Manual

SAIA BURGESS CONTROLS

Integrated system functions - Series xx7

Document 26/797; Edition EN05 │ 2013-12-19

0 Inhalt

1 Introduction

2 Standard system functions

3 Onboard functions

4 Communication functions

5 Memory

0.1 Document History

Version	Date	Changed	Remarks
EN04	2006-04-30	-	Initial version
EN05	2013-12-19	-	New logo and new company name

0.2 Brands and trademarks

Saia PCD® and Saia PG5®
are registered trademarks of Saia-Burgess Controls AG.

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

Technical modifications are based on the current state-of-the-art technology.

Published in Switzerland

1 Introduction

Saia PCD® Series xx7 controllers are equipped with several system functions. These functions fall into three categories:

- Standard system functions (Component number < 200)
- Extended system functions (Component number ≥ 200)
- Configuration Data Block (CDB)

The standard system functions are the System Function Blocks (SFB) and System Function Calls (SFC); their functions are equivalent to the Siemens SIMATIC® S7 controllers® of the same name.

The extended system functions are Saia Burgess Controls (SBC) specific SFBs and SFCs, used to program the onboard functions.

The SBC specific SFBs and SFCs, which have wide-ranging functionality (e.g. serial communication), are only mentioned here for the sake of completeness and are dealt with in more detail in other sections, or other manuals.

The Configuration Data Block is a data block that is run once when the CPU is started up. It can be used to change a variety of settings for interfaces, memory allocation and system functions.

1.1 Extended system functions

All the extended system functions are to be found in the operating system. They can be copied from the CPU to the offline project. Alternatively, the functions can be copied from the SBC library. The SBC library can be downloaded free of charge from http://www.sbc-support.com.

Not all extended system functions are available in the SBC library. If required, these functions can be copied from the CPU operating system to the offline project. SFBs are not generally available in the SBC library.

All system functions set the BIE bit in the status word to 0 if the function terminated with an error. Some system functions also return a further error code. If the function executes without any error, the BIE bit is set to 1.

2 Standard system functions

All the standard system functions supported are to be found in the operating system. Alternatively, they may be copied from the CPU to the offline project, or copied from the appropriate Step®7 library.

!

Not all system functions are supported by Siemens®.

The supported system functions are set out in the list below. A detailed description of the components is held within the Help function of the SIMATIC® manager.

2.1 IEC Timer and IEC Counter

SFB No.	Name	Designation	Remarks
0	CTU	Forward counter	
1	CTD	Backward counter	
2	CTUD	Forward/backward counter	
3	TP	Pulse	
4	TON	Switch-on delay	
5	TOF	Switch-off delay	

2.2 Communication via planned connections

SFB No.	Name	Designation	Remarks
12	BSEND	Send block-based data	
13	BRCV	Receive block-based data	
14	GET	Read data from partner	

A description of components SFB12 - SFB14 can be found in Manual 26/794: Serial Communication.

2.3 System diagnostics

SFC No.	Name	Designation	Remarks
6	RD_SINFO	Read start information	
52	WR_USMSG	Entry in diagnostics buffer	

2.4 CPU clock and run time meter

SFC No.	Name	Designation	Remarks
0	SET_CLK	Set clock time	
1	READ_CKL	Read clock time	
2	SET_RTM	Set run time meter	
3	CTRL_RTM	Control run time meter	
4	READ_RTM	Read run time meter	
64	TIME_TCK	Read system time	

2.5 Manipulation of data blocks

SFC No.	Name	Designation	Remarks
20	BLKMOV	Copy data block	
21	FILL	Pre-fill data block	
22	CREAT_DB	Create data block	
23	DEL_DB	Delete data block	
24	TEST_DB	Test data block	
25	COMPRESS	Compress memory	
44	REPL_VAL	Enter replacement value	

2.6 Decentralized peripherals (PROFIBUS-DP)

SFC No.	Name	Designation	Remarks
7	DP_PRAL	Trigger process alarm on DP Master	PCD2.M487 only
11	DPSYN_FR	SYNC/FREEZE	
13	DPNRM_DG	Read diagnostic data	
14	DPRD_DAT	Read slave data	
15	DPWR_DAT	Write slave data	

2.7 Program control

SFC No.	Name	Designation	Remarks
43	RE_TRIGR	Trigger cycle time monitoring	
46	STP	Switch to STOP status	
47	WAIT	Delay the processing of the user program	PCD2.M487 and PCD3 only

2.8 Update process map

SFC No.	Name	Designation	Remarks
26	UPDAT_PI	Update process map for outputs	
27	UPDAT_PO	Update process map for outputs	

2.9 Interrupt events

SFC No.	Name	Designation	Remarks
28	SET_TINT	Set clock time alarm	
29	CAN_TINT	Cancel clock time alarm	
30	ACT_TINT	Activate clock time alarm	
31	QRY_TINT	Query clock time alarm	
32	SRT_DINT	Start delay alarm	
33	CAN_DINT	Cancel delay alarm	
34	QRY_DINT	Query delay alarm	
36	MSK_FLT	Mask synchronous fault	
37	DMSK_FLT	De-mask synchronous fault	

38	READ_ERR	Read event register
39	DIS_IRT	Disable asynchronous fault
40	EN_IRT	(Re)enable asynchronous fault
41	DIS_AIRT	Delay asynchronous fault
42	EN_AIRT	Enable (upgrade) asynchronous fault

2.10 Record transfer

SFC No.	Name	Designation	Remarks
58	WR_REC	Write record	(SIWAREX only per DP standard)
59	RD_REC	Read record	(SIWAREX only per DP standard)

2.11 Global data communication

SFC No.	Name	Designation	Remarks
60	GD_SND	Send GD packet	
61	GD_RCV	Receive GD packet	

2.12 Communication via unplanned connections

SFC No.	Name	Designation	Remarks
65	X_SEND	Send data (external)	(not for PCD1.M137)
66	X_RCV	Receive data (external)	(not for PCD1.M137)
67	X_GET	Read data (external)	(not for PCD1.M137)
68	X_PUT	Write data (external)	(not for PCD1.M137)
69	X_ABORT	Cancel external connection	(not for PCD1.M137)

3 Onboard functions

The PCD2 Series xx7 controllers are equipped with a number of onboard functions. These are accessed via system functions. The onboard functions include:

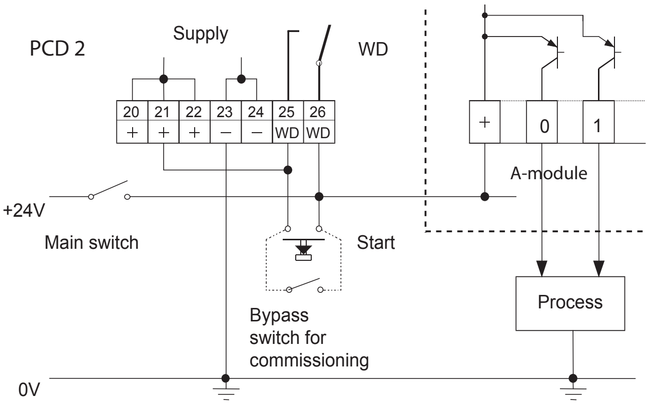
- Hardware watchdog, SFC239 WDOG
- Interrupt inputs, SFC250 INP_INT, SFC256 CONF_AL (PCD2.M487)
- Unidirectional counter, SFC251 INTCNTR, SFC252 READCNTR
- Bidirectional counter, SFC248 INTDIR, SFB256 BOARDCNT (PCD2.M487)
- Integrated SSI interface, SFC253 READ_SSI, SFC254 GRAY2BIN

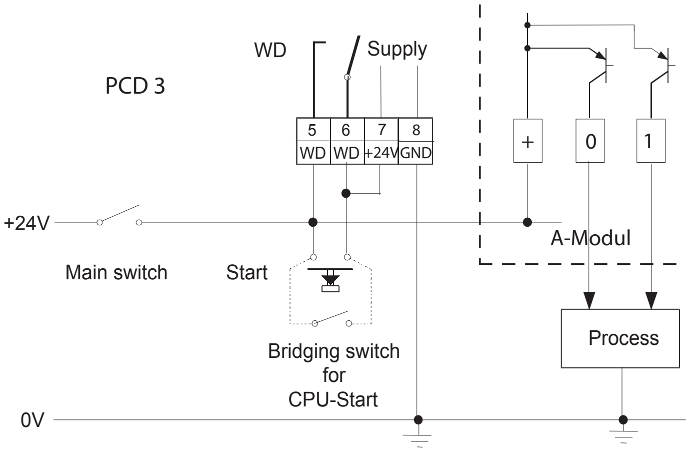
3.1 Hardware watchdog (SFC239 WDOG)

The PCD2 CPU has an integrated watchdog relay, which can be used e.g. to regulate the load voltage. After activation, the watchdog must be triggered within approx. 350ms (320...380ms) by a call to SFC239 (WDOG), to keep the watchdog relay closed. This guarantees a high degree of security even when the PLC processor fails.

The watchdog relay is present only on controllers of type PCD2/PCD3. This component is not supported by the PCD1 operating system.

Switching example and connection layout PCD2 and PCD3:





Operation

Provided that the PLC cycle time does not exceed 350ms (320...380ms), a single call within the cyclic PLC program is sufficient to keep the watchdog relay closed. With a longer cycle time, SFC239 can be called more than once in the PLC cycle, or programmed within a suitably parameterized alarm function (e.g. OB35, run every 300ms).

Example:

```
Network 1: If SPS in RUN state, keep watchdog relay closed

      CALL  “WDOG“                  // Call SFC239
```

Address 255 is reserved for the watchdog. On the PCD2.M177 only, address 511 is also reserved for the watchdog. Regardless of whether the watchdog is used, not all PCD2/PCD3 modules can be used without restriction in module slot 16. The table below gives a list of PCD2 modules showing whether or not they can be used in module slot 16.

PCD2/PCD3 modules	Usable in slot 16?
16 dig. inputs	Yes
16 dig. outputs	No
W1xx	Yes
W2xx	No
W3xx	No (exc. W3x5)
W4xx	Yes
W5xx inputs	Yes
W5xx outputs	No
W6xx	No (exc. W6x5)
H modules	No

3.2 Interrupt inputs

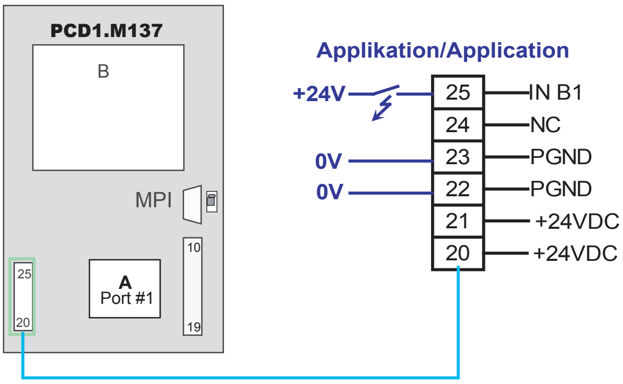
The integrated interrupt inputs allow rapid response times to be achieved independently of the PLC cycle. After the interrupt is triggered, process alarm OB40 to OB47 is called.

On the PCD2, the functionality for the interrupt inputs and the counter functions utilize the same components. Either the interrupt functionality or the counter functionality can be used.

	Number of interrupt inputs	Programmable with SFC	Process alarms
PCD1	1	250	OB40
PCD2.M1x7	2	250	OB40
PCD2.M487	4	256	OB40 to OB47
PCD3.Mxxx7	2	256	OB40 to OB47

3.2.1 Switching example and connection layout

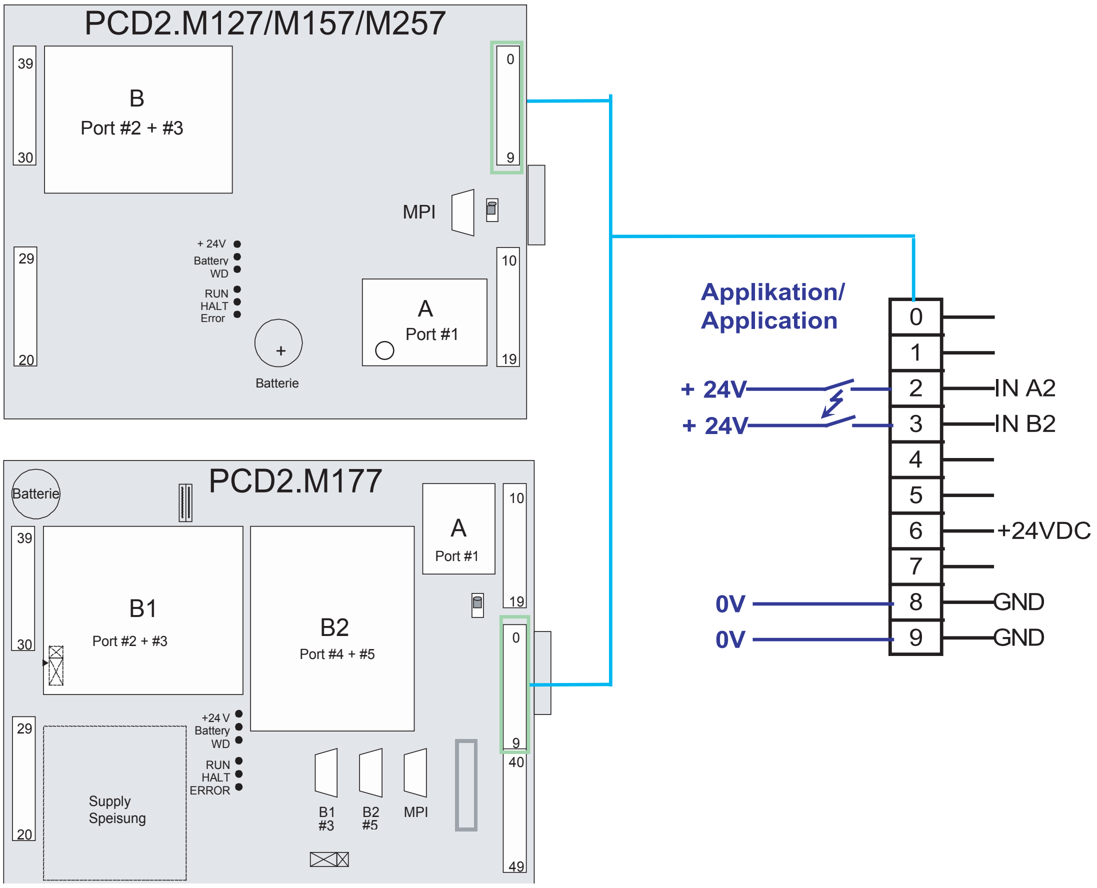
Switching example - PCD1



PCD1 mapping:

Name	Terminal	Signal	Interrupt triggered by:
Interrupt input 1	25	IN B1	Positive edge

Switching example - PCD2.M1x7



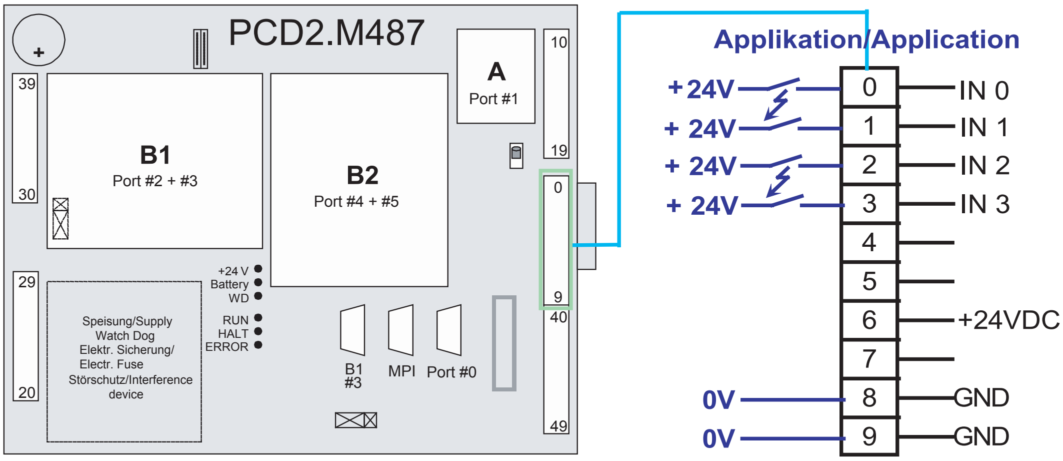
PCD2.M1x7 mapping

Name	Terminal	Signal	Interrupt triggered by:
Interrupt input 0	2	IN A2	Negative edge
Interrupt input 1	3	IN B2	Positive edge

Restriction:

If Terminal 3 (Interrupt input 1) is running at 24V, a negative edge on Terminal 2 (Interrupt input 0) will not trigger an interrupt. This means that it is not possible to capture the positive and negative edge of a signal.

Switching example - PCD2.M487

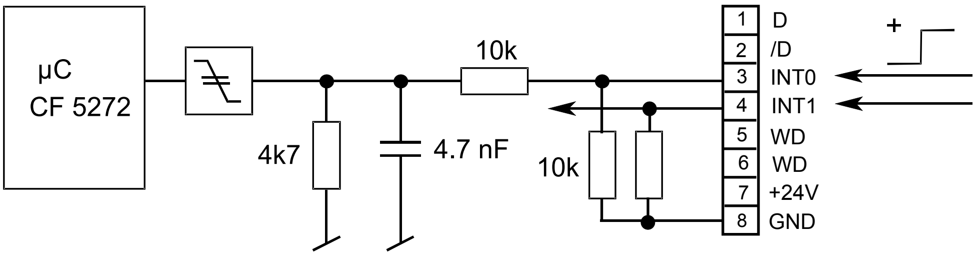


PCD2.M487 mapping

Name	Terminal	Signal	Interrupt triggered by:
Interrupt input 0	0	IN0	Positive edge
Interrupt input 1	1	IN1	Positive edge
Interrupt input 2	2	IN2	Positive edge
Interrupt input 3	3	IN3	Positive edge

The four interrupt inputs can be configured and enabled / disabled independently.

Switching example - PCD3.Mxxx7



PCD3.Mxxx7 mapping

Name	Terminal	Signal	Interrupt triggered by:
Interrupt input 0	3	INT0	Positive edge
Interrupt input 1	4	INT1	Positive edge

The two interrupt inputs can be configured and enabled / disabled independently.

3.2.2 Enable / disable interrupt (SFC 250 INP_INT)

SFC250 (INP_INT) is used to enable or disable the interrupt inputs. On the PCD2. M1x7, both interrupts are disabled / enabled together. A single SFC250 call in the application program is sufficient for enabling / disabling.

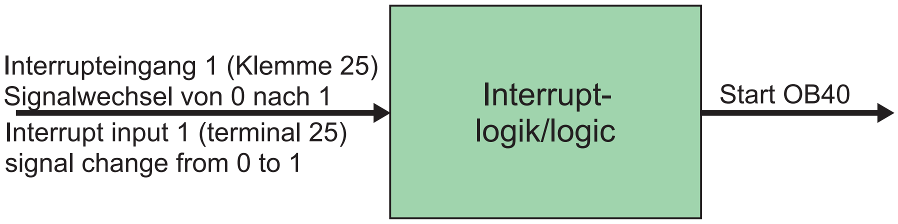
SFC250 parameters:

Parameter	Declara-tion	Type	Range	Description
Enable (IN0)	Input	BOOL	TRUE/ FALSE	TRUE: Enable interrupts FALSE: Disable interrupts
Ret_Val	Output	WORD	0	0: No error

The SFC250 functionality is not available on the PCD2.M487 and the PCD3.Mxxx7. Instead, function SFC 256 CONF_AL can be used.

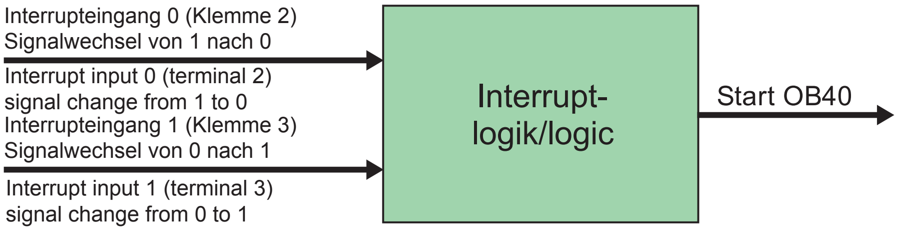
PCD1:

After enabling the interrupt input, OB40 is called automatically where a signal change from 0 to 1 occurs at Interrupt input 1 (Terminal 25).


Interrupteingang 1 (Klemme 25)
Signalwechsel von 0 nach 1
Interrupt input 1 (terminal 25)
signal change from 0 to 1
Interrupt-
logik/logic
Start OB40

PCD2:

If the interrupt inputs are enabled, a signal change from 1 to 0 at Interrupt input 0 (Terminal 2) or from 0 to 1 at Interrupt input 1 (Terminal 3) will cause OB40 to be called automatically.


Interrupteingang 0 (Klemme 2)
Signalwechsel von 1 nach 0
Interrupt input 0 (terminal 2)
signal change from 1 to 0
Interrupteingang 1 (Klemme 3)
Signalwechsel von 0 nach 1
Interrupt input 1 (terminal 3)
signal change from 0 to 1
Interrupt-
logik/logic
Start OB40

OB40 processing can determine at which input the interrupt was triggered. This is done by analysing the local data byte “OB40-SRT-INF”. The following mapping applies:

PCD2:	OB40_SRT_INF = B#16#41 → Interrupt input 0 (Terminal 2)
	OB40_SRT_INF = B#16#42 → Interrupt input 1 (Terminal 3)
PCD1:	OB40_SRT_INF = B#16#42 → Interrupt input 1 (Terminal 25)

Programming examples:

Calling SFC250:

```
Network 1: Enable interrupt
      U     E 0.7                     // Activate/de-activate
      FP    M 1.7
      SPBN  int0                      // Pulse in
// Enable interrupt
      CALL  “INP_INT“                 // SFC250
       IN0    :=TRUE                  // Enable interrupt
       RET_VAL:=MW250                 // Return value

Network 2: Disable interrupt
int0: U     E 0.7                     // Activate/de-activate
      FN    M 1.6
      SPBN  int2                      // Pulse out
// Disable interrupt
      CALL  “INP_INT“                 // SFC250
       IN0    :=FALSE                 // Disable interrupt
       RET_VAL:=MW250                 // Return value
int2: NOP   0
```

Determining the interrupt source in OB40:

```
OB40:  “Hardware interrupt”
Network 1: Analyse which interrupt?
      L     #OB40_STRT_INF
      L     B#16#41                   // Interrupt input 0?
      ==I
      SPB   AlA0                      // Jump to alarm 0
      TAK
      L     B#16#42                   // Interrupt input 1?
      ==I
      SPB   AlA1                      // Jump to alarm 1
      BEA

Network 2: Interrupt from input 0
// If interrupt 0 was triggered, output A0.2 should be
// reset.
AlA0: L     AB     0                  // Current state of output byte 0
      L     2#11111011                // Mask out Bit 2
      UW                              // and
      T     PAB    0                  // reset immediately
      BEA

Network 3: Interrupt from input 1
// If interrupt 1 was triggered, output A0.3 should be
// reset.
AlA1: L     AB     0                  // Current state of output byte 0
      L     2#11110111                // Mask out Bit 3
      UW                              // and
      T     PAB    0                  // reset immediately
      BEA
```

If a further interrupt is triggered during handling of the OB40 process alarm, a timer error will occur and OB80 will be called automatically.

3.2.3 Configure interrupt inputs (SFC 256 CONF_AL)

On the M487 and PCD3.Mxxx7, the interrupt inputs can be configured independently. For each interrupt input, the OB to be called is configured, and the start information held in this OB. A single SFC256 call in the application program is sufficient for configuration and enabling / disabling.

The SFC256 functionailty is not available on the PCD1 and the PCD2.M1x7. Instead, function SFC 250 INP_INT can be used.

SFC256 parameters:

Parameter	Declaration	Type	Range	Description
IRQ_NO (IN0)	Input	INT	0...3[1]	Interrupt input, corresponds to Terminals IN0..IN3.
REQ_TYPE (IN1)	Input	BOOL	TRUE/ FALSE	TRUE: Enable interrupts FALSE: Disable interrupts
OB_NR (IN2)	Input	INT	40...47	Number of the OB to be called in case of an interrupt.
OB_INFO (IN3)	Input	WORD	XXXX[2]	Start information for the interrupt OB. The value is held in local data.
Ret_VAL	Output	WORD	YYYY[3]	Error message: 0x0000: No error 0x8080: IRQ_NO outside the permitted range (0..3) 0x8081: OB_NR outside the permitted range (40..47)

[1] PCD3.Mxxx7 0...1 Interrupt input matches terminals INT0...INT1

[2] [3] 2-byte ranges from 0x0000 to 0xFFFF

[4] PCD3.Mxxx7, permitted range 0...1

By evaluating the local data word “OB_4x_MDL_ADDR” against the planned start information, the interrupt source can be determined when processing the interrupt OB.

Programming example:

Calling SFC256

```
Network 1: Enable interrupt 0
      U     E 0.7               // Enable/disable
      FP    M 1.7
      SPBN  int0                // Pulse in
// Enable interrupt 0
      CALL  "CONF_AL"
       IN0     :=0              // Interrupt input 0, Terminal IN0
       IN1     :=TRUE           // Enable
       IN2     :=41             // for positiver edge, OB41 called
       IN3     :=W#16#CAFE      // Start information in OB41
       RET_VAL:=MW250           // MW contains error information
```

```
Network 2: Disable interrupt 0

int0: U     E 0.7                    // Enable/disable
      FN    M 1.6
      SPBN  int1                     // Pulse out
// Disable interrupt 0
      CALL  “CONF_AL“
       IN0    :=0                  // Interrupt input 0, Terminal IN0
       IN1    :=FALSE              // Disable
       IN2    :=41                 // This info not analyzed here
       IN3    :=W#16#CAFE          // This info not analyzed here
       RET_VAL:=MW250              // MW contains error information

int1: NOP   0
```

Analysis in interrupt OB41

```
OB41:  “Hardware Interrupt”
Network 1:

      L     #OB41_MDL_ADDR           // Start information
      L     W#16#CAFE                // Planned start information for
      ==I                            // Interrupt 0
      SPB   Int0                     // Jump to interrupt 0
      BEA

Network 2: Interrupt 0 triggered

Int0: NOP   0
// Execute interrupt function
```

3.3 Onboard counters

The onboard inputs can be used to implement counters independently of the PLC cycle. When a reference value is reached, an OB process alarm is called.

On the PCD2, the functionality for the interrupt inputs and the counter functions utilize the same components. Either the interrupt functionality or the counter functionality can be used.

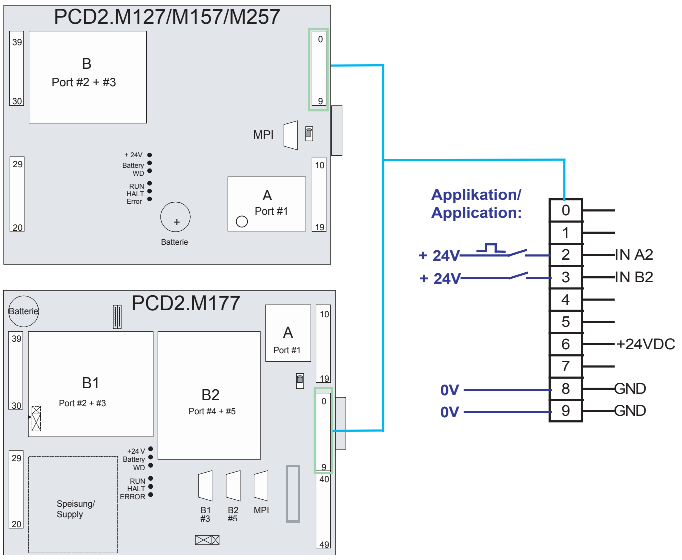
	Number of counters	Programmable	Process alarms
PCD1	0	-	-
PCD2.M1x7	1	SFC251 / 252	OB40
PCD2.M1x7	1	SFC248	OB41
PCD2.M487	2	SFB256	OB40 to OB47 / onboard output
PCD3.Mxxx7	1	SFB256	OB40 to OB47

3.3.1 Unidirectional counter (SFC251 INTCNTR, SFC252 READCNTR)

With the integrated interrupt inputs, a unidirectional counter can be used up to approx. 5 kHz. This counter is unidirectional, i.e. counting can only be in one direction. When up to 2 reference values are reached, an interrupt is triggered. After triggering the interrupt, process alarm OB40 is called. An external enable signal can be used to disable or activate the counter. The counter status can be read using SFC252.

The unidirectional counter functionality is **not** available on the PCD1, the PCD2.M487 and the PCD3.Mxxx7. On the PCD2.M487 and the PCD3.Mxxx7, the SFB256 BOARDCNT function can be used as an alternative.

Switching example and connection layout:

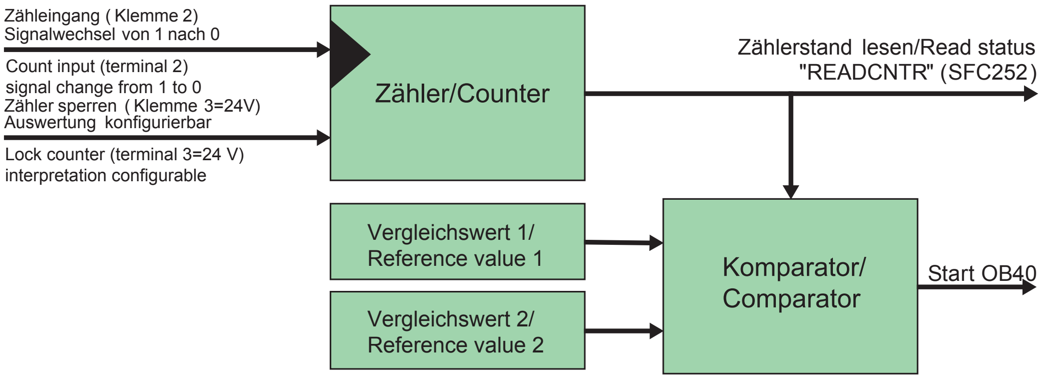


PCD2.M1x7 mapping

Name	Terminal	Signal	Description
Counter input	2	IN A2	Counter pulse on signal change from 1 to 0.
Enable counter	3	IN B2	Analyzed, if configured in program

Operation:

If the unidirectioal counter is started and configured, its basic operation is as follows:


Zähleingang (Klemme 2)
Signalwechsel von 1 nach 0
Count input (terminal 2)
signal change from 1 to 0
Zähler sperren (Klemme 3=24V)
Auswertung konfigurierbar
Lock counter (terminal 3=24 V)
interpretation configurable
Zähler/Counter
Zählerstand lesen/Read status
"READCNTR" (SFC252)
Vergleichswert 1/
Reference value 1
Vergleichswert 2/
Reference value 2
Komparator/
Comparator
Start OB40

SFC251 (INTCNTR) is used to configure and start or stop the unidirectional counter. A single SFC251 call in the application program is sufficient to configure / start the counter.

SFC251 parameters:

Parameter	Declaration	Type	Range	Description
START	Input	BOOL	TRUE/ FALSE	TRUE: Start counter FALSE: Stop counter
ENABLE	Input	BOOL	TRUE/ FALSE	Signal at Terminal 3: TRUE: Analyze FALSE: Do not analyze
INT2	Input	BOOL	TRUE/ FALSE	When VALUE2 reached: TRUE: Call OB40 FALSE: Do not call OB40
INT1	Input	BOOL	TRUE/ FALSE	When VALUE1 reached: TRUE: Call OB40 FALSE: Do not call OB40
VALUE2	Input	WORD	0x0002 ... 0xFFFF	Reference value 2: VALUE2 must be greater than VALUE1
VALUE1	Input	WORD	0x0002 ... 0xFFFF	Reference value 1: VALUE2 must be greater than VALUE1
RET_VAL	Output	WORD	YYYY[1)]	Error message: 0x0000: No error 0x00FE: Invalid reference value

[1)] 2-byte ranges from 0x0000 to 0xFFFF.

Programming example:
Calling SFC251

```
Network 1: Configure counter

      U     E      0.7           // Configuration handover
      FP    M      1.7
      SPBN  NOKO                 // Impulse - configure counter
      CALL  "INTCNTR"            // Call SFC251
       START  :=E0.0             // 1 -> start, 0 -> stop counter
       ENABLE :=E0.1             // 1 -> to Terminal 3 24V = disable, 0 -> no
effect
       INT2   :=E0.2             // 1 -> start OB40 at reference value 2
       INT1   :=E0.3             // 1 -> start OB40 at reference value 1
       VALUE2 :=W#16#A           // Reference value 2 (must be > VALUE1)
       VALUE1 :=W#16#5           // Reference value 1
       RET_VAL:=MW240            // Error code
NOKO: NOP   0                    // Jump label
```

After initialization with SFC251, the counter stands at VALUE2. After the first counter pulse, it counts to 1.

In parallel with the counting, the counter can be read within the application program using SFC252 (READCNTR).

SFC252 parameters:

Parameter	Declaration	Type	Range	Description
RET_VAL	Output	WORD	YYYY[1]	Current counter status

[1] 2-byte ranges from 0x0000 to 0xFFFF.

Programming example:
Calling SFC252

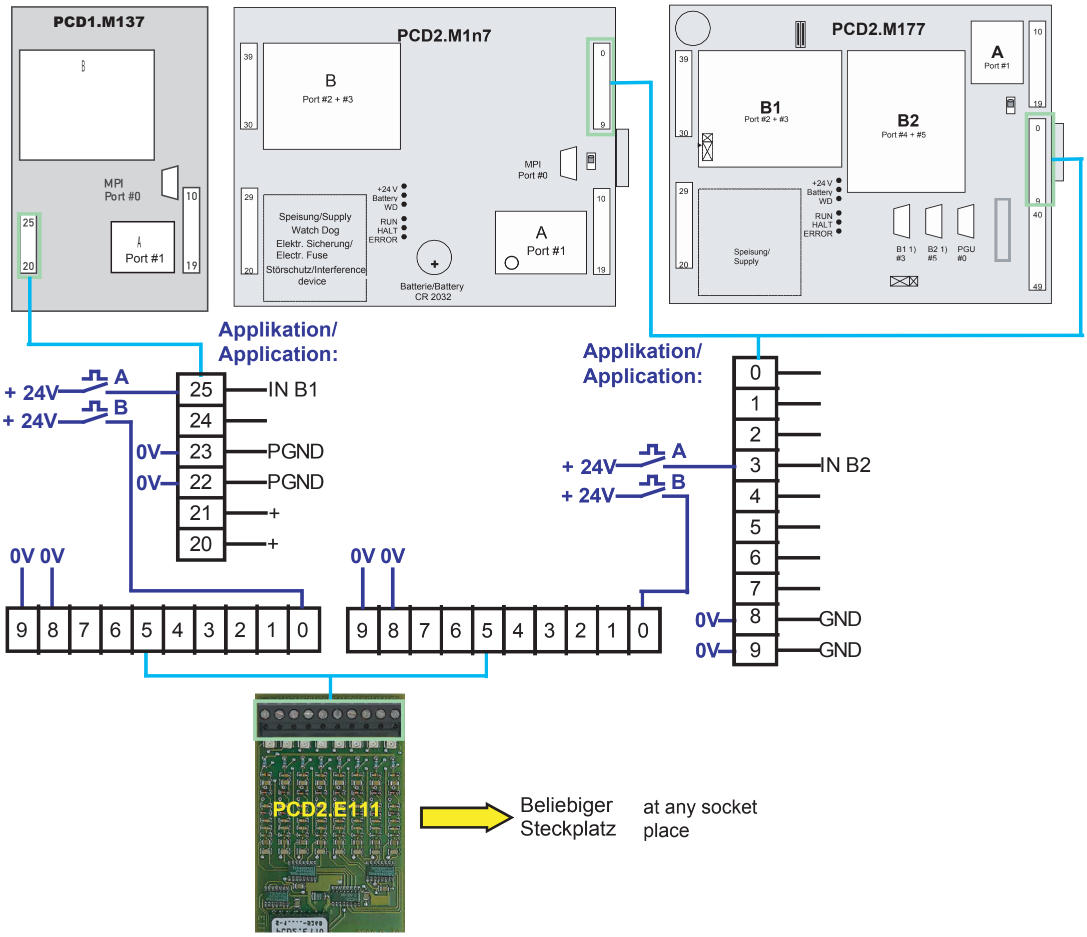
```
Network 1: Read counter status

      CALL  "READCNTR"               // Call SFC252
       RET_VAL:=MW252                // Read counter status
```

3.3.2 Bidirectional counter (SFC248 INTDIR)

The integrated interrupt input 1 and Bit 0 of a digital input module can be used to implement a bidirectional counter. The change of direction is detected by Bit 0 of the digital input module (e.g. PCD2.E110, PCD2.E111). The counting frequency is dependent on the input filter in the input module. If the PCD2.E111 is used, the input filter is set to 0.2 ms, i.e. the maximum counting frequency is approx. 5 kHz. Counting can be continuous or up to a maximum value. Overflow handling can be set. When the maximum value is reached, an interrupt can be triggered. After triggering the interrupt, process alarm OB41 is called.

The bidirectional counter (SFC248) functionality is **not** available on the PCD2.M487 and the PCD3.Mxxx7. Function SFB256 BOARDCNT can be used as an alternative.

Switching example and connection layout



PCD1 mapping:

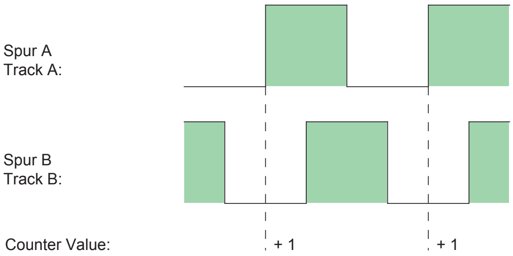
Name	Terminal	Signal	Description
Counter input	25	IN B1	Counter pulse from positive edge
Direction	0	Bit 0	Digital input module

PCD2.M1x7 mapping

Name	Terminal	Signal	Description
Counter input	3	IN B2	Counter pulse from positive edge
Direction	0	Bit 0	Digital input module

Restriction: Only the positive edges are analyzed at the counter input (1-way mode).

An incremental encoder can be connected directly to the controller.
Input A = Counter input on Terminal 3 (25),
Input B = Direction input on Terminal 0 (Bit 0).
The phase shift between Inputs A and B is used to detect the direction:

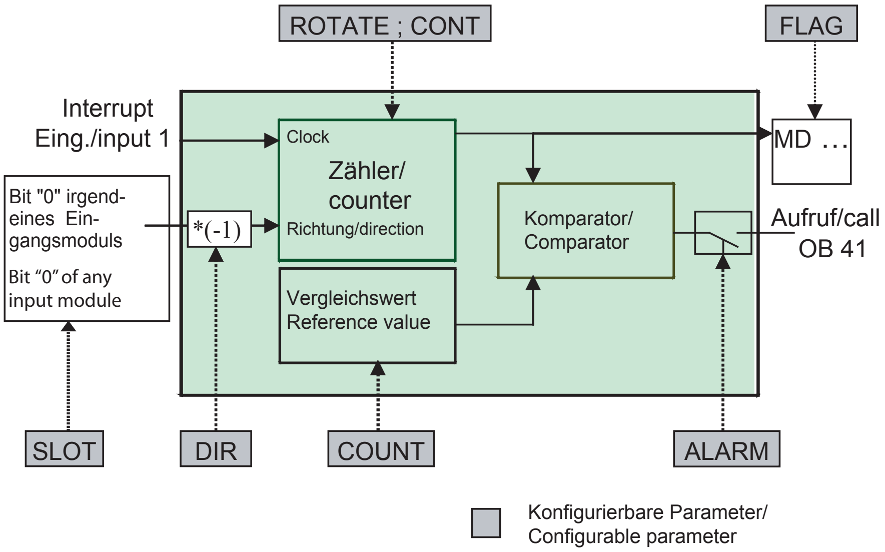


Only the positive edge from Input A is interpreted as a counter pulse.

Operation

SFC248 (INTCNTR) is used to configure and start or stop the bidirectional counter. A single SFC248 call in the application program is sufficient to configure / start the counter.

If the bidirectional counter is started and configured, its basic operation is as follows:



SFC248 parameters

Parameter	Declaration	Type	Range	Description
START	Input	BOOL	TRUE/ FALSE	TRUE: Start counter FALSE: Stop counter
CONT	Input	BOOL	TRUE/ FALSE	TRUE: Count continuously FALSE: Count until maximum value reached
ALARM	Input	BOOL	TRUE/ FALSE	TRUE: OB41 called when maximum value reached. FALSE: OB41 not called

ROTATE	Input	BOOL	TRUE/ FALSE	TRUE: Overflow handling (see below) FALSE: No overflow handling
DIR	Input	BOOL	TRUE/ FALSE	TRUE: Direction signal inverted FALSE: Direction signal unchanged
SLOT	Input	INT	1...8	Module slot of input module for the direction bit.
COUNT	Input	DWORD	XXXX[1]	Maximum value and reference value for the counter.
FLAG	Input	INT	YYYY[2]	The first byte of the double-word containing the counter value.
RET_VAL	Output	INT	YYYY[2]	Error message: 0: No error -1: Incorrect counter value (=0?) -2: Incorrect FLAG address -3: Incorrect SLOT address

[1] 4-byte ranges from 0x0000`0000 to 0xFFFF`FFFF.

[2] Integer ranges from -32768 to +32767.

ROTATE: If this value is set to 1, counting continues; i.e. for forward counting, up to the maximum value (COUNT parameter) then starting again at 0, and for backward counting, down to 0 then starting again at the maximum value (COUNT parameter).

If this value is set to 0, counting is between 0 and 0xFFFF FFFF.

OB41 parameters

When OB41 is called, two pieces of information are provided in the local data byte:

#OB41_RESERVED_1[BYTE]:

Value=1 Reference value reached when counting upwards,
Value=0 Significant when counting downwards.

#OB41_POINT_ADDR[DWORD]:

Contains the reference (maximum) value of the counter at the time of calling OB41.

Programming examples:
Calling SFC248

```
Network 1: Start / configure bidirectional counter

      U     E      0.7             // Configuration handover
      FP    M      0.7
      SPBN  CNT0                   // Impulse - configure counter
      CALL  “INT_DIR“
       START  :=E0.2               // 1->start, 0->stop
       CONT   :=E0.3               // 1->continuous, 0->stop at max
       ALARM  :=E0.4               // 1->call OB41, 0->no call
       ROTATE :=E0.5               // 1->overflow handling, 0->none
       DIR    :=E0.6               // 1->reverse, 0->no reverse
       SLOT   :=1                  // Module slot for dig. input module
       COUNT  :=DW#16#A            // Reference value (maximum value = 10)
       FLAG   :=10                 // ->MD10 contains counter value
       RET_VAL:=MW248              // Return value, error
CNT0: NOP   0
```

Analysis of the interrupt in OB41

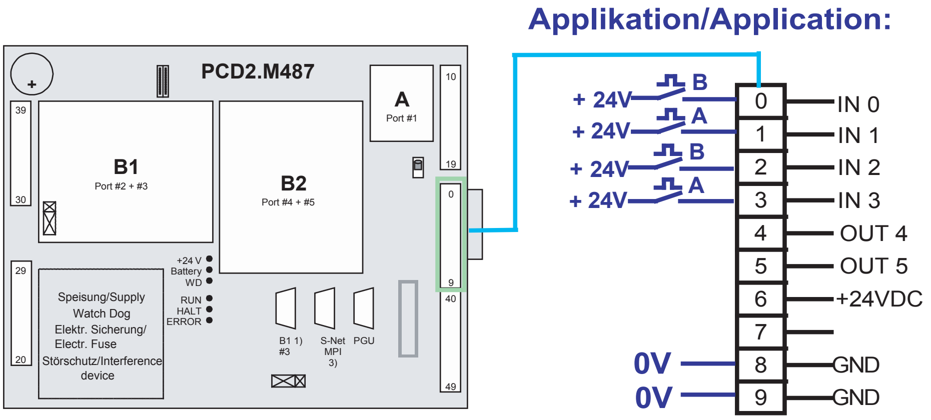
```
OB41:  “Hardware Interrupt”
Network 1: Determine direction of counting

      L     0                            // Counting downwards?
      L     #OB41_RESERVED_1             // Direction of counting
      ==I
      SPB   Down
      L     1                            // Counting upwards?
      ==I
      SPB   UP
      BEA

Network 2: Count downwards

Down: L     #OB41_POINT_ADDR             // Read reference value
// ... other program

      BEA

Network 3: Count upwards

UP:   L     #OB41_POINT_ADDR             // Read reference value
// ... other program

      BEA
```

3.3.3 Onboard counter on PCD2.M487/PCD3.Mxxx7 (SFB256 BOARDCNT)

The four onboard interrupt inputs (IN0 to IN3) can be used to implement two independent bidirectional counters. The pulses (positive edges) are analyzed at the relevant counter input. The signal at the second input establishes the direction of counting. The maximum counting frequency is approx. 1 kHz. Overflow handling may be set. When the reference value is reached, an interrupt can be triggered and an onboard output (Out4,Out5) set. After triggering the interrupt, the planned alarm OB (40..47) is called.

The SFB256 (onboard counter) functionality is **not** supported on the PCD1 and the PCD2.M1x7. On these systems, the functions SFC 251 (INTCNTR) or SFC 248 INT-DIR can be used as alternatives.

Switching example and connection layout - PCD2.M487



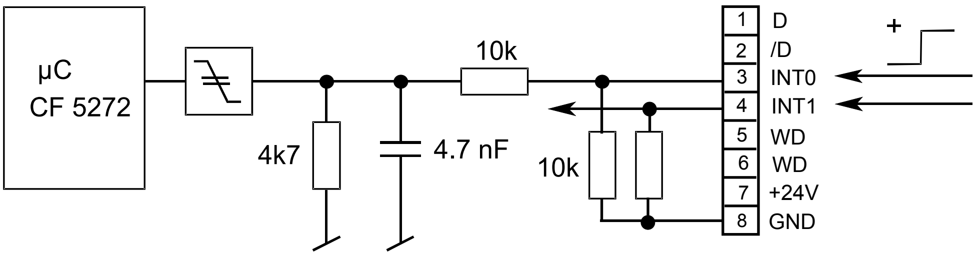
For technical reasons, it is necessary to connect the direction signal (Signal B) to Input 0(2) and the counter signal (Signal A) to Input 1(3).

PCD2.M487 mapping

Name	Terminal	Signal	Description
Counter input 1	1	IN 1	Counter pulse from positive edge - counter 1
Direction 1	0	IN 0	Direction input - counter 1
Counter input 2	3	IN 3	Counter pulse from positive edge - counter 2
Direction 2	2	IN 2	Direction input - counter 2
Max. value reached	4	OUT 4	Counter 1 has reached the REF value.
Max. value reached	5	OUT 5	Counter 2 has reached the REF value.

Restriction: Only the positive edges are analyzed at the counter input (1-way mode).

Switching example - PCD3.Mxxx7

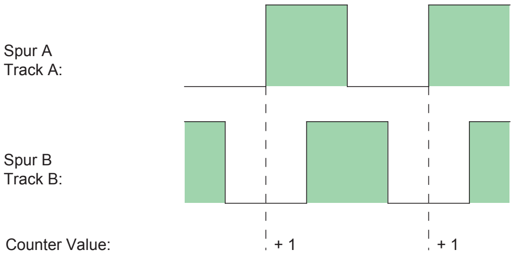


PCD3.Mxxx7 mapping

Name	Terminal	Signal	Description
Interrupt input 0	3	INT0	Positive edge
Interrupt input 1	4	INT1	Positive edge

The two interrupt inputs can be configured and enabled / disabled independently.

An incremental encoder can be connected directly to the controller.
Input A = Counter input on Terminal 1 (3),
Input B = Direction input on Terminal 0 (2).
The phase shift between Inputs A and B is used to detect the direction:



Only the positive edge from Input A is interpreted as a counter pulse.

Operation

SFB256 (BOARDCNT) is used to configure and start or stop the two bidirectional counters on the M487 / the bidirectional counter on the PCD3.Mxxx7. A single SFC256 call in the application program is sufficient to configure / start the counter. The current counter status can be read at any time via the following peripheral addresses:

- Counter 1: PED 65000
- Counter 2: PED 65004 (PCD2.M487 only)

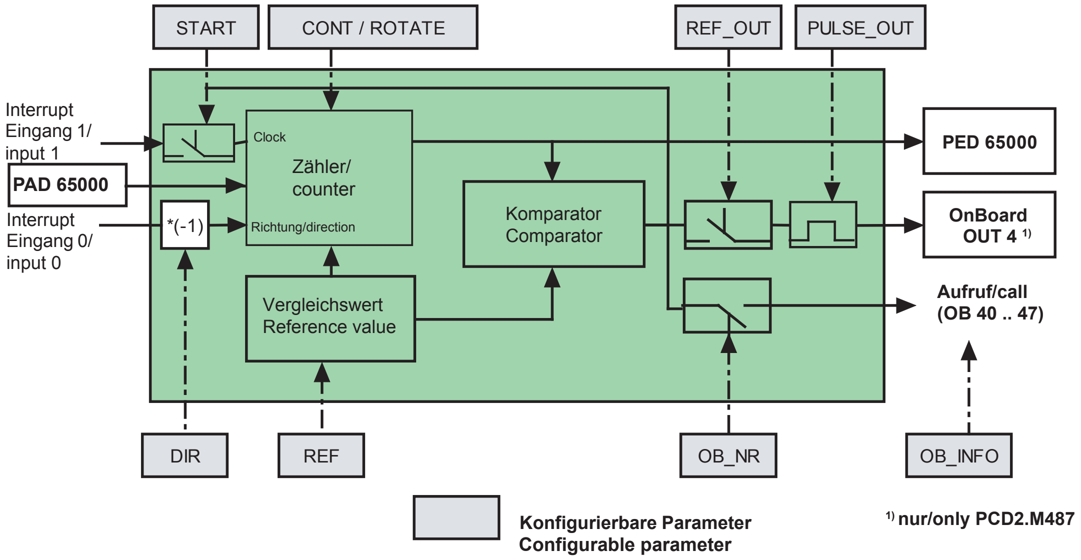
The counter value can be read at any time via the following peripheral addresses:

- Counter 1: PAD 65000
- Counter 2: PAD 65004 (PCD2.M487 only)

When the Saia PCD® is started up, both counters are set to 0. If the Saia PCD® enters a Stop state, the counters are stopped.

The parameters can only be changed when the counter is stopped. Exception: The counter can be set to a new value at any time.

The figure below shows the basic operation of the onboard counter on the PCD2. M487 / PCD3.Mxxx7, using Counter 1 as an example:



[1] nur/only PCD2.M487

SFC248 parameters

Parameter	Declaration	Type	Range	Description
COUNT_NUM	Input	INT	1...2	1: Counter 1 (IN0 , IN1, OUT4) 2: Counter 2 (PCD2.M487 only) (IN2, IN3, OUT5)
START	Input	BOOL	TRUE/ FALSE	TRUE: Start counter FALSE: Stop counter
CONT	Input	BOOL	TRUE/ FALSE	TRUE: Count continuously FALSE: Count until reference value reached, then stop counter.
ROTATE	Input	BOOL	TRUE/ FALSE	TRUE: Overflow handling (see end of table) FALSE: No overflow handling
DIR	Input	BOOL	TRUE/ FALSE	TRUE: Direction signal inverted FALSE: Direction signal unchanged.
REF_OUT (PCD2.M487 only)	Input	BOOL	TRUE/ FALSE	TRUE: If the counter reaches the reference value, the onboard outputs are set. Counter 1: OUT4 Counter 2: OUT 5 FALSE: Reaching the reference value has no effect on the onboard output.
PULSE_OUT	Input	BOOL	TRUE/ FALSE	TRUE: The onboard output is reset to 0 on the next counter pulse. FALSE: The onboard output remains set until it is reset by the user. (See end of table)
REF	Input	DWORD	XXXX[1)]	Maximum value and reference value for the counter.
OB_NR	Input	INT	0 40...47	0: When the reference value is reached, no OB is called. 40..47: When the reference value is reached, the OB set here as a parameter is called.
OB_INFO	Input	WORD	YYYY[2)]	The value is copied into the local data word OB_4x_MDL_ADDR when the counter OB is called.

RET_VAL	Output	INT	ZZZZ[3]	Error and status message: 0: Counter started. 1: Counter already running. (called when START=TRUE) 2: Counter stopped. -2: Parameter COUNT_NUM is invalid (1..2). (PCD3.Mxxx7 only [1]) -3: Parameter OB_NR is invalid (40..47 or 0) -4: The counter inputs are already configured as interrupt inputs with SFC256. -5: Counter initialised with COUNT = TRUE and ROTATE = TRUE, and REF=0.
RESERVED	Output	BOOL	TRUE/ FALSE	Reserved

[1] 4-byte ranges from 0x0000 0000 to 0xFFFF FFFF.

[2] 2-byte ranges from 0x0000 to 0xFFFF.

[3] Integer ranges from -32768 to +32767.

If SFB256 completed without any error, the BIE bit is reset. In case of error, the BIE bit is set and RET_VAL contains the error information.

ROTATE: If this value is set to 1, counting continues; i.e. for forward counting, up to the reference value(REF parameter) then starting again at 0, and for backward counting, down to 0 then starting again at the reference value (REF parameter).

If this value is set to 0, counting is between 0 and 0xFFFF`FFFF.

Resetting the onboard output: The command “T PAB 65533” is used to write Bit 4 from Akku 1 to the onboard output OUT4. This enables the output for Counter 1 to be reset (PCD2.M487 only).

The command “T PAB 65534” is used to write Bit 5 from Akku 1 to the onboard output OUT5. This enables the output for Counter 2 to be reset (PCD2.M487 only).

The command “T PAB 65535” can be used to write Bits 4 and 5 from Akku 1 to the onboard outputs OUT4 and OUT5. This enables the output for Counters 1 and 2 to be reset at the same time (PCD2.M487 only).

Programming examples:

Calling SFB256

```
Network 1: Configure and start counter 1

      U     E      0.7                // Configuration handover and start
      FP    M      0.7
      SPBN  CNT0                      // Pulse - configure counter
      CALL  “BOARDCNT“ , “BOARDCNT1“  // Call SFB256 with InstanzDB
       IN0    :=1                     // COUNT_NUM: Configure counter 1
       IN1    :=E0.1                  // START: 1->start, 0->stop
       IN2    :=E0.2                  // CONT: 1->continuous, 0->stop at REF
       IN3    :=E0.3                  // ROTATE: 1->overflow handling, 0->none
       IN4    :=E0.4                  // DIR: 1->reverse, 0->no reverse
       IN5    :=E0.5                  // REF_OUT: 1: Set OUT4 on REF, not set
       IN6    :=E0.6                  // PULSE_OUT: 1:OUT4 remains at 1 until next counter pulse
       IN7    :=DW#16#A               // REF: Reference value (maximum value=10)
       IN8    :=41                    // OB_NR: If REF reached, call OB41
       IN9    :=W#16#C001             // OB_INFO: Start information in OB41
       RET_VAL:=MW240                 // RET_VAL: possible error message or status information
       OUT10  :=M250.0                // RESERVED: Reserved.
CNT0: NOP   0
```

Analysis of the interrupt in OB41

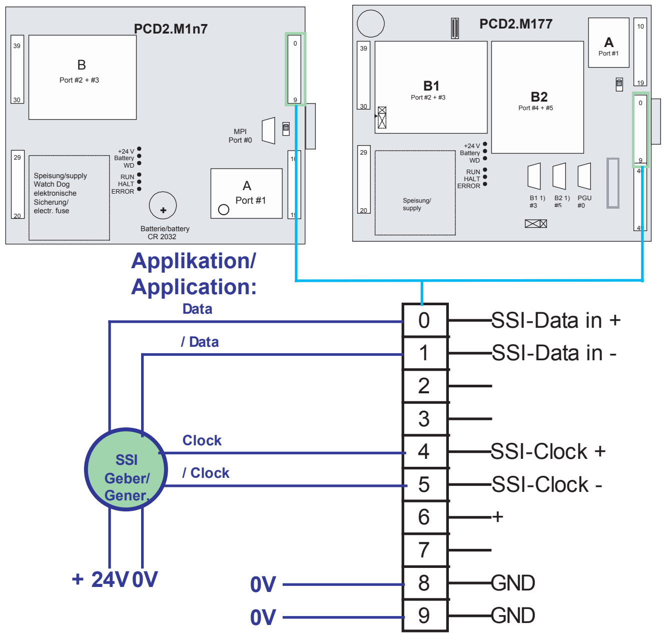
```
OB41: “Counter 1 interrupt”
Network 1:

      L      #OB41_MDL_ADDR          // Start information
      L      W#16#C001               // Planned start information
      ==I
      SPB    Int0                    // Jump to interrupt 0
      BEA

Network 2: Counter 1 has reached REF

Int0: NOP   0
// Execute interrupt function

// ...
```

3.4 Integrated SSI interface (SFC253 READ_SSI, SFC254 GRAY2BIN)

The integrated SSI interface allows absolute value encoders to be read. Bit format and value encoding can be customized individually.

The SSI interface functionality is **not** available on the PCD1, the PCD2.M487 and the PCD3.Mxxx7.

Switching example and connection layout:



PCD2.M1x7 mapping

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

Operation:

The SSI interface is always active. As soon as SFC253 (READ_SSI) is called within the application program, the current stored position value can be read. If the value is held in Gray code, it can easily be converted into a binary format that can be used by the application program.

SFC253 (READ_SSI) parameters:

Parameter	Declaration	Type	Range	Description
BIT_CNT	Input	BYTE	1...32	Number of bits to be read in.
RET_VAL	Output	DWORD	YYYY[1)]	Value read.

SFC254 (GRAY2BIN) parameters:

Parameter	Declaration	Type	Range	Description
GRAY	Input	DWORD	YYYY[1)]	Value in Gray code
RET_VAL	Output	DWORD	YYYY[1)]	Binary value

[1)] 4-byte ranges from 0x0000 0000 to 0xFFFF FFFF.

Programming examples:

```
Network 1: Read SSI interface and convert value

// Read interface
      CALL  “READ_SSI“                // Call SFC253
       BIT_CNT:=B#16#24               // 24 bits read
       RET_VAL:=MD250                 // Temporary variable
// Convert Gray code to binary code
      CALL  “GRAY2BIN“                // Call SFC254
       GRAY   :=MD250                 // Temporary variable
       RET_VAL:=MD100                 // Current position in binary format
```

4 Communication functions

The Saia PCD® Series xx7 controllers are distinguished by their wide range of communication facilities. From the simple serial data interface using MPI networking through to field bus connections and telecommunications via modem, the Saia PCD® controllers offer a huge variety of communications solutions. In the sections below, only the extended system functions will be described, as an example of the communication functions available. More detailed information can be found in the function-specific manuals.

4.1 MPI protocol

As well as the usual MPI interface, the PCD1, PCD2 and PCD3 Series xx7 controllers can also be programmed via a serial user interface. This allows the serial interface to be connected directly to the PC for programming, without an MPI adapter. For the PCD1 and PCD2.M1x7, a serial connector cable is required (see manual 26/794 “Serial Communication”), with an RS-232 interface module to connect it to the COM interface on your PC. If Port 0 (the default port) on the PCD2.M487 or the PCD3.Mxxx7 is used, no interface module is required. The connector cable PCD8.K111 PGU is also needed. In the SIMATIC Manager, the interface must be set to “MPI-Adapter”, to run at 19200 baud.

The Configuration Data Block (CDB) can be used to configure any interface to the MPI protocol. If no CDB is present, the MPI protocol is always activated on Port 1 (or Port 0 on the PCD2.M487 or PCD3).

After Power On, the MPI driver is active on the serial interface. You can then go online with your programming software immediately.

On the PCD1, the MPI protocol can be used on Port 1 only with a modem. For Ports 2 and 3, a separate module with a special cable is required.

With the serial interface, you can only access directly connected PCD1 / PCD2.M1x7 devices; other controllers connected via MPI cannot be reached.

The PCD2.M487 and the PCD3.Mxxx7 have gateway functionality; i.e. you can use the serial interface to reach controllers connected to a PCD2.M487 or a PCD3.Mxxx7 via the MPI network

4.1.1 Disabling / enabling the MPI protocol (SFC200 control)

SFC200 allows the programmer to enable and disable the MPI protocol on the serial interface. SFC200 activates/ de-activates the MPI driver on the interface defined at Power On.

On the PCD2.M487 and the PCD3.Mxxx7, SFC200 activates / de-activates the MPI driver on the last-activated interface. To activate / de-activate the MPI driver on any interface, SFC300 can be used.

Calling SFC200 with VKE = 0 de-activates the MPI protocol. Calling SFC200 with VKE = 1 re-activates the MPI protocol.

Programming example:

```
Network 1: Enable / disable MPI protocol

       U      E 0.7                  // Enable / disable
       FP     M 10.0                 // Edge pos. marker
       SPBN   Off
       SET                           // VKE on
       CALL   SFC 200                // MPI on

Off:   U      E 0.7                  // Enable / disable
       FN     M 10.1                 // Edge neg. marker
       SPBN   End
       CLR                           // VKE off
       CALL   SFC 200                // MPI off

End:   NOP    0
```

4.1.2 Extended disabling / enabling of the MPI protocol (SFC300 XControl)

SFC300 allows the programmer to enable and disable the MPI protocol on any serial interface. SFC300 activates / de-activates the MPI driver on the defined interface.

The SFC300 function is **not** available on the PCD1 and the PCD2.M1x7. The SFC200 Control function can be used as an alternative.

SFC300 parameters:

Parameter	Type	Type	Range	Description
PORT	IN 0	INT	0...6	Interface number
DoStart	IN 1	BOOL	TRUE/ FALSE	TRUE: Starts the MPI driver FALSE: Stops the MPI driver
Ret_Val	Output	INT	XXXX[2)]	Error message: 0:Function completed without error 1:Function completed -1:Invalid interface number -2:Internal error -3:Driver error

[1)] PCD3..Mxxx7 Ports nos 0 to 2

[2)] Integer ranges from -32768 to +32767

Before the MPI protocol can be activated on a new interface, it must be de-activated on the old interface.

SFC300 can only be used to activate the MPI protocol if the interface has been previously initialized with a CDB entry or with SFC245 with the following parameters:

Mode:	1 = DK3964R
Baud rate:	19200 or 38400
Data bit:	8
Stop bit:	1
Parity:	2 = Odd
ZVZ:	0 = Default value
QVZ:	0 = Default value
Send buffer:	256
RCV buffer:	256

Programming example:

```
Network 1: Enable / disable MPI protocol

       U      E 0.7                // Enable / disable
       FP     M 10.0               // Edge pos. marker
       SPBN   next
       CALL   SFC 300
         IN0      :=1              // Interface 1
         IN1      :=TRUE           // MPI on
         RET_VAL  :=MW102          // Error message
next:  U      "Switch 7"           // Enable / disable
       FN     M 10.1               // Edge neg. marker
       SPBN   nex2
       CALL   SFC 300
         IN0      :=1              // Interface 1
         IN1      :=FALSE          // MPI off
         RET_VAL  :=MW102          // Error message
nex2:  NOP 0
```

4.1.3 Enable MPI protocol with Configuration Data Block (CDB)

To configure an interface other than the default interface to the MPI protocol, the Configuration Data Block (CDB, Ident = “SBC xx7 CDB”) must be programmed.

The following interfaces are supported:

PCD1: Interface 1..0.3 (default: 1), 19200 baud only
  Interface 1 can only be used with a modem.
  (no F120 module).
PCD2.M127, PCD2.M157: Interface 1..0.3 (default: 1) 19200 or 38400 baud
PCD2.M177: Interface 1...5 (default: 1) 19200 or 38400 baud
PCD2.M487: Interface 0...5 (default: 0) 19200 or 38400 baud
PCD3.Mxxx7: Interface 0..1 (default: 0) 19200 or 38400 baud

On the PCD2.M1x7, the interfaces in slots B1 and B2 support only 19200 baud by default. With a CDB entry, the interfaces can support 38400 baud instead of 19200 baud. This setting applies to all interfaces on the relevant slot.

Example interface 2:

Address	Name	Type	Initial value
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	

Activating an interface on MPI disables the MPI protocol on the default interface. The CDB can also be used for the default interface.

The Configuration Data Block (CDB) is only retrieved after a cold start of the CPU (Power On).

For further information, see the section on System configuration (CDB).

4.2 LON communication

The PCD2.M1x7 Series controllers can be equipped with a LON (**L**ocal **O**perating **N**etwork) connection. This LON connection is programmed using SFC calls. SNVT network variables and explicit messages are held in data blocks. There are 3 SFCs available:

1. To initialise the LON interface, SFC220, LON_INIT
2. To send an SNVT network variable, SFC221, NV_SEND
3. To send a message, SFC223, MSG_SEND

The SFC220, SFC221 and SFC223 components are only present in the CPU operating system when the LON module is plugged in. Alternatively, they can be copied from the SBC library to the offline project.

The LON functions are **not** supported by the PCD1, the PCD2.M487 and the PCD3. Mxxx7.

For more information, please refer to manual 26/767 “LON”. The LON network configuration tool SNET32 for xx7 is free of charge from www.sbc-support.com.

4.2.1 Initialising the LON interface (SFC220, LON_INIT)

SFC220 is used to initialise the LON interface and to define the LON data blocks.

SFC220 parameters:

Parameter	Type	Type	Range	Description
REQ	Input	BOOL	TRUE/ FALSE	TRUE: Initialization incl. reset
DB_NO	Input	WORD	1...1023	DB-No. with LON configuration data
Ret_Val	Output	WORD	0	Error information: see section Error information for the LON SFCs

After initialization, the CPU calls OB86. If OB86 has not been programmed, the CPU goes into a Stop state.

OB86 can be used to determine whether an error occurred during initialization, or initialization was executed correctly.

Programming example:
LON initialization in OB100:

```
Network 1: Initialize LON interface

       SET
       R      M 100.0              // LON not initialized
       L      512                  // Config-DB number = 512
       T      MW 200
       CALL  “LON_INIT”
         REQ      :=TRUE           // REQ: Start initialization
         DB_NO    :=MW200          // DB_NO: Config-DB = 512
         RET_VAL  :=MW204
```

Interpretation in OB86:

```
Network 1: Test whether LON initialization OK.

       L      #OB86_EV_CLASS
       L      B#16#39              // Error
       <>I
       SPB    ok                   // no -> LON initialiation OK
       R      M 100.0
       L      512
       T      MW 200
       CALL   “LON_INIT”           // Re-initialization of LON
         REQ      :=TRUE           // REQ: Start initialization
         DB_NO    :=MW200          // DB_NO: Config-DB = 512
         RET_VAL  :=MW204
       BEA
ok:    S      M 100.0              // LON initialization OK
```

The possible error messages for OB86 are described in manual 26/767 “LON”.

4.2.2 Sending an SNVT network variable (SFC221, NV_SEND)

SFC221 sends a network variable (NV). This is sent to a specified node.

SFC221 parameters:

Parameter	Type	Type	Range	Description
REQ	Input	BOOL	TRUE/ FALSE	TRUE: SEND
VAR_NAME	Input	ANY		Symbolic name of network variables
Ret_Val	Output	WORD	0	Error information: see section on "Error information for the LON SFCs"
BUSY	Output	BOOL	TRUE/ FALSE	TRUE: Send request running
DONE	Output	BOOL	TRUE/ FALSE	TRUE: Send request completed without error
ERROR	Output	BOOL	TRUE/ FALSE	TRUE: Send request completed with error

Programming example:

```
Network 1: Send LON network variable

       CALL  "NV_SEND"
         REQ      :=M100.1
         VAR_NAME :=DB120.DBD1      // Symbolic name of NV
         RET_VAL  :=MW222
         BUSY     :=M200.0
         DONE     :=M200.1
         ERROR    :=M200.2
```

4.2.3 Sending a message (SFC223, MSG_SEND)

SFC223 sends an explicit message. This explicit message is sent to the specified node.

SFC223 parameters:

Parameter	Type	Type	Range	Description
REQ	Input	BOOL	TRUE/ FALSE	TRUE: SEND
MSG	Input	ANY		Message name
LEN	Input	INT	1... 32’767	Length of message
Ret_Val	Output	WORD	0	Error information: see section “Error information for LON SFCs”
BUSY	Output	BOOL	TRUE/ FALSE	TRUE: Send request running
DONE	Output	BOOL	TRUE/ FALSE	TRUE: Send request completed without error
ERROR	Output	BOOL	TRUE/ FALSE	TRUE: Send request completed with error

Programming example:

```
Network 1: Send LON message

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

4.2.4 Error information for LON SFCs

SFCs 220, 221 and 223 reset the binary result bit (Bit 8) in the CPU status register when they have completed without error. In case of error, the binary result (BIE) bit is set and the RET_VAL variable contains further error information. The error codes mainly follow the S7 standard. There are some additional error codes specific to xx7 LON.

BUSY	DONE	ERROR	BIE	RET_VAL	Description
0	0	0	0	0x7000	Initial call with REQ = 0: no data transfer active
1	0	0	0	0x7001	Initial call with REQ = 1: data transfer started
1	0	0	0	0x7002	Intermediate call (REQ irrelevant): Data transfer already active
0	1	0	0	0x0000	Data transfer completed successfully
0	0	1	0	0x0001	LON interface malfunction
0	0	1	0	0x0002	The job cannot be accepted within the time.
0	0	1	0	0x4000	Data transfer not successful
0	0	1	1	0xFFFF	The job could not be accepted because too little (internal) memory available. Try again later.
0	0	1	1	0xFFFE	The specified network variable was not found.
0	0	1	1	0xFFFD	Error in LON configuration data block
0	0	1	1	0xFFFC	LON driver not yet initialized
0	0	1	1	0xFFFB	The data block number does not match initialization
0	0	1	1	0xFFFA	Message too long
0	0	1	1	0x803A	Specified data block (e.g. SNVT DB) not found
0	0	1	1	0x8022	Data block does not contain the SNVT or message (DB length error)

4.3 Serial communication

The PCD1/PCD2.M1x7 Series controllers support 2 options for serial communication.

1. SBC mode
2. CP 441

The PCD2.M487 and PCD3.Mxxx7 Series controllers support only 1 option for serial communication.

1. CP 441

Configuration
SFC245 can be used to configure the COM port.

SBC mode
The mode that has been available since the launch of the PCD1/PCD2.M1x7 Series includes 4 SFCs (240…243). These can be used to access the send and receive buffers transparently. The options are completely open, although restricted by cycle-time and buffer-size.

CP 441
This mode allows communication with the SIMATIC® environment. It supports the most widely used protocols, such as ASCII, DK3964R and RK512.

SFBs 12, 13 and 14, which are the same SFBs provided with SIMATIC®, are used to manage communications.

There are a further 2 protocols available that are not compatible with SIMATIC®, as they contain more functionality.

1. RK512MP: like RK512, but supports a master-slave network (multi-point MP). The PCD2.M487 and the PCD3.Mxxx7 do not support the RK512MP protocol.
2. Transparent: like SBC mode, but can process more data.

Modem signals
There is also SFC244, for direct access to modem signals. For controls in the PCD2.M487 and PCD3.Mxxx7 series, SFC344 provides extended access to the modem signals.

i In the sections below on serial communication, only the extended system functions in SBC mode and the initialization component SFC245 for CP441 mode are described, by way of example. More detailed information can be found in manual 26/794 “Serial Communication”.

! SFCs 240 to 243, and the RK512MP protocol are **not** supported on the PCD2.M487 and the PCD3.Mxxx7.

4.3.1 Reading data from the serial interface (SFC240 COM_RCV)

When a serial interface has been initialized, the operating system begins to receive characters over the serial interface independently, and stores them in the receive buffer. Calling SFC240 “COM_RCV” causes the specified number of bytes to be transferred from the receive buffer to a data area selected by an ANY pointer. The ANY pointer specifies not only the start address but also the length of the data area. The data area may be from one byte to 128 bytes (size of the receive buffer) in size.

SFC240 is **not** supported on the PCD2.M487 and the PCD3.Mxxx7.

SFC240 parameters:

Parameter	Type	Type	Range	Description
COM_NR	Input	BYTE	1...5 / 1...3	Indicates which interface should be initialized. Permitted values: PCD2.M177 Port 1...5 PCD1 and PCD2.M127/157/257 Port 1...3
BUFFER	Input	ANY		Specifies the start address and length of the data area into which the bytes received should be transferred. (Pointer to data area)
Ret_Val	Output	INT	-1...1	Error message: 0: No error 1: Not enough bytes in receive-buffer -1: Invalid interface number

Programming example:

```
Network 1: Read data from interface

       CALL   “COM_RCV“
         COM_NR    :=B#16#1                  // Interface No. 1
         BUFFER    :=P#M 1000.0 BYTE 20      // Target range for received
                                                data
         RET_VAL   :=MW240                   // Error message
```

4.3.2 Sending data to the serial interface (SFC241 COM_SEND)

After calling SFC241 “COM_SEND”, a data area selected via an ANY pointer is transferred to the send buffer. The ANY pointer specifies not only the start address but also the length of the data area. The data area may be from one byte to 128 bytes (size of the send buffer) in size. The actual transmission is handled by the operating system in the backgroiund.

SFC241 is **not** supported on the PCD2.M487 and the PCD3.Mxxx7.

SFC241 parameters:

Parameter	Type	Type	Range	Description
COM_NR	Input	BYTE	1...5 / 1...3	Indicates which interface should be initialized. Permitted values: PCD2.M177 Port 1...5 PCD1 and PCD2.M127/157/257 Port 1...3
BUFFER	Input	ANY		Specifies the start address and length of the data area (bytes) to be sent. (Pointer to data area)
Ret_Val	Output	INT	-1...1	Error message: 0: No error 1: Not enough bytes in send buffer -1: Invalid interface number

Programming example:

```
Network 1: Send data to the interface

       U      E 0.0                         // Send trigger
       FP     M 0.0
       SPBN   NSND                          // Pulse
       CALL   “COM_SEND‟
         COM_NR    :=B#16#1                 // Interface No. 1
         BUFFER    :=P#M 2000.0 BYTE 20     // Source range for send data
         RET_VAL   :=MW240                  // Error message
NSND:  NOP     0
```

4.3.3 Reading the status of the serial interface (SFC242 COM_STAT)

After calling SFC242 “COM_STAT”, the status of the specified interface is returned. SFC242 supplies the following information:

- The number of characters received into the receive buffer.
- The number of characters in the send buffer, not yet sent.
- Receive buffer overflow, i.e. more characters have been received than fit in the receive buffer. A receive buffer overflow occurs when data cannot be retrieved from the receive buffer with SFC240 “COM_RCV” quickly enough.
- Interface error, i.e. incorrect characters were received (e.g. wrong baud rate, wrong parity, EMC faults etc.).

SFC242 is **not** supported on the PCD2.M487 and the PCD3.Mxxx7.

SFC242 parameters:

Parameter	Type	Type	Range	Description
COM_NR	Input	BYTE	1...5 / 1...3	Indicates which interface should be initialized. Permitted values: PCD2.M177 Port 1...5 PCD1 and PCD2.M127/157/257 Port 1...3
Ret_Val	Output	INT	-1...0	Error message: 0: No error -1: Invalid interface number
RCV_CNT	Output	INT	0...128	Number of bytes in receive buffer
SND_CNT	Output	INT	0...128	Number of bytes in send buffer
STATUSL	Output	INT	0...1	Error message: Bit 0 = 1 Receive buffer overflow Bit 1 = 1 Interface error

Programming example:

```
Network 1: Read status of interface

       CALL   "COM_STAT"
         COM_NR    :=B#16#1          // Interface No. 1
         RET_VAL   :=MW240           // Error message
         RCV_CNT   :=MW242           // Number of bytes in receive buffer
         SND_CNT   :=MW244           // Number of bytes in send buffer
         STATUS    :=MW246           // Overflow / interface error
```

4.3.4 Initialize serial interface (SFC 243 COM_INIT)

SFC243 can be used to initialize interfaces 1 to 5 (not MPI). The SFC is generally called just once before the start of serial communication, e.g. in OB100.

SFC243 is **not** supported on the PCD2.M487 and the PCD3.Mxxx7.

SFC243 parameters:

Parameter	Type	Type	Range	Description
COM_NR	IN 0	BYTE	1...5 / 1...3	Indicates which interface should be initialized. Permitted values: PCD2.M177 Port 1...5 PCD1 and PCD2.M127/157/257 Port 1...3
SELECT	IN 1	BYTE	0...3	Interface mode: 0: RS-232 1: RS-485 2: RS-422 3: TTY 20mA
BAUDRATE	IN 2	DINT	300... 38’400	Baud rate: permitted values 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 (COM1 only)
COM_PAR	IN 3	WORD		Interface parameters: Bit 1...0: Number of data bits: 00=5, 01=6, 10=7, 11=8 Bit 4...2: Parity: 000=even, 001=odd, 010=forced low, 011=forced high, 10x=none Bit 5: Stop bits: 0=1 stop bit, 1=2 stop bits
Ret_Val	Ret_Val	INT	-1...0	Error message: 0: No error -1:Invalid interface number

Programming example:

```
Network 1: Initialize serial interface

       CALL   “COM_INIT“
         COM_NR   :=B#16#1      // Interface no. 1
         SELECT   :=B#16#0      // RS-232
         BAUDRATE :=L#9600      // 9600 baud
         COM_PAR  :=W#16#3      // 8 data bits, even parity, 1 stop bit
         RET_VAL  :=MW240       // Error ?
```

4.3.5 Access to modem signals (SFC244 COM_SIG)

A call to SFC244 allows the

- **DTR** (Data Transmit Receive) signal to be changed, and the status of the signals
- **DCD** (Data Carrier Detect) and
- **DSR** (Data Set Ready)

from the COM1 interface to be queried.

This component is for Interface 1 only. The DTR signal is inverted. By default, it is in signal state 1. To activate it, SFC244 is called with a 0 at the DTR input. To deactivate it, SFC244 is called with a 1 at the DTR input.

SFC244 parameters:

Parameter	Type	Type	Range	Description
DTR	IN 0	BOOL	TRUE/ FALSE	Control Data Transmit Receive signal
DCD	OUT 1	BOOL	TRUE/ FALSE	Query Data Carrier Detect signal
DSR	OUT 2	BOOL	TRUE/ FALSE	Query Data Set Ready signal

Programming example:

```
Network 1: Access to modem signals

       CALL   "COM_SIG"
         DTR:=M1.0                     // Control DTR
         DCD:=M1.1                     // Status of DCD signal
         DSR:=M1.2                     // Status of DSR signal
```

4.3.6 Extended access to modem signals (SFC344 XCOM_SIG)

Calling SFC344 allows the

- **DTR** to be changed, and the status of the signals
- **DCD** and
- **DSR**

from a COM interface on the **PCD2.M487 and PCD3.Mxxx7** to be queried.

The DTR signal is inverted. By default, it is in signal state 1. To activate it, SFC344 is called with a 0 at the DTR input. To de-activate it, SFC344 is called with a 1 at the DTR input.

This function is not supported on the PCD1 and the PCD2.M1x7. As an alternative, the SFC244 COM_SIG function can be used.

SFC344 parameters:

Parameter	Type	Type	Range	Description
Port	IN 0	INT	Valid port no.	Interface number
DTR	IN 1	BOOL	TRUE/ FALSE	Control Data Transmit Receive signal

DCD	OUT 2	BOOL	TRUE/ FALSE	Query Data Carrier Detect signal
DSR	OUT 3	BOOL	TRUE/ FALSE	Query Data Set Ready signal
RetVal	RET_ VAL	INT	-2...0	Error code: 0: No error -1: Invalid interface number -2: Selected interface does not support modem-signals

Programming example:

```
Network 1: Extended access to modem signals

      CALL  SFC 344
       IN0     :=2                          // Interface 2
       IN1     :=M1.0                       // Control DTR
       OUT2    :=M1.1                       // Status of DCD signal
       OUT3    :=M1.2                       // Status of DSR signal
       RET_VAL:=MW240                       // Error message
```

4.3.7 Initialize serial interface (CP441) (SFC 245 B_INIT)

SFC245 can be used to initialise interfaces 0 to 6 (CP441). Initialisation is associated with communication components SFB12, SFB13 and SFB14, which can also be found in the standard library for Step7® software. Further associations can be found in manual 26/794 “Serial Communication”. The SFC is generally called just once before the start of serial communication, e.g. in OB100.

It is possible to switch the interface driver or the protocol during execution, but note the following:

- The buffer size cannot be changed
- The send and receive components require a positive edge at the enable input to recognise the new mode.

The table below refers to transparent mode as an example. More detailed descriptions of the parameters and values can be found in manual 26/794 “Serial Communication”.

SFC245 parameters:

Parameter	Type	Type	Range	Description
COM_NR	IN 0	INT	Valid port no.	Indicates which interface should be initialized. Permitted values: PCD2.M487 Port 0 to 6 PCD2.M177 Port 1 to 5 PCD1 and PCD2.M127/157/257 Port 1 to 3 PCD3.Mxxx7 Port 0 to 2.
MODE	IN 1	INT	0...8	0=Transparent mode 1=DK3964 2=DK3964R 3=RK512 - DK3964 4=RK512 - DK3964R (Multipoint not supported on the PCD2.M487 and the PCD3.Mxxx7) 5=ASCII - fixed length 6=ASCII - 1 end character 7=ASCII - 2 end characters 8=ASCII - ZVZ (0..8 see manual 26/794 “Serial Communication”)
BAUDRATE	IN 2	DINT	Valid baud rate	Permitted baud rates: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 (COM1 only) **PCD2.M487**: baud rates <1200 baud not supported. Ports 0 and 6 support 115000 Baud. **PCD3.Mxxx7**: baud rates <1200 baud not supported. Ports 0 to 2 support 115000 Baud.
DATA_BIT	IN 3	INT	7…8	Number of data bits:
STOP_BIT	IN 4	INT	1…2	Number of stop bits
PARITY	IN 5	INT	0…4	0=None, 1=Even, 2=Odd, 3=Force low, 4=Force high 5=Multi-point (for Mode=4 only). Not supported on the PCD2.M487 and PCD3.Mxxx7
CONTROL	IN 6	INT	0…3	Interface type: 0=RS-232, 1=RS-485, 2=RS-422, 3=TTY
XON	IN 7	BYTE	0…FFh	XON character (default 0x11)
XOFF	IN 8	BYTE	0…FFh	XOFF character (default 0x13)
WAIT_SEND	IN 9	WORD	0…FFFEh	Time from activating the control signal to sending (10ms steps)
WAIT_INACTIVITY	IN 10	WORD	0…FFFEh	Time from end of transmission to removal of RTS (10ms steps)
TEL_COUNT	IN 11	INT	1	Number of telegrams in buffer
OVER_WRITE	IN 12	BOOL	FALSE/TRUE	FALSE: Telegram cannot be overwritten TRUE: Telegram can be overwritten, but only if TEL_COUNT = 1
DEL_RX_BUFFER	IN 13	BOOL	FALSE/TRUE	Receive buffer deleted during start-up

DK_PRIORITY	IN 14	BOOL	FALSE/ TRUE	DK3964R/RK512 only: priority of DK driver (TRUE: high priority)
ZVZ	IN 15	WORD	0… FFFEh	Character delay time after which a telegram end is recognized: The ZVZ should be at least 4 times the send time for one character. 0=Default 220 ms For mode 1 to 4: Character delay time in 10 ms steps For mode 5 to 8: Character delay time in 1 ms steps
QVZ	IN 16	WORD	0… FFFEh	Acknowledgement delay time in 10 ms steps 0 = Default For mode 1 and 3 default 550 ms For mode 2 and 4 default 2000 ms
TRY_CONNECT	IN 17	INT	0…255	Number of attempts to establish a connection 0 = 6 attempts (default)
TRY_SEND	IN 18	INT	0…255	Number of send attempts in the event of error 0 = 6 attempts (default)
FIXED_LENGTH	IN 19	INT	1…1024	Telegram length
END_CHAR1	IN 20	BYTE	0…255	First end character of a telegram (MODE 6 only)
END_CHAR2	IN 21	BYTE	0…255	Second end characterof a telegram (MODE 6 and 7 only)
SEND_Buffer	IN 22	INT	0…4000	Size of send buffer. Dependent on the size of the send telegram.
RCV_Buffer	IN 23	INT	0…4000	Size of receive buffer. Dependent on the size of the receive telegram.
Dummy_I1	IN 24	INT	0	Not used
Dummy_W1	IN 25	WORD	0	Not used (RK 512MP only)
Dummy_W2	IN 26	WORD	0	Not used (RK 512MP only)
Dummy_DW1	IN 27	DWORD	0	Not used (RK 512MP only)

RetVal	OUT	INT	-13...0	Error code: 0: Initialization successful -1: Invalid interface number -2: Not enough S7 memory to create the buffer Consider compression.- -3: Not enough S7 memory to create the buffer after compression. -4: Invalid value for the MODE parameter -5: Invalid interface parameter (baud rate, data bit, stop bit or parity)) -6: Invalid value for the WAIT_SEND or WAIT_INACTIVITY parameter -7: Invalid value for TEL_COUNT parameter -8: Invalid value for the ZVZ or QVZ parameter -9: Invalid value for the TRY_CONNECT or TRY_SEND parameter -10: ASCII - Fixed length: The telegram length is greater than the receive buffer size -11: Invalid value for parameter in SEND_ BUFFER or RCV_BUFFER -12: The total memory size of the send and receive buffers is greater than the maximum allowed 64kB. -13: The SFC was called with a different total send and receive buffer size to the last call.

Not required ▨ Required □

Programming example:

```
Network 1: Initialize serial interface 1, transparent mode

      CALL  “B_INIT“
       COM_NR       :=1                   // Interface no. 1
       MODE         :=0                   // Transparent mode
       BAUDRATE     :=L#9600              // 9600 baud
       DATA_BIT     :=8                   // 8 data bits
       STOP_BIT     :=1                   // 1 stop bit
       PARITY       :=0                   // No parity
       CONTROL      :=0                   // RS-232
       XON          :=B#16#0              // Not used
       XOFF         :=B#16#0              // Not used
       WAIT_SEND    :=W#16#0              // Not used
       WAIT_INACTIV :=W#16#0              // Not used
       TEL_COUNT    :=1                   // Number of telegrams in buffer
       OVER_WRITE   :=FALSE               // Telegram cannot be overwritten
       DEL_RX_BUFFER:=FALSE               // Not used
       DK_PRIORITY  :=FALSE               // Not used
       ZVZ          :=W#16#0              // Not used
       QVZ          :=W#16#0              // Not used
       TRY_CONNECT  :=0                   // Not used
       TRY_SEND     :=0                   // Not used
       FIXED_LENGTH :=0                   // Not used
       END_CHAR1    :=B#16#0              // Not used
       END_CHAR2    :=B#16#0              // Not used
       SEND_BUFFER  :=300                 // Size of send buffer
       RCV_BUFFER   :=300                 // Size of receive buffer
       Dummy_I1     :=0                   // Not used
       Dummy_W1     :=W#16#0              // Not used
       Dummy_W2     :=W#16#0              // Not used
       Dummy_DW1    :=DW#16#0             // Not used
       RET_VAL      :=MW246               // Return value
```

4.4 CAN communication

The PCD3.M6347 Series controllers have a CAN connection. This is programmed using SFC calls.

The CAN functionality is a Layer-2 interface. The user can choose between several operating modes, depending of the ease of use and functionality required. The user can access all functions via system functions.

The CAN interface does not provide any CANopen functions, but the available Layer-2 interface can easily be used to access CANopen slaves. For this, the user has to input all the relevant CANopen messages directly into the user program via the Layer-2 interface.

4.4.1 Overview

CAN Layer-2
The CAN controller on the PLC extension implements a CAN Layer-2 access to the bus. All Layer-2 protocol handling is performed by the controller. The user can send a message to the controller and specify when it was sent. The user can also specify which messages are to be received by the controller and read out of the controller memory after reception.

Operating modes

The CAN controller provides several buffers (message objects) which can be freely configured as transmission or reception buffers for any message ID. The user is provided with three different types of access to the controller functionality:

- **CAN Direct Access** (FullCAN): Direct access to all 32 buffers. The user can independently access all 32 buffers. The interface provides all functionality integrated in the CAN controller. This mode is similar to the FullCAN principle implemented in the controller itself.
- **CAN Basic Services** (BasicCAN): one transmit and one receive queue allow a simple handling of CAN communication in the user program. Several messages can be sent to the queue without waiting for confirmation of each message. The receive queue ensures that no message is lost (provided that the queue depth is designed accordingly). The user interface is considerably simplified in comparison to the CAN Direct Access interface. This mode is similar to the BasicCAN principle implemented in several simple CAN controllers with only one receive and one transmit path.
- **CAN Data Mapping**: Data mapping simplifies and automates the cyclic exchange of process data. This mapping is configured at start-up with the message IDs to be handled. The message data is directly mapped to process data of the PLC. All output messages are automatically sent by the data manager at specified intervals. The received messages are mapped to the process data by the manager.

The three modes can be used simultaneously with some limitations concerning the message ID ranges assigned to the different modes.

These components are only mentioned here for the sake of completeness. For more information, please e-mail support@saia-pcd.com, or refer to manual 26/840 “CAN Communication for xx7”.

4.4.2 CAN SFBs

The Step7 CAN interface contains a number of SFBs for configuration, data transfer and status.

General functions
SFB403: CAN_CFG, configure CAN driver

SFB404: CAN_STAT, query current status of CAN driver

These functions provide the basic configuration and the status setup required in any operating mode.

CAN Direct Access
SFB402: CAN_CFGO, configure message objects

SFB401: CAN_TX, send messages

SFB400: CAN_RX, receive messages

These functions provide direct, flexible access to the CAN controller hardware. The mode corresponds to the FullCAN principle implemented in the controller itself.

CAN Basic Services
SFB412: CAN_CFGQ, configure queues

SFB411: CAN_TXQ, send messages to the queue

SFB410: CAN_RXQ, receive messages from the queue

These functions provide additional hardware control, e.g. where direct hardware access is handled using queues and the user no longer needs to worry about low-level CAN handling. The concept corresponds to the BasicCAN principle implemented in some controllers.

CAN Data Mapping
SFB413: CAN_DMAP, configure data mapping.

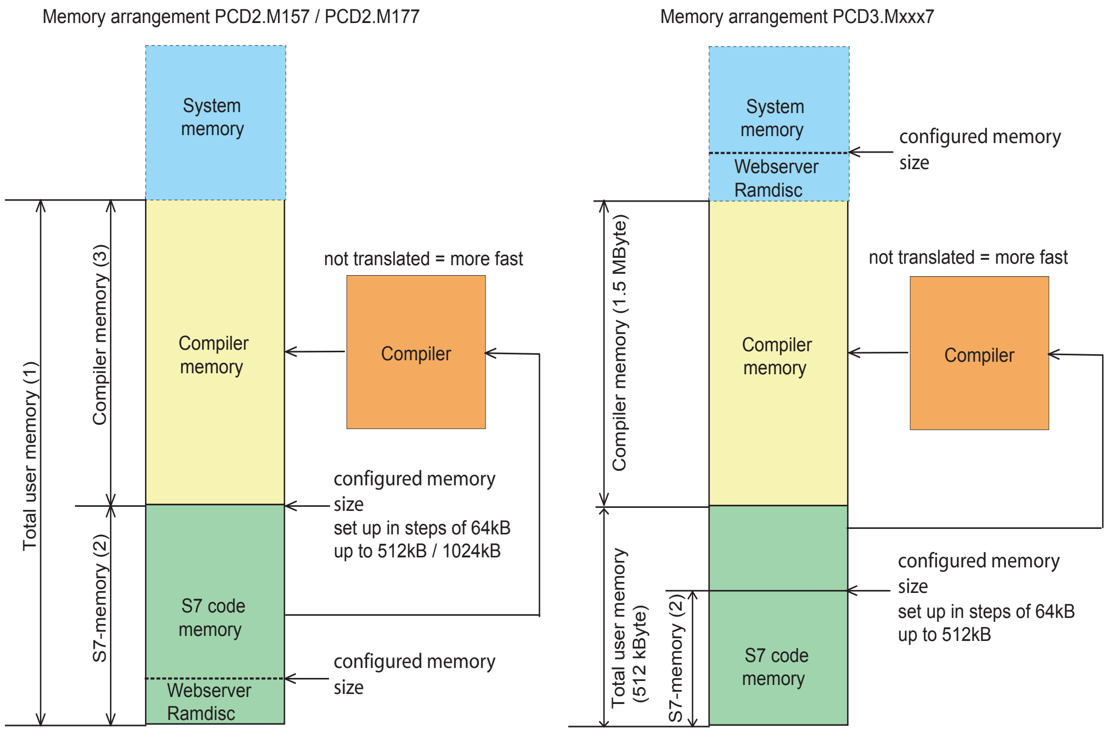
5 Memory

5.1 General

The size of the S7 code memory is dependent on the Saia PCD® type: for some PCD types, this size is fixed; for other types, the size of the S7 code memory can be freely configured in 64kB steps up to a maximum value, using the CDB (Configuration Data Block). The table below summarizes this.

PCD type	S7 code memory size
PCD1.M137	Default: 64 kB. Configurable from 64 kB to 128 kB.
PCD2.M127	Fixed 132 kB
PCD2.Mx57	Default: 256 kB. Configurable from 64 kB to 512 kB.
PCD2.M177	Default: 512 kB. Configurable from 64 kB to 1024 kB.
PCD2.M487	Default: 1 MB. Configurable from 64 kB to 1024 kB.
PCD3.Mxxx7	Default: 512 kB Configurable from 64 kB to 512 kB.

If the S7 code memory is not configured to the maximum configurable size on controllers of type PCD2.Mx57 and PCD2.M177, the unused area is allocated to the compiler. On controllers of type PCD1, PCD2.M487 and PCD3.Mxxx7, the surplus memory remains unused.


Memory arrangement PCD2.M157 / PCD2.M177

Memory arrangement PCD3.Mxxx7

The PCD2.Mx57, M177 and PCD3.Mxxx7 controllers have the capability to compile the application program. In contrast to interpreted code, where every command is translated from scratch at run time, compilation translates these commands into “CPU-language” before the execution of the program. By eliminating the translation of every single line of code, the same program can then be processed 2 to 3 times faster on average, depending on the program structure and commands used. This translation requires its own compiler memory space, located in the RAM area mentioned above.

The System Configuration section describes how the boundary between user and compiler areas is defined using an entry in the CDB.

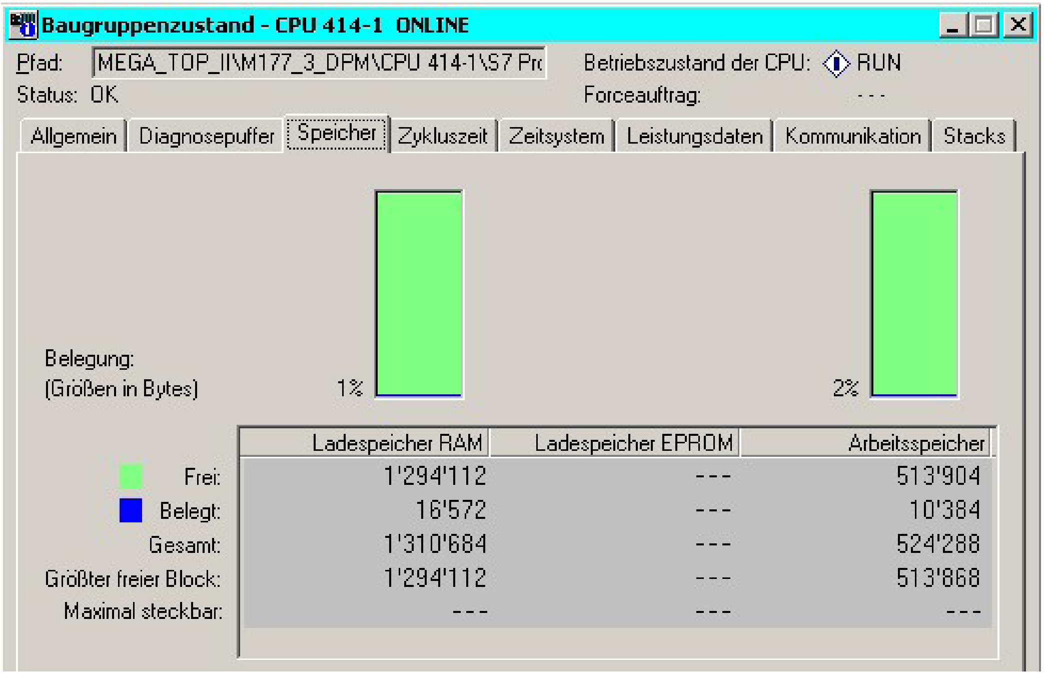
A Step7 instruction can result in up to 10 times more compiler code. With large programs, it may then happen that not all components can be compiled. In this case, any uncompiled components will be interpreted.

By prioritizing the components, the Saia PCD® cycle time can be optimized further. This is achieved particularly by compiling frequently called components. This prioritization is also controlled by parameters in the CDB.

If the web server is activated, the PCD2.M157 and M177 take the memory required for the RAM disk is taken from the S7 code memory. This means that the available memory for the Step7 components will be reduced by the size of the web server RAM disk. For the PCD2.M487 and the PCD3.Mxxx7, there is a specific memory area reserved for the RAM disk. The size of the RAM disk can be specified by a CDB entry. For further information, see the Web Server section.

Display of available and utilized memory in online set-up:

The sample screenshot below shows memory utilization for an M177.



Baugruppenzustand - CPU 414-1 ONLINE

Pfad: MEGA_TOP_II\M177_3_DPM\CPU 414-1\S7 Pr
Betriebszustand der CPU: RUN
Status: OK
Forceauftrag: - - -

Belegung:
(Größen in Bytes)

	Ladespeicher RAM	Ladespeicher EPROM	Arbeitsspeicher
🟩 Frei:	1'294'112	---	513'904
🟦 Belegt:	16'572	---	10'384
Gesamt:	1'310'684	---	524'288
Größter freier Block:	1'294'112	---	513'868
Maximal steckbar:	---	---	---

For the PCD2.M157 / M177 / PCD3, the meaning of the entries is as follows:

Load memory	Free	Total user memory - compiler memory in use. The freely available memory for the compiler = this value - configured S7 code memory
	In use	Memory utilized by the compiler
	Total	Total available user memory
Working memory	Free	Unused S7 code memory
	In use	Memory utilized by S7 components
	Total	Planned S7 code memory - web server RAM disk size

On a PCD2.M157/M177/PCD3, the % display of the load memory can never reach 100%.

5.2 Flash backup functions

5.2.1 Backup user program

Flash EPROMs on an xx7 controller can be used as backup memory for the user program, using the “Copy RAM to ROM...“ function from the SIMATIC® Manager “PLC target system” menu. The size of the backup memory area in the Flash EPROM matches that of the user memory area in RAM.

5.2.2 Memory data block

The unused Flash EPROM area can also be used as data block memory. This is divided into two equal memory areas. The size of the data block memory is dependent on the size of the configured user memory. The formula for calculating a data block memory area is as follows:

DB memory area = (size of Flash - size of user memory) / 2

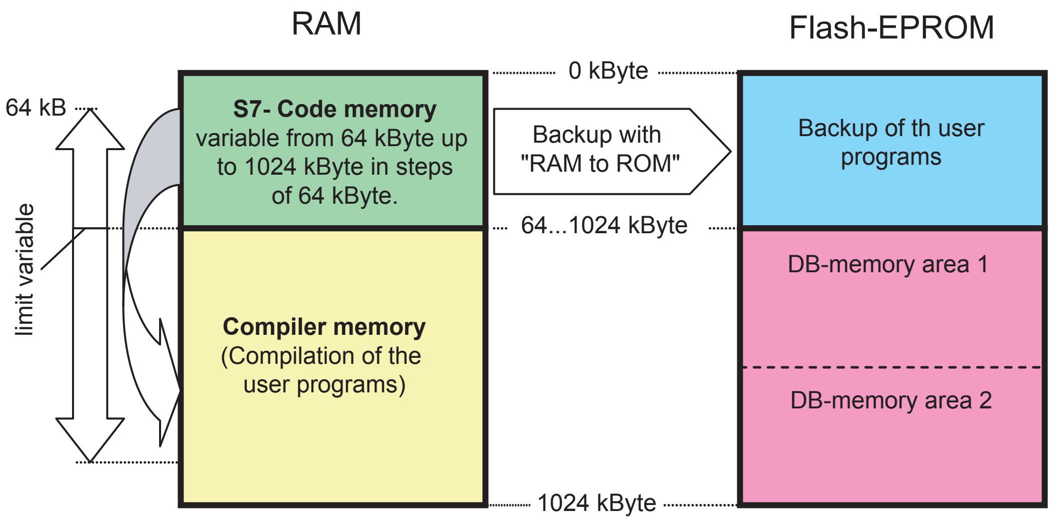
The table below shows the size of the data block memory area in the Flash EPROM for different Saia PCD® types.

PCD/Flash type	Flash EPROM size	Size of DB memory
PCD1.M137	128 kB	0 to 32 kB, in 32 kB steps
PCD2.M127	512 kB	190 kB, fixed
PCD2.Mx57	512 kB	0 to 224 kB, in 32 kB steps
PCD2.M177 / PCD7.R400	1 MB	0 to 480 kB, in 32 kB steps
PCD2.M487 / PCD7.R400	1 MB	0 to 480 kB, in 32 kB steps
PCD3.Mxxx7[1] / PCD7.R500	1 MB	512 to 992 kB, in 32 kB steps

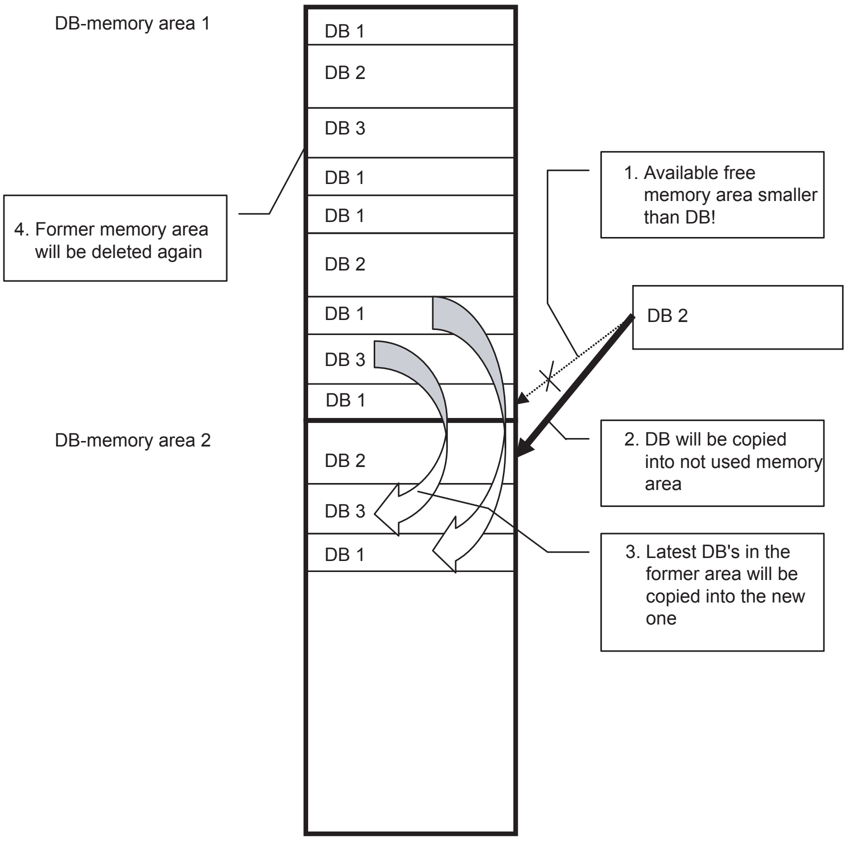
[1] The backup memory and DB Flash functionality are only available on Slot M1.

If the S7 code memory is configured to the maximum configurable size, there will be no DB memory available in Flash. (Does not apply to PCD3)

Example of memory allocation on a PCD2.M177:



The DB memory area must be partitioned, to prevent “over-filling” the Flash EPROM. This would occur relatively quickly, as old DB versions in Flash EPROM are not deleted but simply marked as no longer valid.
The DBs are written to a DB area with every write command. This is true even where the same DB is written to Flash several times. In this case, the latest DB is the valid one.


DB-memory area 1
DB-memory area 2
DB 1
DB 2
DB 3
DB 1
DB 1
DB 2
DB 1
DB 3
DB 1
DB 2
DB 3
DB 1
DB 2
1. Available free memory area smaller than DB!
2. DB will be copied into not used memory area
3. Latest DB's in the former area will be copied into the new one
4. Former memory area will be deleted again

If a DB is larger than the available memory area, it will be copied together with current DBs to the second DB area. The whole DB area with the old DBs will then be deleted, releasing the memory.

Of course it is still possible to fill the Flash EPROM so that no more memory can be released. To address this, a function has been provided to allow both memory areas to be deleted.

The “Save”, “Read” and “Delete” functions are implemented in SFB240.

SFC240 parameters:

Parameter	Type	Type	Range	Description
REQ	IN 0	BOOL		Start of processing on positive edge
FUNCTION	IN 1	BYTE	1...3	Function number B#16#1 : Back up DB to Flash B#16#2 : Restore DB B#16#3 : Delete DB areas
DB_NO	IN 2	INT		Number of data block to be backed up and restored
DONE	OUT 3	BOOL		Processing completed successfully
ERROR	OUT 4	BOOL		Error during processing
STATUS	OUT 5	WORD		Status/error information 1: A Flash function is already running 2: Unknown function requested. Function number outside the valid range 1...3 3: DB not found. DB to be backed up is not in the RAM user memory / the DB to be restored is not in the Flash EPROM data block memory area. 5: Flash EPROM is full. No further DB backup possible, until the data block memory areas are deleted (Function 3). 2F: Error when saving (e.g. not enough memory→compress) FF: Flash EPROM not present or defective AA: Flash EPROM data corrupt. DB structure incorrect (invalid size, wrong header code, etc.) F1: Write No error→Status display for SFB240 process F2: Read. No error→Status display for SFB240 process F3: Delete. No error→Status display for SFB240 process

5.3 File system

5.3.1 General

The PCD3 family provides various file systems. These can be accessed from the user program by means of the system functions (SFB450 to SFB453). It is also possible to access the file systems via FTP / Web server.

The functionality of the Web server is described in manual 26/791 “Ethernet xx7”.

The additional plug-in Flash modules provide a Flash file system. The following Flash modules are currently available:

Flash type	Flash size	Slot	Flash file system name
PCD7.R550M04	4 MB	M1 and M2	M1_Flash M2_Flash
PCD3.R550M04	4 MB	IO slots 0 to 3	SL0Flash SL1Flash SL2Flash SL3Flash
PCD3.R600	SD device 128MB to 1GB	IO slots 0 to 3	SL0Flash SL1Flash SL2Flash SL3Flash
PCD3 with 4 MB internal Flash *)	1MB	internal	INTFLASH
PCD3 with 4MB SRAM *)	1MB	internal (RAM)	USERSRAM

*) These versions are only available as OEM platforms.

The PCD7.R550M04 are sold ready-formatted as standard. They are ready for use as soon as they are plugged into slot M1 or M2 on the CPU housing.

They are organized into pages (256 bytes), blocks (2 kB) and sectors (64 kB). A page is the memory unit for writing, a block represents a file, and a sector is the memory unit that can be deleted.

The PCD7.R550M04 Flash itself is 4 MB in size. Note that 64 kB are reserved and some blocks are needed to store the internal file system. This means that slightly less than the total Flash memory is available to the user.

When a file is created, a block (2 kB) is reserved for it. Even if the file is smaller, these 2 kB are still reserved. When the 2 kB have been filled up, a new block is added, and size of this file unit is now 4 kB. The physical space taken up by a file is 2 kB * ((mod 2 kB of the file size) + 1).

When a file is deleted, the blocks that it used are marked as free. However, it is not possible to write new data to this area immediately. First, all bits in the block have to be set to “1”. This operation is complex and cannot be performed either automatically or by the user program. During this operation, the Flash is busy, and cannot be accessed (in read/write/list mode). During this “compression”, the sector is first copied to a reserved sector, the current compressed sector is deleted, and only the blocks currently used are copied back from the reserved sector. At the end of the operation, the reserved sector is also deleted. This sequence of operations is performed for all sectors in the Flash memory. In a 4 MB Flash, 63 sectors are affected.

If we assume that a sector can be deleted in 2 seconds (manufacturer’s figure), compressing all sectors of a Flash memory will take a maximum of around 400 seconds (over 6 minutes). During this time, the Flash is busy. Some sectors do not need to be compressed, because no blocks within them have been replaced. This significantly reduces the compression time.

The PCD7.R550M04 can be plugged in during use. This means that it can be plugged in at any time during the PLC operations. The file system is set up immediately after it is plugged in, and compression performed where necessary. This means that the Flash memory is visible to the user when it is plugged in, but it is quite possible that it will show as “busy”.

Although the PCD7.R550M04 can in principle be plugged in during use, this should be approached with caution. The Flash has no LED to display whether or not the Flash is currently being accessed.

- Two cases should be considered when unplugging the Flash: some write operations run continuously. These operations are triggered either by the user program or via FTP. Before unplugging the Flash, set the PLC to STOP if the user program is writing to the Flash, and ensure that there is no FTP connection.

- During the compression process: this operation is triggered either by the user program (directly or indirectly by deleting files) or indirectly via FTP after deleting files. These actions are not easily recognized by the user.

For these reasons, and because unplugging the Flash is not expressly forbidden, it is ***not*** advisable to unplug the Flash during use.

Caution is also advised when switching off the PLC. If the user program makes calls to the Flash file system (to write or compress), or an FTP client accesses the Flash while the PLC is switched off, the data recovery time may not be sufficient. At this point, a special algorithm will be triggered to recover as much data as possible, but it is possible that the last data/files written may be lost.

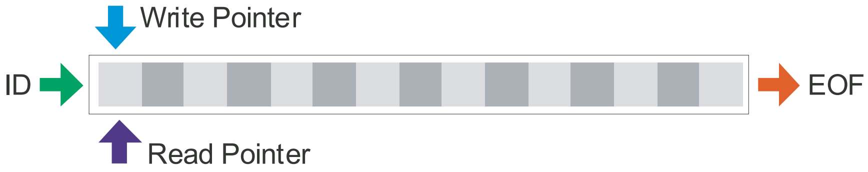
The following steps are required to use the file system functions:

1) Create and open a file.
2) Optional: position the read/write pointer in the file
3) Write the data from the PLC media to the file
4) Read the data from the file into the PLC media
5) Close the file

Functions are also provided for deleting files and for processing directories.

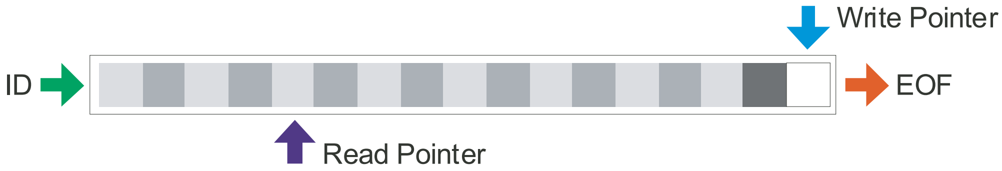
When a file is opened (created), a FileID is returned to the user. This FileID is also assigned two file pointers:

1) Write pointer
2) Read pointer



These pointers are modified by the read, write and search functions.

Example: After opening a file, 4 MB are read and 1 MB written at the end of the file. The file pointers will then have the following positions:



The next few sections describe the system function blocks for file access.

There is a library for these system functions. They can be downloaded from the support home page at www.sbc-support.com.

5.3.2 “FCREATE” SFB450

SFB FCREATE creates a new file. The FileName parameter is used to specify which file system and directory the file should be held in. The file name must always be given as an absolute file name.

SFC450 parameters:

Parameter	I/O	Type	Description
FileName	IN 0	STRING[64]	This parameter comprises the Flash file system name, the directory name and the actual file name. Only alphanumeric characters (no spaces) and “.” can be used for the file name. A file name may by up to 24 characters long. The maximum length of the absolute file name is 64 characters. Example: `M1 Flash:/Report.txt`
GroupID	IN 1	BYTE	Defines which group the new file belongs to. The following 4 groups are defined: 0x10 USER Group 1 0x20 USER Group 2 0x40 USER Group 3 0x80 USER Group 4
GroupAccess	IN 2	BYTE	This parameter can be used to define access groups. A combination of GroupIDs may be used in the definition. A file or a sub-directory can only be generated or deleted within a directory belonging to the relevant Group. The value “0” allows access to all Groups.
Handle	OUT 3	DINT	If it was possible to generate the file, the FileID is held here. This FileID remains valid until the file is closed using FCLOSE. If a negative value is present, the file could not be created. See section on error messages.

Programming example:

```
Network 1: Generate a file

      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 “FCREATE” SFB450

SFB FCREATE creates a new directory. The FileName is used to specify the which file system the directory should be held in. The directory name must always be given as an absolute name.

SFC451 parameters:

Parameter	I/O	Type	Description
FileName	IN 0	STRING[64]	This parameter comprises the Flash file system name and the directory name. Only alphanumeric characters (no spaces) can be used for the directory name. A directory name may by up to 24 characters long. Example: `M1 FLASH:/Config`
GroupID	IN 1	BYTE	Defines which group the new file belongs to. The following 4 groups are defined: 0x10 USER Group 1 0x20 USER Group 2 0x40 USER Group 3 0x80 USER Group 4
GroupAccess	IN 2	BYTE	This parameter can be used to define access groups. A combination of GroupIDs may be used in the definition. A file or a sub-directory can only be generated or deleted within a directory belonging to the relevant Group. The value “0” allows access to all Groups.
RetVal	OUT 3	DINT	If this function has been executed successfully, RetVal will equal “0”. If a negative value is present, the function could not be executed correctly. See section on error messages.

Programming example:

```
Network 1: Generate a directory

      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

SFB “FOPEN” opens a file. The FileName parameter is used to specify which file system and directory the file should be opened in. The file name must always be given as an absolute file name.

SFC452 parameters:

Parameter	I/O	Type	Description
FileName	IN 0	STRING[64]	This parameter comprises the Flash file system name, the directory name and the actual file name. Only alphanumeric characters (no spaces) and “.” can be used for the file name. A file name may by up to 24 characters long. The maximum length of the absolute file name is 64 characters. Example: `M1 FLASH:/Config/Maschine1.txt`
AccessType	IN 1	BYTE	Defines the access rights to the file to be generated. 0x01 Read only. The file can only be read. A write access to the file will generate an error. 0x02 Write only. The file can only be written to. A read access to the file will generate an error. 0x03 Read/Write. The file can be both read and written to.
Handle	OUT 2	DINT	If it was possible to open the file, the FileID is held here. This FileID remains valid until the file is closed using FCLOSE. If a negative value is present, the file could not be opened. See section on error messages.

Programming example:

```
Network 1: Open a file

      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

SFB “FSEEK” is used to manipulate the write or read pointer in an open file. Handle specifies in which file the pointers should be modified. The SeekPos parameter indicates the new position of the pointer in the file.

SFC453 parameters:

Parameter	I/O	Type	Description
Handle	IN 0	DINT	FileID returned when a file is opened/generated. Handle uniquely identifies the file to be processed.
SeekPos	IN 1	DINT	New position of the file pointer. The new position is given in relative terms. The value “0” sets both file pointers (read/write) to the beginning of the file.
AccessType	IN 2	BYTE	Defines which file pointer is being modified: 0x01 Read pointer modified. 0x02 Write pointer modified. 0x03 Read/write pointers modified.
RetVal	OUT 3	DINT	If this function has been executed successfully, RetVal will equal “0”. If a negative value is present, the function could not be executed correctly. See section on error messages.

Programming example:

```
Network 1: Manipulate file pointers

      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

SFB “FWRITE” is used to transfer data from a PLC medium to an open file. Handle specifies which file the data shoud be transferred into. The Buffer parameter passes the function an ANY pointer to the data to be transferred.

SFC454 parameters:

Parameter	I/O	Type	Description
Handle	IN 0	DINT	FileID returned when a file is opened/generated. Handle uniquely identifies the file to be processed.
WrAttr	IN 1	BYTE	Depending on this parameter, the data will be written either to the end of the file or to the position of the write pointer. 16 Data written to the current position of the write pointer, i.e. the data in the file will be overwritten. This operation is only valid for files in the RAM area. An error message will be returned for files in Flash units. 17 Data appended at the end of the file.
Buffer	IN 2	ANY	The data from the PLC media are passed with an ANY pointer. Databases, flags, input and output process maps can be passed as PLC media. The maximum length is limited to 256 bytes.
RetVal	OUT 3	DINT	If this function has been executed successfully, RetVal will equal “0”. If a negative value is present, the function could not be executed correctly. See section on error messages.

Programming example:

```
Network 1: Write data to a file

      L     17                          // Append
      T     #WriteAttribute

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

5.3.7 “FREAD” SFB455

SFB “FREAD” is used to transfer data from an open file to PLC media. Handle specifies which file the data shoud be transferred from. The Buffer parameter passes the function an ANY pointer to the data to be transferred.

SFC455 parameters:

Parameter	I/O	Type	Description
Handle	IN 0	DINT	FileID returned when a file is opened/generated. Handle uniquely identifies the file to be processed.
Buffer	IN 1	ANY	The data from the PLC media are passed with an ANY pointer. Databases, flags, input and output process maps can be passed as PLC media. The maximum length is limited to 256 bytes.
RetVal	OUT 2	DINT	If this function has been executed successfully, RetVal will be equal to the number of bytes transferred to the buffer. If a negative value is present, the function could not be executed correctly. See section on error messages.

Programming example:

```
Network 1: Read data from a file

      CALL  #FRead, DB455
       Handle:=#FileHandle
       Buffer:=“Blocks“.TempBlock    // Pointer at 256 bytes
       RETVal:=#ReturnValue
      L     #ReturnValue
      L     L#256
      ==D
      SPB   next
      SPA   erro
```

5.3.8 “FLENGTH” SFB456

SFB “FLENGTH” is used to read the size of an open file.

SFC456 parameters:

Parameter	I/O	Type	Description
Handle	IN 0	DINT	FileID returned when a file is opened/generated. Handle uniquely identifies the file to be processed.
RetVal	OUT 1	DINT	If this function has been executed successfully, RetVal will equal the file size. If a negative value is present, the function could not be executed correctly. See section on error messages.

Programming example:

```
Network 1: Read file size

      CALL  #FLENGTH, DB456
       Handle:=#FileHandle
       RetVal     :=#ReturnValue
      L     #ReturnValue
      T     #FileSize
      L     0
      >D                        // File greater than 0
      SPB   next
      SPA   erro
```

5.3.9 “FCLOSE” SFB457

SFB “FCLOSE” is used to close an open file. If the function could not be executed correctly, the FileID is no longer valid.

SFC457 parameters:

Parameter	I/O	Type	Description
Handle	IN 0	DINT	FileID returned when a file is opened/generated. Handle uniquely identifies the file to be closed.
RetVal	OUT 1	DINT	If this function has been executed successfully, RetVal will equal “0”. If a negative value is present, the function could not be executed correctly. See section on error messages.

Programming example:

```
Network 1: Close file

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

5.3.10 “Delete” SFB458

SFB “Delete” is used to delete a file or a whole directory. The FileName parameter is used to specify which file system and directory the file should be deleted from. The file name must always be given as an absolute file name. If only a directory name is given in the FileName parameter, this directory and its entire contents will be deleted.

SFC458 parameters:

Parameter	I/O	Type	Description
FileName	IN 0	STRING[64]	This parameter comprises the Flash file system name, the directory name and the actual file name or directory name.
GroupAccess	IN 1	BYTE	This parameter can be used to define access groups. A combination of GroupIDs may be used in the definition. A file or a sub-directory can only be generated or deleted within a directory belonging to the relevant Group. The value “0” allows access to all Groups.
RetVal	OUT 2	DINT	If this function has been executed successfully, RetVal will equal “0”. If a negative value is present, the file could not be opened. See section on error messages.

Programming example:

```
Network 1: Delete a file

      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

SFB “SWRITE” is used first to open (or generate) the file, then to copy the data from a PLC medium into the same file. At the end of the write process, the file is closed again. These three functions are executed in one call. The Buffer parameter passes the function an ANY pointer to the data to be transferred.

SFC459 parameters:

Parameter	I/O	Type	Description
FileName	IN 0	STRING[64]	This parameter comprises the Flash file system name, the directory name and the actual file name. Only alphanumeric characters (no spaces) and “.” can be used for the file name. A file name may by up to 24 characters long. The maximum length of the absolute file name is 64 characters. Example: `M1 FLASH:/Config/Maschine1.txt`
GroupID	IN 1	BYTE	Defines which group the new file belongs to. The following 4 groups are defined: 0x10 USER Group 1 0x20 USER Group 2 0x40 USER Group 3 0x80 USER Group 4
GroupAccess	IN 2	BYTE	This parameter can be used to define access groups. A combination of GroupIDs may be used in the definition. A file or a sub-directory can only be generated or deleted within a directory belonging to the relevant Group. The value “0” allows access to all Groups.
Buffer	IN 3	ANY	The data from the PLC media are passed with an ANY pointer. Databases, flags, input and output process maps can be passed as PLC media. The maximum length is limitedto 256 bytes.
RetVal	OUT 4	DINT	If this function has been executed successfully, RetVal will equal “0”. If a negative value is present, the function could not be executed correctly. See section on error messages.

Programming example:

```
Network 1: Write data to a file

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

5.3.12 “SREAD” SFB460

SFB “SREAD” is used first to open (or generate) the file and set the read pointer to the Offset position, then to copy the data from file to the relevant PLC media. At the end of the read process, the file is closed again. These four functions are executed in one call. The Buffer parameter passes the function an ANY pointer to the PLC media to be transferred.

SFC460 parameters:

Parameter	I/O	Type	Description
FileName	IN 0	STRING[64]	This parameter comprises the Flash file system name, the directory name and the actual file name. Only alphanumeric characters (no spaces) and “.” can be used for the file name. A file name may by up to 24 characters long. The maximum length of the absolute file name is 64 characters. Example: `M1_FLASH:/Config/Maschine1.txt`
Buffer	IN 1	ANY	The data from the PLC media are passed with an ANY pointer. Databases, flags, input and output process maps can be passed as PLC media. The maximum length is limitedto 256 bytes.
Offset	IN 2	DINT	Sets the read pointer to the specified position when the file is opened. The value “0” indicates the beginning of the file. This parameter must be greater than or equal to “0”.
RetVal	OUT 3	DINT	If this function has been executed successfully, RetVal will be equal to the number of bytes transferred to the buffer. If a negative value is present, the function could not be executed correctly. See section on error messages.

Programming example:

```
Network 1: Read data from a file

      CALL  #SRead, DB460
       FileName   :=“FileNames“.M1Flash_DataFile
       Buffer     :=“Blocks“.TempBlock             // Pointer at 256 bytes
       Offset     :=#ReadOffset
       RETVal     :=#ReturnValue
      L     #ReturnValue
      L     L#256
      ==D
      SPB   next
      SPA   erro
```

5.3.13 “FSCREATE” SFB461

This FB handles the formatting/creation of a file system within a specified unit. This operation runs asynchronously; to complete the operation, more than one call is required.

The actual operation starts when the Req parameter is set to “1”. After the first call, the value of the Req parameter is not read again as long as the Done parameter is set to “1”. When “Start” is recognized, the parameters are initialised and the first call returns to FIRST_CALL (0x7001). The function then moves to INTERIM_CALL (0x7002) until it completes.

When the Busy parameter goes back from “1” to “0”, the RetVal parameter can be read. The function has been completed successfully if a “0” is returned in RetVal. In the event of an error, a negative value will be returned. (See 5.3.16 Error information). In the event of an error, the error message will remain until the Req parameter is set to “0”. With the Req parameter set to “0”, the function returns to CALL_WITHOUT_EXEC (0x7000).

The next call after the completion of this function with Req = “1” will restart the FSCREATE function.

Calling this function with the Force parameter set to “1” will delete all the data in this unit. Calling this function with the Force parameter set to “0” before recognising the unit will start formatting, and any data in this unit will be deleted.

SFC461 parameters:

Parameter	I/O	Type	Description
Req	IN 0	BOOL	The actual operation starts when the Req parameter is set to “1”. If the Done parameter is equal to “1“, the Req parameter will not be read.
Force	IN 1	BOOL	The Force parameter can be set to “0” or “1”. When set to “0”, it allows the function to be called without forcing formatting of the unit, if it already exists. When set to “1”, it allows the function to be called and forces the formatting of the unit, even if there is already a file system on it. This operation deletes all the data on the unit and triggers formatting. If the Force parameter is set to “0” and the unit exists but is not recognised as a file system, formatting will be carried out. All data previously stored on the unit will be lost, e.g. backup data written to the unit.
FSName	IN 2	STRING[16]	This parameter defines which Flash system is to be formatted.
BlockSize	IN 3	WORD	This parameter is not currently supported. The value must be set to 2048.
BlkNbr	IN 4	WORD	This parameter is not currently supported. The value must be set to 256.
MNbr	IN 5	WORD	This parameter is not currently supported. The value must be set to 32.
RetVal	OUT 6	DINT	The RetVal parameter is set during execution to CALL_WITHOUT_EXEC (0x7000), FIRST_CALL (0x7001) or INTERIM_CALL (0x7002). The actual return value is set when the Busy parameter changes from “1” to “0”. A return value of “0” at this point means that everything has been processed successfully and that the unit can be accessed with other FBs.
Busy	OUT 7	BOOL	The Busy parameter is set to “1” as long as the function is running and then set to “0” when it is completed.
Done	OUT 8	BOOL	The Done parameter is set to “0” as long as the function is running and then set to “1” when it is completed successfully. In the event of an error, Done will **not** be set to “1”.

Programming example:

```
Network 1: Format a file system

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

5.3.14 "FSCPRESS" SFB462

This FB handles the compression/recovery of blocks released within a file system in a given unit. This operation runs asynchronously; to complete the operation, more than one call is required.

On the R55Mxx Flash module, a block is marked wither as free (not used) or used (just used) and released (used earlier). Released blocks cannot be used until the sector containing the block has been deleted (all bits set to "1"). Only then can a released block be added to the list of free blocks.

Within the file system, some blocks may be marked as released, but a block is mainly marked as released when a file/directory is deleted, deleting all the blocks contained within it.

Within the file system, compression (recovery of released blocks) is triggered automatically when certain criteria are met, e.g. the number of released blocks amounts to 80% of the total blocks or the number of released blocks is greater than the number of free blocks, where this is less than 1/4 of the total blocks. This can happen at any time, even during an operation, and even when the file system is showing as "busy". All calls to the file system FBs will then retrurn the code FS_DEVICE_BUSY.

This FB can be used to force the compression of the unit, even if the previous criteria are not met, e.g. a file is being deleted and the user wants to recover all blocks in this file immediately.

The function is launched when the Req parameter is set to "1".Upon the first call, the Bust parameter s set to "1" and the Done parameter to "0". The RetVal parameter will return FIRST_CALL (0x7001). Further calls to this function will return INTERIM_CALL (0x7002) until it has finished processing.

With the Req parameter set to "0", the function returns CALL_WITHOUT_EXEC (0x7000).

If the Busy parameter is set to "0", RetVal can be read. In the event of an error, RetVal will return a negative value (see 5.3.16 Error information). The error will remain until the Req parameter is set to "0". If the function was completed successfully, RetVal wil be set to "0".

If the functions are called again after completion with Req="1", FSCPRESS will restart.

SFC462 parameters:

Parameter	I/O	Type	Description
Req	IN 0	BOOL	The actual operation starts when the Req parameter is set to “1”. The value of the Req parameter will not be read again during execution (while Busy is set to “1”).
DRVName	IN 1	STRING[16]	This parameter defines which Flash file system is to be formatted.
Busy	OUT 2	BOOL	The Busy parameter is set to “1” as long as the function is running and then set to “0” when it is completed.
Done	OUT 3	BOOL	The Done parameter is set to “1” as soon as the function is completed successfully.
RetVal	OUT 4	DINT	The RetVal parameter is set during execution to CALL_WITHOUT_EXEC (0x7000), FIRST_CALL (0x7001) or INTERIM_CALL (0x7002). The actual return value is set when the Busy parameter changes from “1” to “0”. In the event of an error, the RetVal parameter will indicate the error that has occurred. If the function has been processed successfully, the RetVal parameter will be set to “0”.

Programming example:

```
Network 1: Compress a file system

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

Because of the Flash technology used in the PCD7.R55x modules, this function takes a very long time to execute. Depending on the number of blocks and sectors to be compressed, execution may last between 1 second and 4 minutes. The controller should not be switched off during this time.

If the controller is switched off while the FSCPRESS function is running, the compress algorithm will be run again at start-up. This may result in this unit being unavailable for several minutes after start-up.

5.3.15 “FSGETSIZ” SFB463

The FSGETSIZ function can be used to read information from a specific unit.

SFC463 parameters:

Parameter	I/O	Type	Description
DRVName	IN 0	STRING[16]	This parameter defines which Flash system information should be read from.
TSIZE	OUT 1	DINT	Size of unit.
FSIZE	OUT 2	DINT	Size of free memory.
USIZE	OUT 3	DINT	Size of memory in use.
RetVal	OUT 4	DINT	If this function has been executed successfully, the RetVal parameter will equal “0”. If a negative value is present, the function could not be executed correctly. See section on error messages.

Programming example:

```
Network 1: Read device information

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

5.3.16 Error information on file system SFBs

In the event of an error, SFBs 450 to 463 contain further error information in the RetVal variable. Some of the error codes follow the S7 standard. Some error codes have been added specifically for file system functions.

RetVal	Description
0x7000	Initial call with REQ = 0: function not active
0x7001	Initial call with REQ = 1: function started
0x7002	Intermediate call (REQ irrelevant): function already active
0x0000	Function completed successfully
0xFF9B	Internal file system error (PLC)
0xFF9C	Invalid type
0xFF9D	Unit not found
0xFF9E	Invalid parameter
0xFF9F	Invalid argument
0xFFA0	File does not exist
0xFFA1	Invalid file name
0xFFA2	Invalid group
0xFFA3	Invalid level
0xFFA4	Invalid access type
0xFFA5	Invalid unit name
0xFFA6	Invalid directory name
0xFFA7	File already exists
0xFFA8	Not enough memory (call to FSCPRESS function needed)
0xFFA9	Max. no of open files exceeded.
0xFFAA	File not open
0xFFAB	File already open
0xFFAC	Invalid access type
0xFFAD	Invalid file type
0xFFAE	Invalid write attribute
0xFFAF	Invalid parameter buffer
0xFFB0	Error during write operation
0xFFB1	Error during read operation
0xFFB2	DAS access not possible
0xFFB3	Access not possible
0xFFB4	Invalid FileID
0xFFB5	Invalid user
0xFFB6	Invalid REGFLAGS
0xFFB7	REG_ENTRY_TABLE_FULL
0xFFB8	INVALID_REGID
0xFFB9	FILE_SYSTEM_CHECK_ERROR
0xFFBA	Invalid name for unit
0xFFBB	Unit not loaded
0xFFBC	Name for unit already exists
0xFFBD	Invalid operation
0xFFBE	Invalid Flash value
0xFFBF	Flash operation could not be executed
0xFFC0	Error during compression process
0xFFC1	Unit busy
0xFFC2	Operation will be executed later
0xFFC3	Function not implemented
0xFFC4	Internal file system error

If a function returns the error message “Unit busy” (0xFFC1), the user should call this function again later. This may be the case where a unit is being compressed, and no data can be written to this unit during this time.

6 PC104 functions for the PCD2.M257 dual-port RAM

6.1 General

On the PCD2.M257 controller (PCD2.M157 with IPC added), dual-port RAM (DPR) offers 2 options for data transfer.

- S-Bus mode
- Transparent mode

The DPR provides an area of memory for the exchange of data between the PLC and the PC104 board. It allows both sides to send or receive data.

In **S-Bus mode**, no further programming is needed on the PLC side, i.e. the exchange of data via DPR is not available from the Step®7 side (no event-driven transmission from the PLC).

In **Transparent mode**, programming is needed on the PLC side. The PLC is active, i.e. it can send data to the PC104 on an event-driven basis. Together with 2 send and 2 receive buffers, this allows a high rate of data interchange to be achieved.

On the PLC side, there are 3 SFCs for Transparent mode, enabling the S7 application to read data from the PC104, or write data to it.

- Reading from dual-port RAM, SFC227, PC104_RD
- Writing to dual-port RAM, SFC228, PC104_WR
- Querying the status of dual-port RAM, SFC 229, PC104_ST

For further information on the software components on the PC side, refer to manual 26/759 “PLCs with Integrated IPC”.

6.2 Reading from dual-port RAM (SFC227, PC104_RD)

SFC227 reads the number of bytes selected from one of the 2 receive buffers and stores them in the relevant data area. There is an ID in the data packet received, to identify it.

SFC227 parameters:

Parameter	I/O	Type	Range	Description
BUF_NO	IN 0	INT	0...2	This parameter allows the user to choose to read the data from a specific receive buffer or to have the Saia PCD® retrieve the data from the “oldest” buffer. 0: first available 1: Receive buffer 1 2: Receive buffer 2
DEST	IN 1	ANY	Valid ANY pointer	Specifies the target area to which the data should be transferred. The length details are ignored. The type must be BYTE.
LEN	IN_OUT 0	INT	Valid length	Size of user target area. This must be greater than or equal to the expected max. data volume. After transmission, “LEN” is set to the number of bytes transferred.
ID	OUT 0	WORD	Valid ID	The ID of the header is returned.
Ret_Val	OUT 1	INT	Error messages	Error message: Apart from the Step®7-specific error messages (e.g. invalid pointer or DB not found etc.), the following error numbers may occur: 0xFFFF: No buffer free 0xFFFE: Invalid buffer number 0xFFFD: Buffer too small 0xFFF0: Protocol error (i.e. the PC has not switched protocol).

Programming example:

```
Network 1: Read data from dual-port RAM

      CALL  “PCD104_RD‟
       BUF_NO :=2
       DEST   :=DB120.DBD4            // Target area for data to be read
       ID     :=MW120
       Ret_val:=MW122
       LEN    :=MW124
```

6.3 Writing to dual-port RAM (SFC228, PC104_WR)

SFC228 writes the number of bytes selected from the relevant data area to the DPR. It also sends an ID to identify the data packet.

SFC228 parameters:

Parameter	I/O	Type	Range	Description
BUF_NO	IN 0	INT	0...2	This parameter allows the user to choose to write the data to a specific send buffer or to have the Saia PCD® write the data to an available buffer. 0: first available 1: Receive buffer 1 2: Receive buffer 2
SRC	IN 1	ANY	Valid ANY pointer	Specifies the source area from which the data should be transferred to DPR. The length details are ignored. The type must be BYTE.
LEN	IN 2	INT	Valid length	Size of user source area.
ID	IN 3	WORD	Valid ID	This ID is entered into the header.
Ret_Val	OUT 0	INT	Error messages	Error message: Apart from the STEP®7-specific error messages (e.g. invalid pointer or DB not found etc.), the following error numbers may occur: 0xFFFF: No buffer free 0xFFFE: Invalid buffer number 0xFFFD: Buffer too small 0xFFF0: Protocol error (i.e. the PC has not switched protocol).

Programming example:

```
Network 1: Write data to dual-port RAM

      CALL  “PCD104_WR“
       BUF_NO :=2
       SRC    :=DB130.DBD4       // Source area for data to be written
       LEN    :=20
       ID     :=MW150
       RET_VAL:=MW228
```

6.4 Status of dual-port RAM (SFC229 PC104_ST)

SFC229 supplies the status of the dual-port RAM buffer. It returns the synchro bytes, whether a buffer is in use, and the number of bytes in the receive buffers.

SFC229 parameters:

Parameter	I/O	Type	Range	Description
BUFBITS	OUT 0	WORD	Status of dual-port RAM	This byte is the XOR result from the two semaphore bytes in the dual-port RAM. Bit 0 = 1: Receive buffer 1 in use Bit 1 = 1: Receive buffer 2 in use Bit 4 = 1: Send buffer 1 in use Bit 5 = 1: Send buffer 2 in use
LEN_1	OUT 1	INT	Number of bytes	Number of bytes in receive buffer 1.
LEN_2	OUT 2	INT	Number of bytes	Number of bytes in receive buffer 2
Ret_Val	OUT 3	INT	Error display	Error message: 0xFFF0: Protocol error (i.e. the PC has not switched protocol).

Programming example:

```
Network 1: Status of dual-port RAM

      CALL  “PCD104_ST“
       BUFBITS:=MW300                  // Buffer in use?
       LEN_1  :=MW302                  // Number of bytes in receive buffer 1
       LEN_2  :=MW304                  // Number of bytes in receive buffer 2
       Ret_val:=MW230
```

7 Smart7 functions

7.1 General

The Smart7 is a credit-card sized module with full PLC functionality, offering simple integration with user-specific electronic controls. An evaluation board is available as a development environment for the 2 CPUs PCD.Smart.M137 and PCD.Smart.M177. Access to the interfaces from the user program in the CPU is via existing interface modules, as with previous Saia PCD® Series xx7 controllers.

Access to user-developed functions for the parallel bus is via two SFBs. These are:

- Write access to chip select, SFB254, WriteCS
- Read access to chip select, SFB255, ReadCS

! The SFB254 and SFB255 components are only present in the operating system in Smart7 CPUs, and can only be run there.

If a component is to be used in multi-instance mode, the Step®7 software must first create the instance DB via an SFB call. The SFB call must be made from the FB for the multi-instance. The SFB can then be declared as a static variable.

i These components are only mentioned here for the sake of completeness. For more information, please contact our field service staff, or e-mail: support@saia-pcd.com

7.2 Smart7 write access to chip select (SFB254, WriteCS)

A call to SFB254 provides write access to the two chip selects (CSDRT and CS_F2) that address the parallel bus (SLOT B1 and B2). This SFB is completely transparent and provides direct access in byte, word or double-word format. A single call causes the source data area to be transferred to the parallel bus.

SFC254 parameters:

Parameter	I/O	Type	Range	Description
AREA	IN 0	INT	0...2	Choice of chip select (slot number) 0: Not used 1: Slot B1 (CSDRT) 2: Slot B2 (CS_F2)
ACCESS	IN 1	INT	0...2	Data access: 0: Byte (size=1) 1: Word (size=2) 2: Double-word (size=4) Word and double-word must start at an even address.
OFFSET	IN 2	WORD		Byte offset in the data area (AREA)
LEN	IN 3	INT	0...32767	Length of the area to be written in byte, word or double-word format, depending on the ACCESS parameter. Value = length in bytes * size (ACCESS)
P_SRC	IN 4	ANY		Pointer to start address of source area
RET_VAL	OUT 5	INT	-6...0	Return values: 0: OK -1: Wrong slot no. (AREA) -2: Wrong access (ACCESS) -3: Error in pointer to source area (P_SRC) -4: Wrong length (LEN) -5: Overflow in pointer to source area (P_SRC) -6: Word or double-word access at un-even address

Programming example:

```
Network 1: Write access to parallel bus

      U     E      0.0                 // Write request
      FP    M      0.0
      SPBN  nWRT                       // Pulse
      CALL  SFB 254 , DB254
       IN0 :=1                         // Slot B1, CsDRT
       IN1 :=0                         // Byte access
       IN2 :=W#16#0                    // Byte offset
       IN3 :=20                        // 20 bytes
       IN4 :=DB11.DBX0.0               // Start address of source area
       OUT5:=MW200
nWRT: NOP 0
```

7.3 Smart7 read access to chip select (SFB255, ReadCS)

A call to SFB255 provides read access to the two chip selects (CSDRT and CS_F2) that address the parallel bus (SLOT B1 and B2). This SFB is completely transparent and provides direct access in byte, word or double-word format. A single call causes the selected target data area to be transferred from the parallel bus to the data area of the user program.

SFC255 parameters:

Parameter	I/O	Type	Range	Description
AREA	IN 0	INT	0...2	Choice of chip select (slot number) 0: Not used 1: Slot B1 (CSDRT) 2: Slot B2 (CS_F2)
ACCESS	IN 1	INT	0...2	Data access: 0: Byte (size=1) 1: Word (size=2) 2: Double-word (size=4) Word and double-word must start at an even address.
OFFSET	IN 2	WORD		Byte offset in the data area (AREA)
LEN	IN 3	INT	0...32767	Length of the area to be written in byte, word or double-word format, depending on the ACCESS parameter. Value = length in bytes * size (ACCESS)
P_SRC	IN 4	ANY		Pointer to start address of target area
RET_VAL	OUT 5	INT	-6...0	Return values: 0: OK -1: Wrong slot no. (AREA) -2: Wrong access (ACCESS) -3: Error in pointer to target area (P_SRC) -4: Wrong length (LEN) -5: Overflow in pointer to target area (P_SRC) -6: Word or double-word access at uneven address

Programming example:

```
Network 1: Read access to parallel bus

      U     E      0.0                     // Read request
      FP    M      0.0
      SPBN  nRD                            // Pulse
      CALL  SFB 255 , DB255
       IN0 :=1                             // Slot B1, CSDRT
       IN1 :=0                             // Byte access
       IN2 :=W#16#0                        // Byte offset
       IN3 :=20                            // Read 20 bytes
       IN4 :=DB11.DBX0.0                   // Start address of target area
       OUT5:=MW200
nRD:  NOP 0
```

8 System configuration (CDB)

8.1 General

Saia PCD® types Series xx7 have configurable properties that cannot be set with the SIMATIC Manager. Before the introduction of the compiler on the PCD2.Mx57, it was only possible to configure the I/Os. The introduction of the Configuration Data Block (CDB) now allows system settings as well as I/Os to be configured. The CDB is designed in such a way that the configuration details can be entered as ASCII text. Most settings can be made via the I/O Builder. This automatically generates the necessary CDB entries. The I/O-Builder can be downloaded free of charge from www.sbc-support.com.

The following configuration options are available:

- Memory allocation
- Priorities for compiling
- Port and modem initialisation
- Web server
- Peripheral access in OB100
- Configuration of Profi-S-IO-Master / MPI interface

Design and structure of the CDB

The CDB is recognized by the operating system under the following conditions:

- The data block number must be DB1, DB511 or DB1023
- The code “SBC xx7 CDB” must appear at the start of the DB (case-sensitive)

This ID must be declared as a STRING with at least 12 characters. If a longer string is declared, only the first 12 characters will be read as the ID. The user can append other information (e.g. version number) to this string.

Example:

Address	Name	I/O	Initial value
0.0		STRUCT	
+0.0	Identificator	STRING[12]	‘SBC xx7 CDB’
=14.0		END_STRUCT	

The Configuration Data Block consists of strings, to allow the information to be read in text format. Note the following string syntax:

- All spaces are ignored when the string is interpreted. They are present for ease of reading only.
- Lower-case characters will be converted to upper-case.

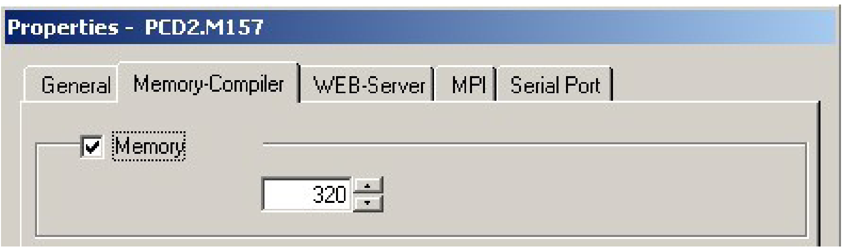
In general, the string comprises a keyword and parameters. The keyword ends with a colon ":". The parameters are dependent on the particular functionality they are configuring. When entering the string, the following rules apply:

- Parameter values are separated by commas ","
- Parameter names are separated from the values by an equals sign "="
- Comments in the CDB start with "//"

Interpretation of the CDB
The CDB is read after Power On and after a Stop Run transition. The memory and communication parameters are only read after Power On.

8.2 Memory scaling

In most Series xx7 controllers, the S7 code memory is scalable. By double-clicking on the relevant controller in the I/O Builder, the memory can be configured in 64 kB steps.



☑ Memory

320

The maximum memory size for the various controllers is shown in the table below:

Saia PCD® type	S7 code memory size
PCD1.M137	Default: 64 kB. Configurable to 64 kB or 128 kB.
PCD2.M127	Fixed 132 kB
PCD2.Mx57	Default: 256 kB. Configurable from 64 kB to 512 kB.
PCD2.M177	Default: 512 kB. Configurable from 64 kB to 1024 kB.
PCD2.M487	Default: 1 MB. Configurable from 64 kB to 1024 kB.
PCD3.Mxxx7	Default: 512 MB. Configurable from 64 kB to 512 kB.

On the PCD2.Mx57 and PCD2.M177 controllers, the whole user memory is split into S7 code memory and compiler memory. For some applications, more S7 memory may be required, e.g. where an S7 project has a high proportion of DBs and few program components. When allocating S7 code memory, it is important to note how much memory is left for the compiler code. The smaller the S7 code memory, the larger the compiler memory.

This restriction does not apply to the PCD2.M487 and the PCD3.Mxxx7.More details can be found in the section on Memory/Flash functions.

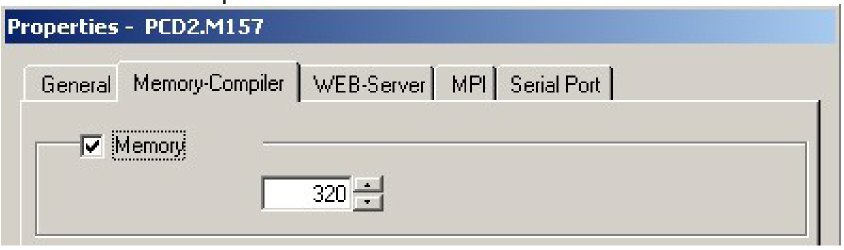
If the functionality of the DB memory is used, it should be noted that the available DB Flash memory space is dependent on the configured S7 code memory size. More details can be found in the section on Flash functions.

Keyword: MEM7

Parameter: S7 code memory size. The size of the S7 code memory must be at least 64 or a multiple of 64 kB. Zero values or values that are not multiples of 64 will be ignored, and have no effect on the controller. If too large a value is set, the memory will be configured to the maximum possible value.

The memory configuration is only read after Power On.

I/O-Builder example:



Properties - PCD2.M157

☑ Memory

320

The I/O Builder settings shown above generate the following CDB.

Example setting in the CDB:

Address	Name		I/O	Initial value
0.0			STRUCT	
+0.0		Identificator	STRING[12]	'SBC xx7 CDB'
+14.0		Memory	STRING[8]	'MEM7:320'
=24.0			END_STRUCT	

The memory available for S7 code is set to 320 kB.

8.3 Priorities for compiling

At every Stop Run transition, an attempt is made to translate the whole Step®7 program. This process may take several seconds. During this time, the 'RUN' LED will flash. As pre-set, components will be translated in the following sequence:

- FCs in descending order,
- FBs in descending order,
- OBs in descending order

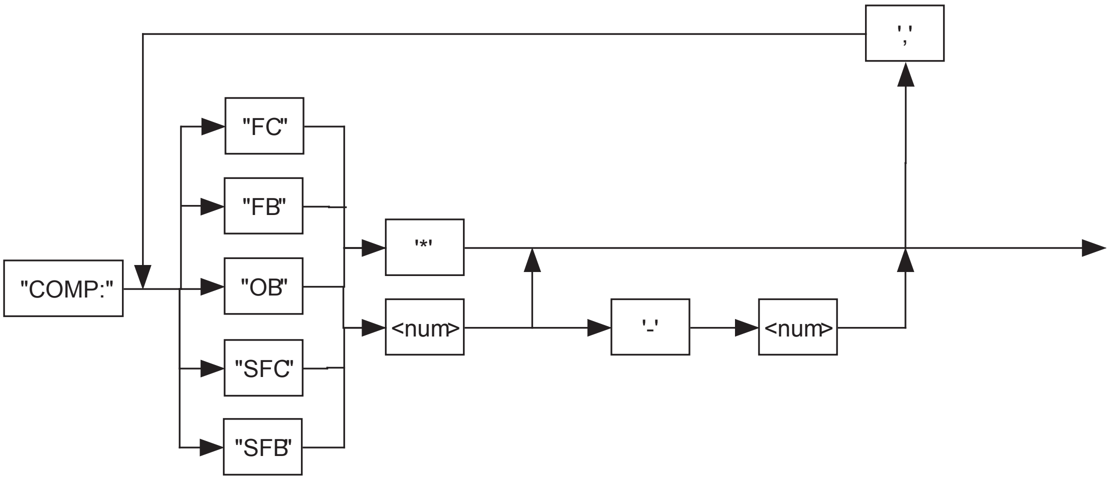
Where not all components can be compiled, the CDB can be used to define a different sequence. This is done using the keyword "COMP". If there is still space in the compiler memory after prioritized compilation, further components will be translated in the pre-set sequence.

Where components are transferred to a Run state, they will be translated if possible. However, this cannot take account of any prioritization. It may well be that a component that was available as a compiled module can no longer be compiled after a change. In this case, a Stop Run transition may cause the component in question to be re-compiled.

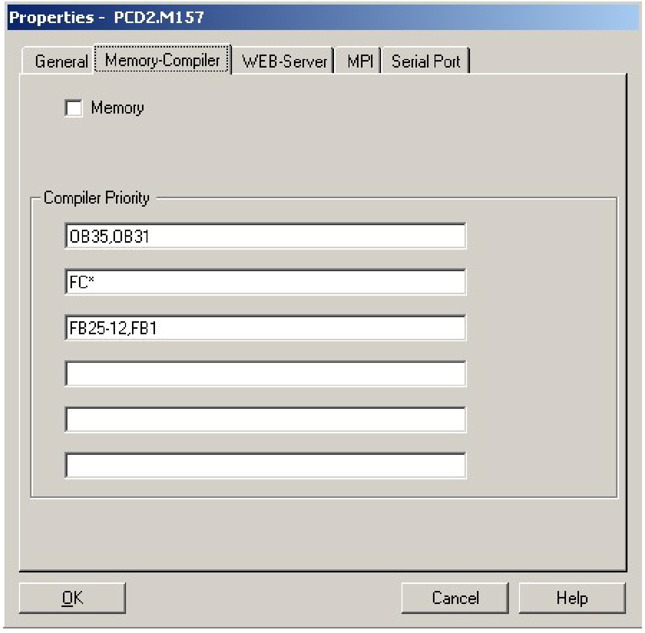
Keyword: COMP:

Parameter: Component type.
'*' All components of the given type will be prioritized for compilation in ascending order. (Reverse of pre-set order)
'-' Component range: If the higher component number is given first, the range will be compiled in descending order.

Syntax diagram:



Example setting in the I/O-Builder:



□ Memory

Compiler Priority

OB35,OB31

FC*

FB25-12,FB1

The I/O Builder settings shown above generate the following CDB.

Example setting in the CDB:

Address	Name	I/O	Initial value
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	

In the example above, the OB35 is compiled first, then the OB31. All FCs are then compiled in ascending order. If there is still enough compiler memory available, FBs 12-25 are compiled in descending order (FB25-FB12). Next, the FB1 is compiled. If there is still free compiler memory available, the remaining FBs will be compiled in descending order. Finally, the remaining OBs will be compiled in descending order.

8.3.1 Reading the compiler status (SFC230)

SFC230 allows the user to test whether the current component is being interpreted or running as compiled code. The user can also use SFC300 to test whether there is compiled code for a particular component. SFC230 has no parameters. It interprets Akku 1 as an input parameter. Feedback is via the status word (bits A0, A1 and VKE).

Input parameter in Akku 1:

Akku 1	
MSW	LSW
Type code (0x08 = OB, 0x0C = FC, 0x0E = FB)	Component number

Meaning of return data:

VKE	A0	A1	Description
0	x	x	Current component being interpreted
1	x	x	Current component running as compiled code
x	0	0	Component not loaded
x	1	0	Component not compiled
x	0	1	Component available as compiled code
x	1	1	Invalid parameter passed to Akku 1.

Example:

```
Network 1: Check whether current component is running as compiled code

       UC   SFC 230                     // Call
       SPB  Comp
// Component being interpreted
       .
       .
       BEA
Comp:  NOP 0
// Component running as compiled code
       .
       .
```

Example:

```
Network 1: Check whether FB 1 available as compiled code

       L    DW#16#000E0001              // Info on FB 1
       UC   SFC 230                     // Call
       SPZ  m_01
       SPM  m_02
       SPP  m_03
       SPU  m_04
       BEA
m_01:  NOP 0      // Component not loaded
       .
       .
       BEA
m_02:  NOP 0      // Component not compiled
       .
       .
       BEA
m_03:  NOP 0      // Component available as compiled code
       .
       .
       BEA
m_04:  NOP 0      // Invalid parameter in Akku 1
       .
       .
       BEA
```

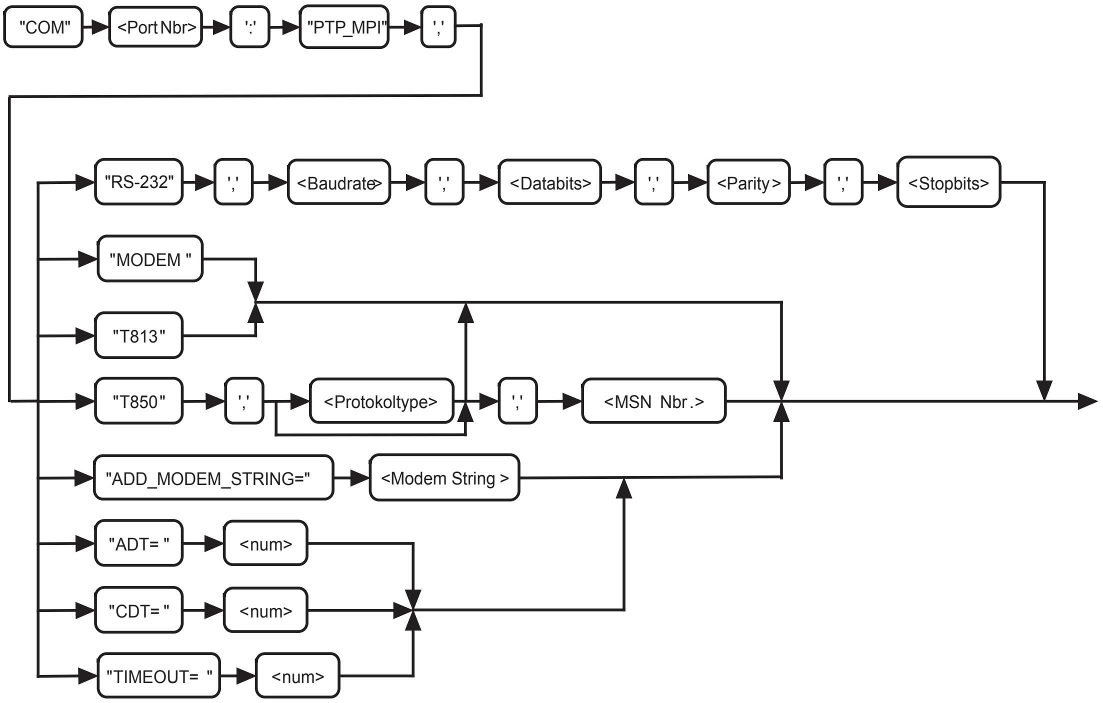
8.4 Initialization of serial programming interface

All serial interfaces for PCD2.xx7 controllers can be initialized and configured as programming interfaces using the Configuration Data Block. Further options are described in manual 26/794 “Serial Communication”.

Please note the following restrictions.

- Only one interface can support the MPI protocol at any given time.
- If no value, or an invalid parameter, is entered, the default value will be used
- The interface configuration is only read after Power On.

The syntax diagram below gives an overview of the possible settings. The sections below explain the individual parameters in more detail.



In the basic configuration, the serial interface COM1 (PCD2.M487 Port 0) is initialized with the following values:

19200 baud, 8 data bits, odd parity, 1 stop bit

This enables the MPI protocol.

The following port numbers are supported:

PCD1.M137:	Port 1 to 3, default Port =1 (Port 1 can only be used for modem)
PCD2.M127/Mx57:	Port 1 to 3, default Port =1
PCD2.M177:	Port 1 to 5, default Port =1
PCD2.M487:	Port 0 to 5, default Port =0
PCD3.Mxxx7:	Port 0 / 1, default Port =0

8.4.1 RS-232 parameters

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

Parameters:

<n>: Interface number (0..5), see table of baud rates
<baud>: The following baud rates are supported:

Saia PCD®	Port number	Baud rates supported
PCD1.M137	Port 1	Can only be used as a modem port.
	Port 2 / 3	300, 600, 1200, 2400, 4800, 9600, 19200, 38400
PCD2.M127/Mx57	Port 1-3	300, 600, 1200, 2400, 4800, 9600, 19200, 38400
PCD2.M177	Port 1-5	300, 600, 1200, 2400, 4800, 9600, 19200, 38400
PCD2.M487	Port 0 / 1	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
	Port 2-5	3300, 600, 1200, 2400, 4800, 9600, 19200, 38400
PCD3.Mxxx7	Port 0 / 1	1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200

Ports 2 and 3 / Ports 4 and 5 support either 19200 or 38400 baud. A CDB entry is used to specify what baud rate the F module should support. See section on Setting the max. baud rate on Slot B1 (B2).

8

<Data>: Number of data bits (7 or 8)
<Parity>: Parity:
E = Even, O = Odd, N = None, L = Force low, H = Force high
<Stop>: Number of stop bits (1 or 2)

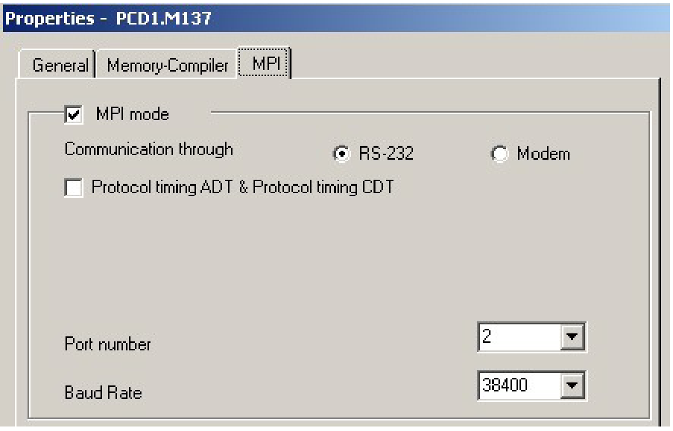
Where the configured serial interface is used as a serial MPI interface, the settings

COM<n>:PTP_MPI,RS-232,19200,8,O,1 or

COM<n>:PTP_MPI,RS-232,38400,8,O,1

must be used, or no connection will be established with the SIMATIC® software.

I/O-Builder example:



Properties - PCD1.M137

☑ MPI mode

Communication through ● RS-232 ○ Modem

□ Protocol timing ADT & Protocol timing CDT

Port number: 2

Baud Rate: 38400

The I/O Builder settings shown above generate the following CDB.

Example setting in the CDB:

Address	Name	I/O	Initial value
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 parameters

Keyword: COM<n>:PTP_MPI,<Modem>

Parameters:

<n>: Interface number (1..5), see table of baud rates

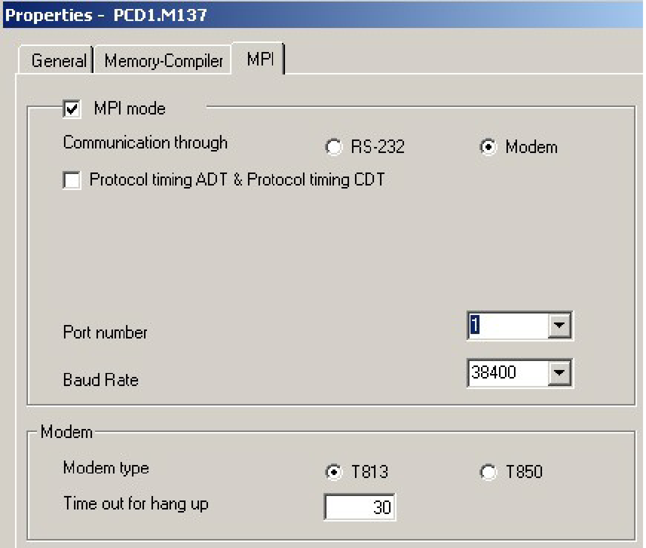
<Modem>: Analog modem type (modem or T813)

This CDB entry generates the following ModemInitString:

“AT&FE0&C1&D3S0=1“

The initialization and configuration of the interface as an analog modem only allow the controller to be called. Outgoing calls are not possible. The default setting is for an existing connection to be canceled after 30 minutes if there is no traffic. This value can be changed via parameter type TIMEOUT.

I/O-Builder example:



Properties - PCD1.M137

☑ MPI mode

Communication through ○ RS-232 ● Modem

☐ Protocol timing ADT & Protocol timing CDT

Port number 1

Baud Rate 38400

Modem

Modem type ● T813 ○ T850

Time out for hang up 30

The I/O Builder settings shown above generate the following CDB.

Example setting in the CDB:

Address	Name	I/O	Initial value
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 parameters

Keyword: COM<n>:PTP_MPI,T850,<protocol>,<MSN>

Parameters:

<n>: Interface number (1..5), see table of baud rates

<Protokol>: Optional. ISDN protocol. The following protocol types are supported:

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

The T850 modem only supports the first 3 protocol types. For all other protocol types, an external ISDN modem must be used.

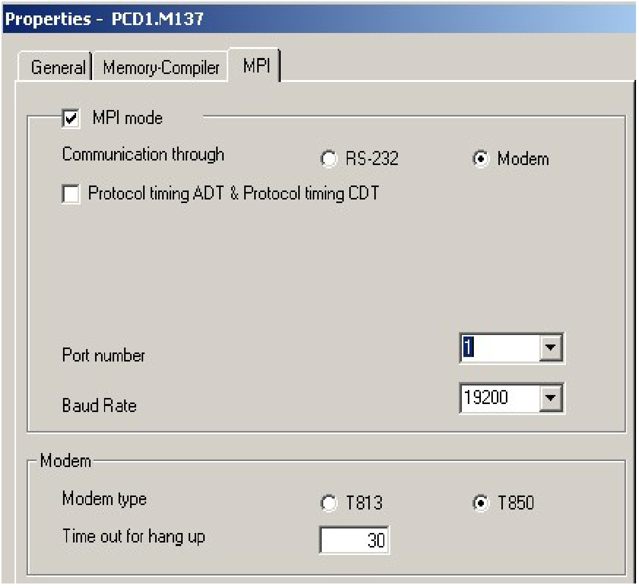
<MSN>: Optional. MSN subscriber number. Default entry = *.
This number may not be longer than 22 digits. If more than 22 digits are entered, this entry will be ignored.

The CDB entry “COM1:PTP_MPI,T850” generates the following ModemInitString:
“AT&FE0S0=1&D2B10#Z=*“

The initialization and configuration of the interface as an ISDN modem only allow the controller to be called. Outgoing calls are not possible. The default setting is for an existing connection to be canceled after 30 minutes if there is no traffic. This value can be changed via parameter type TIMEOUT.

The I/O Builder only allows the CDB entry for the default setting to be selected. If other communication parameters are to be entered for the ISDN modem, the CDB must be amended and updated manually. (See example below).

I/O-Builder example:


Properties - PCD1.M137

☑ MPI mode

Communication through ○ RS-232 ● Modem

□ Protocol timing ADT & Protocol timing CDT

Port number: 1

Baud Rate: 19200

Modem

Modem type ○ T813 ● T850

Time out for hang up: 30

Example CDB entry:

The CDB entry has been changed to use the V.110 protocol and to set the MSN number to 21.

Address	Name	I/O	Initial value
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 Additional modem string

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

Parameters:

<n>:	Interface number (1..5), see table of baud rates
<InitString>:	Additional modem string

The additional modem string is appended to the ModemInitString.

This CDB entry is only read in where a modem has been previously configured.

This CDB entry cannot be made via the I/O Builder.

Example:

Address	Name	I/O	Initial value
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	

The example above generates the following ModemInitString:

“AT&FE0S0=1&D2B0#Z=*AT&FB10“

8.4.5 Timeout parameter

The default setting is for an existing modem connection to be cancelled after 30 minutes if there is no traffic. This value can be changed using parameter type TIMEOUT. The timeout can be disabled with value 0.

The values for character delay time (CDT, default =220 ms / 1 s) and answer delay time (ADT, default = 2000 ms / 10 s) for the communication protocol can also be set. The default values for CDT and ADT with serial programming are 220 ms and 2000 ms. For programming via modem, CDT and ADT are set to 1 second and 10 seconds.

TIMEOUT

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

Parameters:

<n>:	Interface number (1...5), see table of baud rates
<timeout>:	cancel after (0 = no timeout, 1..65535 in mins)

Character delay time

Keyword: COM<n>:PTP_MPI,CDT=<cdt>

Parameters:

<n>:	Interface number (1...5), see table of baud rates
<cdt>:	Character delay time (0..65535 in ms) Default = 220 ms / 1 s (serial/modem)

Both communication parameters should be set to the same value. The setting is converted internally in 10 ms steps.

Answer delay time

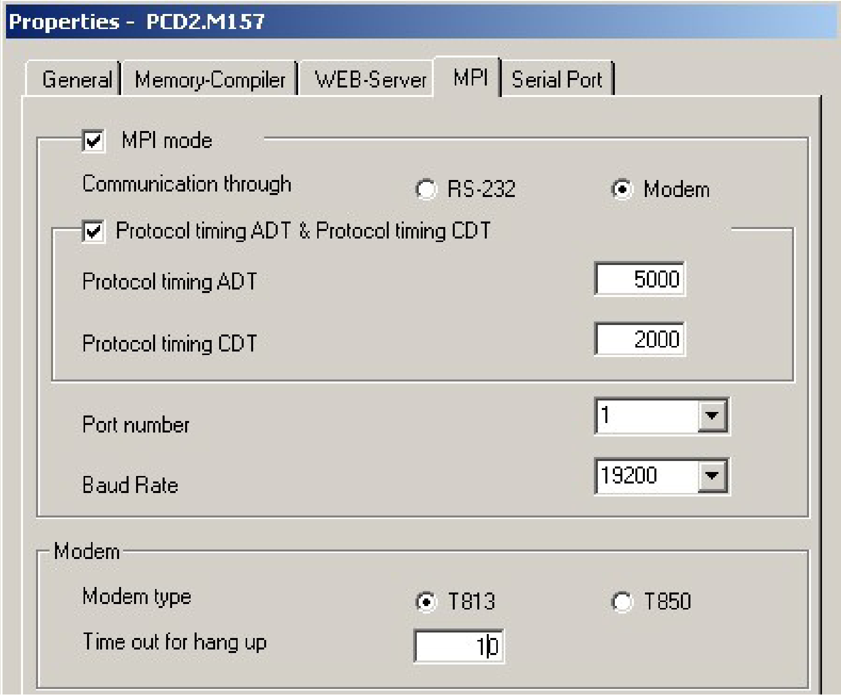
Keyword: COM<n>:PTP_MPI,ADT=<adt>

Parameters:

<n>:	Interface number (1...5), see table of baud rates
<adt>:	Answer delay time (0 ... 65535 in ms) Default = 2000 ms / 10 s (serial/modem)

Both communication parameters should be set to the same value. The setting is converted internally in 10 ms steps.

I/O-Builder example:



Properties - PCD2.M157

☑ MPI mode

Communication through ○ RS-232 ● Modem

☑ Protocol timing ADT & Protocol timing CDT

Protocol timing ADT 5000

Protocol timing CDT 2000

Port number 1

Baud Rate 19200

Modem

Modem type ● T813 ○ T850

Time out for hang up 10

The I/O Builder settings shown above generate the following CDB.

Example setting in the CDB:

Address	Name	I/O	Initial value
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 Setting the max. baud rate on Slot B1 (B2)

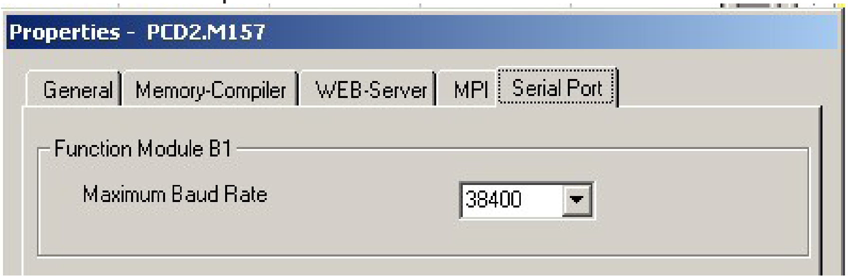
The serial interfaces on the F5xx modules support either 19200 or 38400 baud. This setting applies to both interfaces on the module.

Keyword: SLOT_B<n>:ENABLE_38400

Parameters:
<n>: Slot number (1...2)

This CDB entry is ignored on the PCD1.M137.

I/O-Builder example:


Properties - PCD2.M157

Function Module B1

Maximum Baud Rate 38400

The I/O Builder settings shown above generate the following CDB.

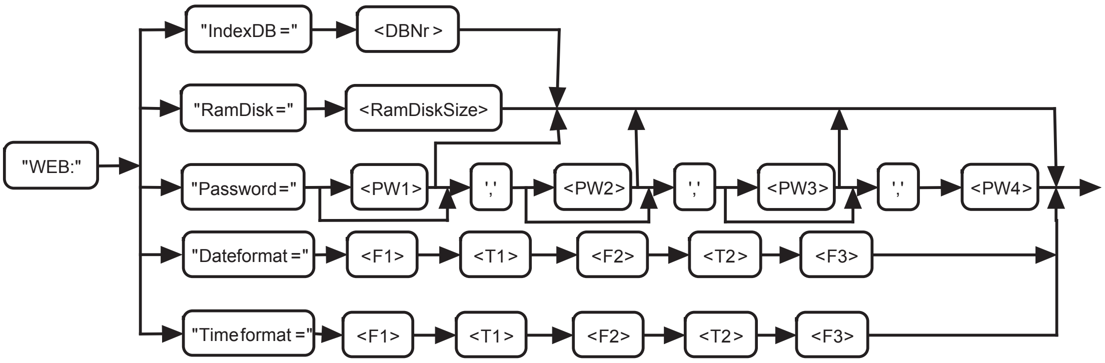
Example setting in the CDB:

Address	Name	I/O	Initial value
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

The CDB can be used to initialize and configure the PCD2.M157, PCD2.M177, PCD2.M487 and PCD3.Mxxx7 controllers to use the web server. The settings for the serial interface were described in the previous section. There is more on the web server in manual 26/775 “Web Server”.

The syntax diagram below gives an overview of the configuration options for the web server. The sections below explain the individual parameters in more detail.



The web server configuration is evaluated after every Stop Run transition. Exception: the RamDisk parameter requires a “Power On” before it is evaluated.

Parameter IndexDB=:
<DBNr> = Number of IndexDB: this data block contains the list of all files loaded as data blocks (DB No + n) into the xx7 controller.

Parameter RamDisk=:
<RamdiskSize> = Size of internal RAM disk. The default setting is 2 kB. This can be increased if required. On the PCD2.M157 and PCD2.M177, the additional RAM disk requirement is taken from Step®7 memory. On the PCD2.M487, the additional memory requirement for the RAM disk is taken from the system memory. See also Memory / Flash functions.

! The RamDisk parameter is only evaluated after a Power On.

i On PCD3.Mxxx7 systems, special Flash devices can be used to store web projects.

Parameter Password=:
<PWx> = Specification of up to 4 passwords (4-level) separated by “,“. (, , = placeholder). With the passwords, there is no distinction between upper and lowercase, e.g. “SBC” and “sbc” are the same password. Each password can be up to 16 characters long, excluding commas and spaces. It is not necessary to define a password.

Parameter Dateformat=:
By default, the web server displays the date from the Step7 DATA_AND_TIME (DT) variable as follows: DD.MM.YYYY (e.g. 10.09.2001). This format can be changed in the CDB.

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

F1,F2,F3 D -> Display day without leading zero
DD -> Display day with leading zero
M -> Display month without leading zero
MM -> Display month with leading zero
YY -> Display year
YYYY -> Display year, 4-digit

T1,T2: Valid separator.

A valid separator must fall into the following range: decimal values of ASCII characters between 33 and 47 or between 58 and 64.

In the event of incorrect entries, the default format will be used.

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

Parameter Dateformat=:
By default, the web server displays the time from the Step7 DATA_AND_TIME (DT) variable as follows: HH:MM:SS (e.g. 11:55:00). This format can be changed in the CDB.

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

F1,F2,F3 H -> Display hours without leading zero
HH -> Display hours with leading zero
M -> Display minutes without leading zero
MM -> Display minutes with leading zero
S -> Display seconds without leading zero
SS -> Display seconds with leading zero

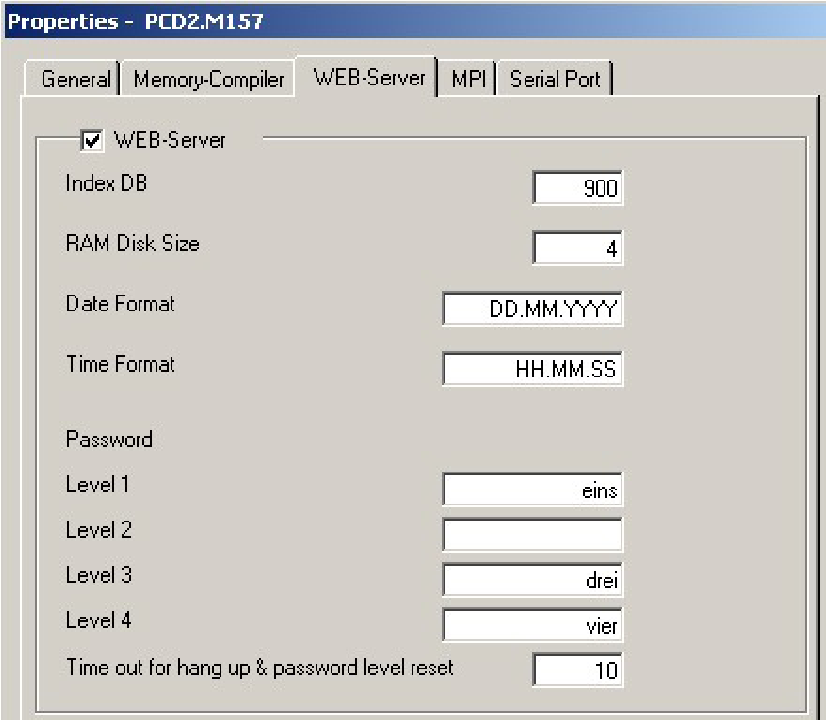
T1,T2 Valid separator.

A valid separator must fall into the following range: decimal values of ASCII characters between 33 and 47 or between 58 and 64.

In the event of incorrect entries, the default format will be used.

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

I/O-Builder example:



☑ WEB-Server

Index DB	900
RAM Disk Size	4
Date Format	DD.MM.YYYY
Time Format	HH.MM.SS
Password	
Level 1	eins
Level 2	
Level 3	drei
Level 4	vier
Time out for hang up & password level reset	10

The I/O Builder settings shown above generate the following CDB.

Example setting in the CDB:

Address	Name	I/O	Initial value
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=one,,three,four´
=132.0		END_STRUCT	

8

8.6 Peripheral access in OB100

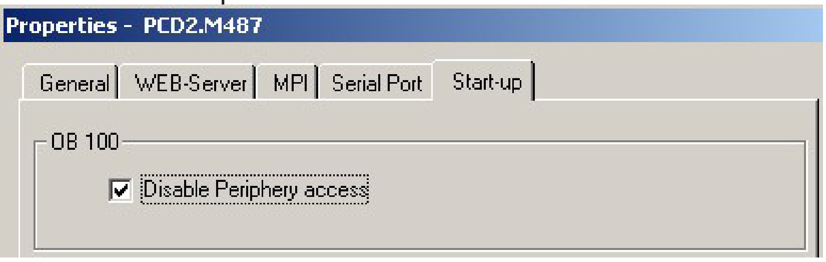
On the PCD1 and PCD2.M1x7 systems, the I/O Reset signal is set during start-up. This means that OB100 cannot be used to access peripheral modules. On the PCD2.M487 and the PCD3.Mxxx7, this signal is not set during start-up. Using the CDB entry Peripherie = Disabled, the user can set the M487 and the PCD3 to the same start-up behavior as the older systems.

Keyword: OB100:Peripherie=[Disabled | Enabled]

This CDB entry is evaluated under the following conditions:

- on every Stop - Run transition
- on the PCD2.M487 and PCD3.Mxxx7 only

I/O-Builder example:



Properties - PCD2.M487

OB 100

☑ Disable Periphery access

8

The I/O Builder settings shown above generate the following CDB.

Example setting in the CDB:

Address	Name	I/O	Initial value
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 Configuration of Profi-S-IO-Master

The PCD2.M487 supports various DP-Master interfaces. The following options are available:

- F750 module on Slot 1
- F750 module on Slot 2
- DP-Master communication on the S-Net/MPI port (Profi-S-IO-Master)

The Profi-S-IO-Master can only be active when the Profi-S-IO:Enable parameter is entered in the CDB. The table below shows all supported combinations of DB configuration.

No	Profi-S-IO flag	DP-Slave	Profi-S-IO-Master	F750 on B1	F750 on B2	Comments
1	ENABLE	-	< V2.0 Int or CP	-	-	Where no F750 module present.
2	DISABLE	< V2.0 Int	-	-	-	
3	ENABLE	-	< V2.0 Int	< V2.0 CP	-	Where only one F750 module plugged in.
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	Where 2 F750 modules are plugged in

[1] Where two F750 modules are plugged in and the Profi-S-IO flag in the CDB has been set to Enable, the F750 module in Slot 2 will not be used. This is the combination shown in row 3 of the table above.

Further information can be found in the manuals “Profibus DP master and slave module documentation” and “Preliminary version of the documentation regarding FDL Master-Master communication”.

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

This CDB entry is evaluated under the following conditions:

- on every Stop - Run transition
- on the PCD2.M487 and PCD3.M5xx7 only

Example setting in the CDB:

Address	Name	I/O	Initial value
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 MPI configuration on Port 2

On the PCD3.M6347, the MPI interface has been replaced by the CAN interface. If you still need to provide an MPI interface on this PCD3 platform, this can be done with a CDB entry. Instead of the RS-485 interface on Port 2, this Port can be switched to MPI.

Further information can be found in the manuals “Profibus DP master and slave module documentation” and “Preliminary version of the documentation regarding FDL Master-Master communication”.

Keyword: MPI:PORT2_ON

This CDB entry is evaluated under the following conditions:

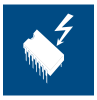
- on every Power on transition
- on the PCD3.M6347 only

Example setting in the CDB:

Address	Name	I/O	Initial value
0.0		STRUCT	
+0.0	Identificator	STRING[12]	‘SBC xx7 CDB’
+14.0	COM	STRING[17]	´MPI:PORT2_ON´
=34.0		END_STRUCT	

A Annex

A.1 Icons

	In manuals, this symbol refers the reader to further information in this manual or other manuals or technical information documents. As a rule there is no direct link to such documents.
	This symbol warns the reader of the risk to components from electrostatic discharges caused by touch. **Recommendation:** Before coming into contact with electrical components, you should at least touch the Minus of the system (cabinet of PGU connector). It is better to use a grounding wrist strap with its cable permanently attached to the Minus of the system.
	This sign accompanies instructions that must always be followed.
	Explanations beside this sign are valid only for the Saia PCD® Classic series.
	Explanations beside this sign are valid only for the Saia PCD® xx7 series.

A.2 Contact

Saia-Burgess Controls AG
Bahnhofstrasse 18
3280 Murten
Switzerland

Phone .... +41 26 672 72 72
Fax.. +41 26 672 74 99

Email support: .... support@saia-pcd.com
Supportsite: .... www.sbc-support.com
SBC site: .... www.saia-pcd.com
International Represetatives &
SBC Sales Companies: .... www.saia-pcd.com/contact

Postal address for returns from customers of the Swiss Sales office

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