 kByte. Bei Bedarf kann diese vergössert werden. Der Zusätzliche Bedarf an Ramdisk wird auf der PCD2.M157 und PCD2.M177 dem Step®7-Speicher entnommen. Auf der PCD2.M487 wird der zusätzliche Speicherbedarf für die Ramdisk dem Systemspeicher entnommen. Siehe auch Speicher / Flash Funktionen.

Der Parameter RamDisk wird nur nach einem Netz Ein ausgewertet.

Auf den PCD3.Mxxx7 Systemen können spezielle Flash Devices für die Speicherung der Webprojekte verwendet werden.

Parameter Password=:
<PWx> = Festlegung von bis zu 4 Passwörter (4 Level) durch “,“ getrennt. (, , = Platzhalter). Bei den Passwörtern wird nicht zwischen Gross- und Kleinschreibung unterschieden. Z.b. bedeutet “SBC“ und “sbc“ das gleiche Passwort. Jedes Passwort kann bis zu 16 Zeichen lang sein, ausgenommen Komma und Leerzeichen. Es ist nicht notwendig, ein Passwort zu definieren.

Parameter Dateiformat=:
Der Web-Server zeigt das Datum aus der Step7 DATA_AND_TIME (DT) Variablen defaultmässig wie folgt an: DD.MM.JJJJ (Beispiel: 10.09.2001). Dieses Format kann im CDB geändert werden:

WEB:DATEFORMAT=<F1><T1><F2><T2><F3>

F1,F2,F3 D -> Anzeige Tag ohne Füllnull
DD -> Anzeige Tag mit Füllnull
M -> Anzeige Monat ohne Füllnull
MM -> Anzeige Monat mit Füllnull
YY -> Anzeige Jahr
YYYY -> Anzeige Jahr, 4-stellig

T1,T2: Gültiger Separator.

Ein gültiger Separator muss in folgendem Bereich liegen: Dezimal Werte von ASCII Charaktern zwischen 33 und 47 oder zwischen 58 und 64.

Bei fehlerhaften Eingaben wird das Default-Format verwendet.

Beispiele:
WEB:DATEFORMAT=D/M/YY
WEB:DATEFORMAT=YYYY.MM.DD

Parameter Timeformat=:
Der Web-Server zeigt die Zeit aus der STEP7 DATA_AND_TIME (DT) Variablen defaultmässig wie folgt an: HH:MM:SS (Beispiel: 11:55:00). Dieses Format kann im CDB geändert werden:

WEB:TIMEFORMAT=<F1><T1><F2><T2><F3>

F1,F2,F3 H -> Anzeige Stunden ohne Füllnull
HH -> Anzeige Stunden mit Füllnull
M -> Anzeige Minuten ohne Füllnull
MM -> Anzeige Minuten mit Füllnull
S -> Anzeige Sekunden ohne Füllnull
SS -> Anzeige Sekunden mit Füllnull

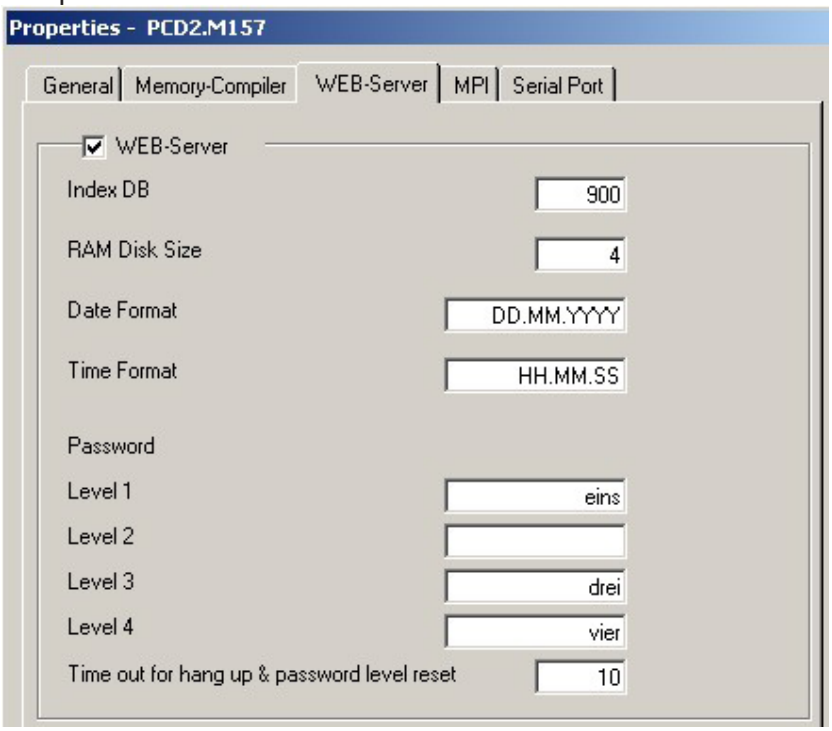
T1,T2 Gültiger Separator.

Ein gültiger Separator muss in folgendem Bereich liegen: Dezimal Werte von ASCII Charaktern zwischen 33 und 47 oder zwischen 58 und 64.

Bei fehlerhaften Eingaben wird das Default Format verwendet.

Beispiel:
WEB:TIMEFORMAT=H/M/S
WEB:TIMEFORMAT=HH.MM.SS

Beispiel im I/O-Builder:



Die obige Einstellung im I/O-Builder erzeugt den folgenden CDB.

Beispiel Einstellung im CDB:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	IndexDB	STRING[15]	´WEB:IndexDB=900´
+32.0	RamDisk	STRING[13]	´WEB:RamDisk=4´
+48.0	DateFormat	STRING[25]	´WEB:DateFormat=DD.MM.YYYY´
+76.0	TimeFormat	STRING[23]	´WEB:TimeFormat=HH.MM.SS´
+102.0	Password	STRING[28]	´WEB:Password=eins,,drei,vier´
=132.0		END_STRUCT	

8.6 Peripherie Zugriff im OB100

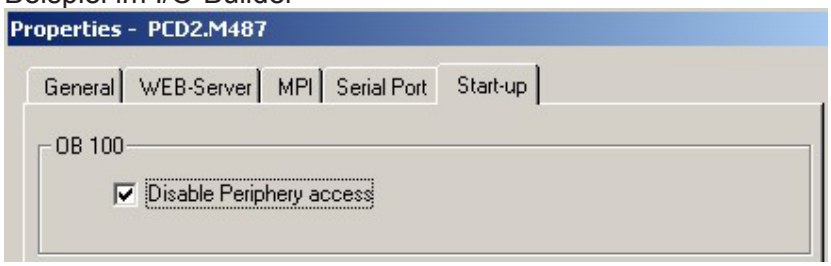
Auf den Systemen PCD1 und PCD2.M1x7 ist das Signal I/O-Reset während des Aufstartens gesetzt. Dies hat zur Folge, dass im OB100 nicht auf die Peripheriemodule zugegriffen werden kann. Auf der PCD2.M487 und PCD3.Mxxx7 ist dieses Signal während des Aufstartens nicht gesetzt. Mit dem CDB Eintrag Peripherie = Disabled kann der Anwender auf der M487 und der PCD3 das gleiche Aufstartverhalten wie auf den älteren Systemen einstellen.

Schlüsselwort: OB100:Peripherie=[Disabled | Enabled]

Dieser CDB-Eintrag wird unter folgenden Bedingungen ausgewertet:

- bei jedem Stop - Run Übergang
- nur auf der PCD2.M487 und PCD3.Mxxx7

Beispiel im I/O-Builder



Die obige Einstellung im I/O-Builder erzeugt den folgenden CDB.

Beispiel Einstellung im CDB:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	Peripherie	STRING[25]	´OB100:Peripherie=Disabled´
=42.0		END_STRUCT	

8.7 Konfiguration Profi-S-IO-Master

Die PCD2.M487 unterstüzt verschiedene DP-Master Schnittstellen. Folgende Mög-li-chkeiten bestehen:

- F750 Modul auf Slot 1
- F750 Modul auf Slot 2
- DP-Master Kommunikation auf dem S-Net/MPI Port (Profi-S-IO-Master)

Der Profi-S-IO-Master kann nur aktiv werden, falls im CDB der Parameter Profi-S-IO:Enable eingetragen wird. Die folgende Tabelle zeigt alle unterstützten Kombinationen der DB-Konfiguration.

Nr	Profi-S-IO-Flag	DP-Slave	Profi-S-IO-Master	F750 auf B1	F750 auf B2	Kommentar
1	ENABLE	-	< V2.0 Int or CP	-	-	Falls kein F750 Modul vorhanden ist.
2	DISABLE	< V2.0 Int	-	-	-	
3	ENABLE	-	< V2.0 Int	< V2.0 CP	-	Falls nur ein F750 Modul gesteckt ist.
4	ENABLE	-	< V2.0 Int	-	< V2.0 CP	
5	DISABLE	-	-	< V2.0 Int or CP	-	
6	DISABLE	-	-	-	< V2.0 Int or CP	
7	don‘t care	V2.0 Int	-	V2.0 CP	-	
8	don‘t care	V2.0 Int	-	-	V2.0 CP	
9	DISABLE[1)]	-	-	< V2.0 Int	< V2.0 CP	Falls 2 F750 Module gesteckt sind.

[1)] Falls zwei F750 Module gesteckt sind und im CDB das Profi-S-IO-Flag auf Enable gesetzt wurde, so wird das F750 Modul auf Steckplatz 2 nicht benutzt. Dies entspricht der Kombination in Zeile 3 in obiger Tabelle.

Weitere Informationen findet man in den Handbüchern “Profibus DP master and slave module documentation” und “Preliminary version of the documentation regarding FDL Master-Master communication”.

Schlüsselwort: Profi-S-IO:[Disable | Enable]

Dieser CDB-Eintrag wird unter folgenden Bedingungen ausgewertet:

- bei jedem Stop - Run Übergang
- nur auf der PCD2.M487 und PCD3.M5xx7

Beispiel Einstellung im CDB:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	SIOMaster	STRING[17]	´Profi-S-IO:Enable´
=34.0		END_STRUCT	

8.8 Konfiguration MPI auf Port 2

Auf der PCD3.M6347 ist die MPI-Schnittstelle durch die CAN Schnittstelle ersetzt worden. Wenn es trotzdem erforderlich ist, auf dieser PCD3 Plattform eine MPI-Schnittstelle zur Verfügung zu stellen ist dies durch einen CDB-Eintrag möglich. Anstelle der RS-485-Schnittstellen auf Port 2 kann diese Schnittstelle auf MPI umgeschalten werden.

Weitere Informationen findet man in den Handbüchern “Profibus DP master and slave module documentation” und “Preliminary version of the documentation regarding FDL Master-Master communication”.

Schlüsselwort: MPI:PORT2_ON

Dieser CDB-Eintrag wird unter folgenden Bedingungen ausgewertet:

- bei jedem Power on Übergang
- nur auf der PCD3.M6347

Beispiel Einstellung im CDB:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Identificator	STRING[12]	´SBC xx7 CDB´
+14.0	COM	STRING[17]	´MPI:PORT2_ON´
=34.0		END_STRUCT	

A Anhang

A.1 Symbole

	Diese Symbol verweist den Leser innerhalb eines Handbuches auf weiterführende Informationen in diesem oder einem anderen Handbuch, oder in technischen Informationsbroschüren. In der Regel besteht kein direkter Link zu diesen Dokumenten.
	Dieses Symbol warnt den Leser vor dem Risiko elektrischer Entladung durch Berühren. **Empfehlung:** Bevor Sie in Kontakt mit elektronischen Bauteilen kommen, sollten Sie zumindest vorher den Minuspol des Systems (Gehäuse der PGU-Buchse) berühren. Besser ist es, permanent mit einer Erdungslasche am Handgelenk mit dem Minuspol verbunden zu sein.
	Dieses Zeichen steht neben Anweisungen, die befolgt werden müssen.
	Erklärungen neben diesem Zeichen sind nur für die Saia PCD® Classic Serie gültig.
	Erklärungen neben diesem Zeichen sind nur für die Saia PCD® xx7 Serie gültig.

A

A.2 Kontakt

Saia-Burgess Controls AG
Bahnhofstrasse 18
3280 Murten

Telefon +41 26 672 72 72
Telefax +41 26 672 74 79

E-mail: support@saia-pcd.com
Homepage: www.saia-pcd.com
Support: www.sbc-support.com

Postadresse für Rücksendungen von Kunden des Verkauf Schweiz:

Saia-Burgess Controls AG
Service Après-Vente
Bahnhofstrasse 18
3280 Murten

A