Serial Communication & Protocol

PCD xx7 Serie

ASCII

DK3964 (R)

RK512 (R)

Transparent mode Driver

Contents

1 Product description

1.1 Introduction

The communication protocols implemented (ASCII, DK3964, RK512 and Transparent) allow you to exchange data in point-to-point mode between PCDs, Terminals and computer. There is also the possibility to use the RK512 in multi-point mode, but this is only between PCD xx7 and with the interface RS485. The ASCII Driver allows you to create your own communication protocol quite easily.

Use

Different protocol can be use simultaneously in different port and also with different sub-module interface. All the interface are supported: RS232, RS422, RS485 and 20 mA loop .

1.2 PCD implementation

Those protocols are standard features. They are available on all the PCD xx7 series, from the firmware version as indicated :

	Point-to-Point	Multi-Point
PCD1.M137	V1.310	V1.400
PCD2.M1x7	V1.300	V1.316
PCD2.M487	from beginning	from beginning

1.3 Operating

In all the different Protocols and Driver, operations are made with the same System Function. Of course the calling parameters of the functions will have different meanings depending the protocol or Driver you are using.

The functions available are the following.

Function	**Protocol**	**Description**
SFC245	ASCII, DK3964, RK512, Transparent.	Select the desired Protocol, configure and initialize the serial communication port.
SFB12	ASCII, DK3964, RK512, Transparent.	Send data.
SFB13	ASCII, DK3964, Transparent.	Receive data.
SFB14	RK512(R)	Get Data.

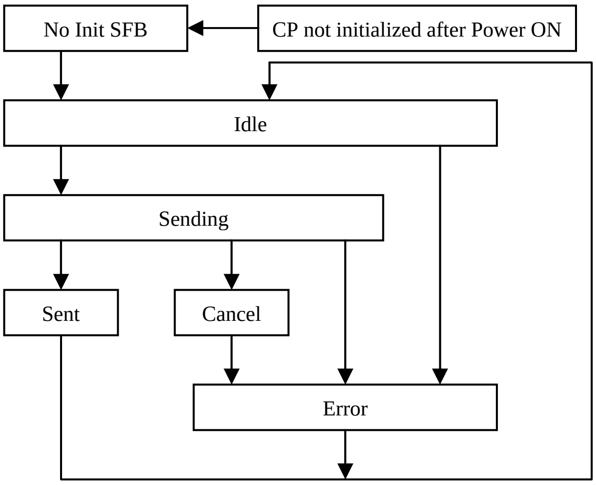
All those functions will be clearly described for each protocol in their respective chapter.

1.4 Function State machine

All the functions listed in the chapter 1.3, except the SFC245, are state machines. It means that the execution of the function is done in few steps, this because each protocol has some rules and the whole execution of the function can't be done in one cycle.

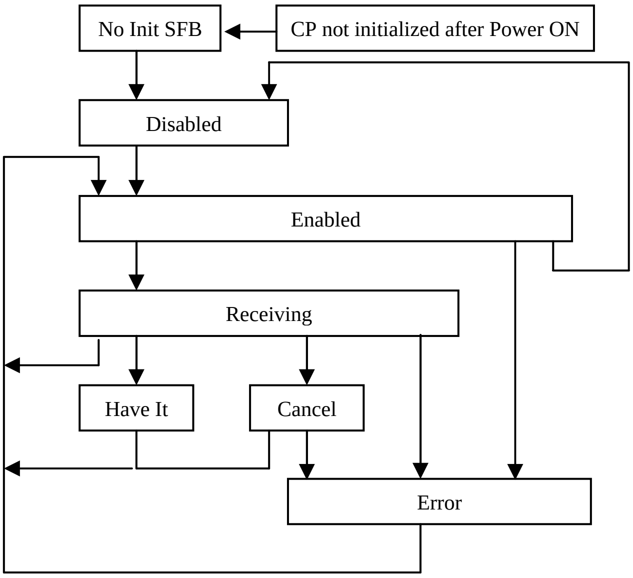
The status of the function can change depending on events. There are two sources of events. One is the S7 program calling the SFB to give commands, the other one is internal to the function. Here following are described the events which change the state of the function for all of them.

1.4.1 Possible states of SFB12



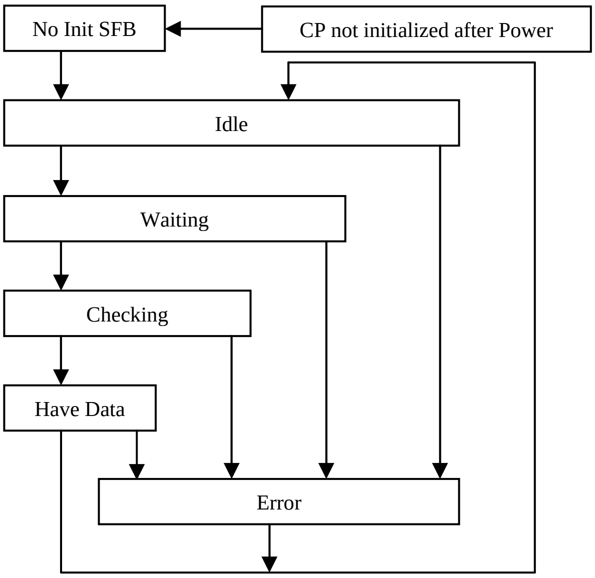
From Status	To Status	Event
No CP	No Init	Calling the SFC245 will enable the CP functionality and always put the SFB in No Init
No Init	Idle	By calling the SFB12 the first time
Idle	Sending	Positive edge on the input REQ of the SFB12
Idle	Error	Problem during communication
Sending	Sent	Operation has been done
	Cancel	Positive edge on the input R or the specified S7-object doesn't exist.
	Error	Problem during communication
Sent	Idle	Next call of the SFB12
Error	Idle	Next call of the SFB12

1.4.2 Possible states of SFB13



From Status	To Status	Event
No CP	No Init	Calling the SFC245 will enable the CP functionality and always put the SFB in No Init
No Init	Disabled	By calling the SFB13 the first time
Disabled	Enabled	Calling the SFB13 with EN_R = 1
Enabled	Disabled	Calling the SFB13 with EN_R = 0
Enabled	Receiving	A telegram is incoming
Receiving	Have It	The whole telegram arrived
Have It	Enabled	Next call of the SFB13
Cancel		
Error		
Receiving	Enabled	Calling the SFB13 with EN_R = 0
Receiving	Cancel	The specified S7-object doesn't exist
Receiving	Error	Error with the receiving Buffer (too small)

1.4.3 Possible states of SFB14



From Status	To Status	Event
No CP	No Init	Calling the SFC245 will enable the CP functionality and always put the SFB in No Init
No Init	Idle	By calling the SFB14 the first time
Idle	Waiting	Positive edge on the input REQ of the SFB12
Waiting	Checking	Internal event
Checking	Have Data	Data received are correct
Have Data	Idle	Next Call of the SFB14
Idle	Error	Problem of communication or the specified S7-object doesn’t exist.
Waiting		
Checking		
Have Data		
Error	Idle	Next Call of the SFB14

2 The serial communications interfaces (F-Modules)

Chapter 8 of the **manual 26/757** describes in detail:

- Slots (Space) A, B, B1 and B2
- Serial interface modules
- Terminals and connections
- Cables and wiring

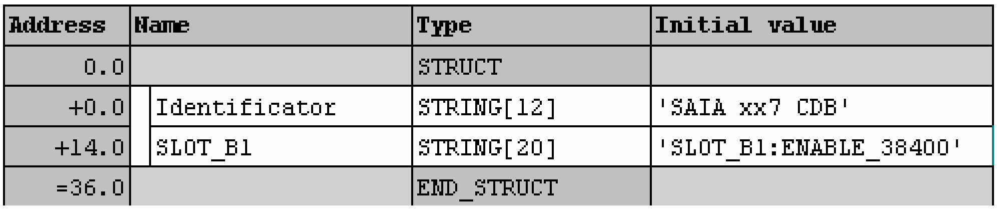
2.1 Supported interfaces and baud rates

The following table shows the supported interfaces and baudrates.

	Slot	Port	Baud rate
PCD1.M137	A	1	300,600,1200,2400,4800,9600,19200,38400
	B	2	300,600,1200,2400,4800,9600,19200,38400
	B	3	300,600,1200,2400,4800,9600,19200,38400
PCD2.M127/ 157	A	1	300,600,1200,2400,4800,9600,19200,38400
	B	2	300,600,1200,2400,4800,9600,19200[1],38400[1]
	B	3	300,600,1200,2400,4800,9600,19200[1],38400[1]
PCD2.M177	A	1	300,600,1200,2400,4800,9600,19200,38400
	B1	2	300,600,1200,2400,4800,9600,19200[1],38400[1]
	B1	3	300,600,1200,2400,4800,9600,19200[1],38400[1]
	B2	4	300,600,1200,2400,4800,9600,19200[1],38400[1]
	B2	5	300,600,1200,2400,4800,9600,19200[1],38400[1]
PCD2.M487	PGU	0	1200,2400,4800,9600,19200,38400,57600,115200
	A	1	1200,2400,4800,9600,19200,38400,57600,115200
	B1	2	1200,2400,4800,9600,19200[1],38400[1]
	B1	3	1200,2400,4800,9600,19200[1],38400[1]
	B2	4	1200,2400,4800,9600,19200[1],38400[1]
	B2	5	1200,2400,4800,9600,19200[1],38400[1]
	(27-29)	6	1200,2400,4800,9600,19200,38400,57600,115200

Note:

[1]Tthe maximum baud rate of 19200 or 38400 baud of the interfaces 2…5 can only be changed by entering a string in the configuration data block (CDB). The following example sets the maximum baud rate for slot B or B1 (port 2 and 3) to 38400 baud.



Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	Identificator	STRING[12]	'SAIA xx7 CDB'
+14.0	SLOT_B1	STRING[20]	'SLOT_B1:ENABLE_38400'
=36.0		END_STRUCT	

To create a configuration data block and to enter the string into it one needs the I/O-Builder, which can be downloaded under www.sbc-support.ch .

Restriction:

- The default setting for interfaces 2…5 is 19200 baud
- When one setting is enabled the other one is disabled. This means when 38400 baud is set, 19200 baud will not be supported and vice versa.

3 ASCII

Introduction

The ASCII communication driver is not really handling a protocol, but it’s more a communication frame control. There are also 4 different ways to control a frame:

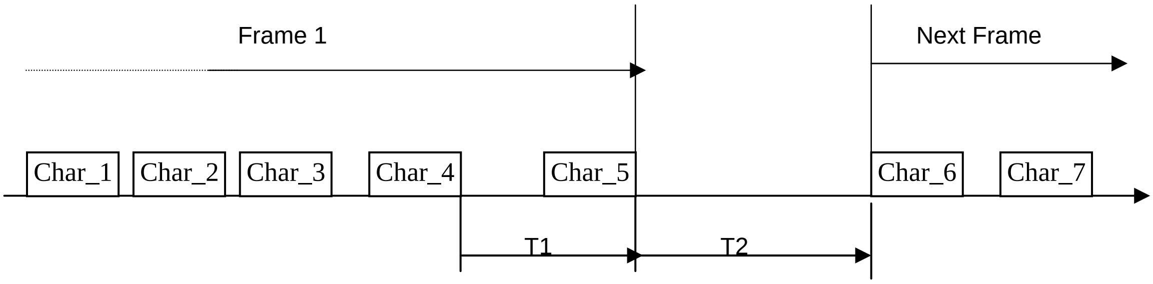
- Time out
- Fixe length
- One end character
- Two end characters

When the frame end is identified, the system will notify the user that a complete frame was received and that the frame was transferred to a memory area. It’s actually not possible to have more than one frame in the receiving buffer. If a frame is in the receiving buffer, all the character coming after will be lost, until the frame waiting in the receiving buffer is transfer to some memory area.

We will see now in details the 4 different ASCII mode.

3.1 Time out

The time out mode, is a way to find the end of a communication Frame. If after a predefined time (Time out,ZVZ) no character arrived The receiving interface will understand a end of a frame. Characters coming after will belong to another frame and will not be considered.



ASCII Figure 1

T1 is smaller than the time predefine for **Time Out (ZVZ),** so the char_5 is on the frame, but the time T2 is higher than **Time Out (ZVZ)**, the Char_5 is then the last char of the frame. Char_6 will belong to the next Frame.

3.2 Fixed length

This mode is very simple. Each frame has a fixed number of characters. When the receiving interface has received the number of character predefined for it, the frame is finished. The following character will belong to the next frame.
But you still have to respect a maximum time between characters. If the receiving time between two character is longer than the time predefined **(ZVZ)**, the system will return you an error message.

3.3 One end character

The end of the frame will be identified by a predefined character. Every time this predefined character will be encountered by the receiver it will identify the end of the frame. But you still have to respect a maximum time between characters. If the receiving time between two characters is longer than the time predefined **(ZVZ)**, the system will return you an error message.

3.4 Two end Characters

Like for one end character (3.3), but in this case two following predefine character are needed to identify the end of the frame.

3.5 Initialize the PCD.xx7 in the ASCII mode

To use one of those ASCII modes you need to initialize and configure the COM port of the PCD. This is done with the SFC245. To set a mode, you need to execute the SFC245 only one time.
This SFC is also use to configure the COM port for the other communication protocol as RK512 and DK 3964. Then some parameters are not needed for the ASCII Driver. The column "Selected Mode" will indicate you, which are the parameters needed for the ASCII Driver and also depending which mode you are using in the ASCII Driver.

Name	Para	Type	Possible Values	Selected Mode				Remark
				5	6	7	8	
Port	IN0	INT	0…6	■	■	■	■	COM port number (see chapter 2.1)
Mode	IN1	INT	5…8	■	■	■	■	5 ASCII – Fixed Length 6 ASCII – 1 End char 7 ASCII – 2 End char 8 ASCII – Time Out
Baud Rate	IN2	DINT		■	■	■	■	Baud rate (see chapter 2.1)
Data Bit	IN3	INT	7…8	■	■	■	■	Number of Data bit
Stop Bit	IN4	INT	1…2	■	■	■	■	Number of stop bit
Parity	IN5	INT	0…4	■	■	■	■	0 None 1 Even 2 Odd 3 Force Low 4 Force High
Control	IN6	INT	0…3	■	■	■	■	Interface type 0 RS 232 1 RS485 2 RS422 3 TTY
XON	IN7	BYTE	0…FFh	░	░	░	░	Not use
XOFF	IN8	BYTE	0…FFh	░	░	░	░	Not use
WaitSend	IN9	WORD	0…FFFEh	░	░	░	░	Not use
WaitInactiv	IN10	WORD	0…FFFEh	░	░	░	░	Not use
TelCount	IN11	INT	1	░	░	░	░	Number of Frame in the buffer
Overwrite	IN12	BOOL	FALSE / TRUE	░	░	░	░	FALSE: Can't overwrite frame TRUE: can overwrite frame in the buffer, but only if TelCount = 1.
DelRxPuffer	IN13	BOOL	FALSE / TRUE	░	░	░	░	Not use
DKPriority	IN14	BOOL	FALSE / TRUE	░	░	░	░	Not use
ZVZ	IN15	WORD	0, 1…FFFEh				■	Time out set in ms, set to 0 for the default value of 4 ms.
QVZ	IN16	WORD	0, 1…FFFEh	░	░	░	░	Not use
TryToConnect	IN17	INT	0…255	░	░	░	░	Not use
TryToSend	IN18	INT	0…255	░	░	░	░	Not use
FixedLen	IN19	INT	1…1024	■				Frame length in receiving
EndChar1	IN20	BYTE	0…255		■			End character 1
EndChar2	IN21	BYTE	0…255		■	■		End character 2
SENDBuffer	IN22	INT	0…4000	■	■	■	■	Send buffer size, depending on the frame length you need to send. (in bytes)
RCVBuffer	IN23	INT	0…4000	■	■	■	■	Receive buffer size, depending on the frame length you need to receive. (in bytes)
Dummy_I0	IN24	INT	0	░	░	░	░	Not use
Dummy_W1	IN25	WORD	0	░	░	░	░	Not use
Dummy_W2	IN26	WORD	0	░	░	░	░	Not use
Dummy_DW1	IN27	DWORD	0	░	░	░	░	Not use
RetVal	OUT	WORD		■	■	■	■	Result of the operation, see chapter 3.5.2

░ Not needed ■ Required

3.5.1 Example:

In this example we configure the COM port1 of the PCD.xx7 to use the ASCII Driver in the Time Out mode. Baud rate 9600, 1 stop bit, no parity. The interface is a RS232. The Time Out (ZVZ) is fixed at 20 ms. The size of the receiving and sending buffer is 300 bytes.

```
CALL  SFC  245
       IN0    :=1              // Serial Port N°1
       IN1    :=8              // Time out mode
       IN2    :=L#9600         // Baud Rate
       IN3    :=8              // Data Bit
       IN4    :=1              // Stop Bit
       IN5    :=0              // Parity (None)
       IN6    :=0              // RS232 interface
       IN7    :=B#16#0         // Not use
       IN8    :=B#16#0         // Not use
       IN9    :=W#16#0         // Not use
       IN10   :=W#16#0         // Not use
       IN11   :=1              // TelCount
       IN12   :=FALSE          // Can’t Overwrite Frame
       IN13   :=FALSE          // Not use
       IN14   :=FALSE          // Not use
       IN15   :=W#16#14        // ZVZ (Time Out) = 20 ms
       IN16   :=W#16#0         // Not use
       IN17   :=0              // Not use
       IN18   :=0              // Not use
       IN19   :=0              // Not use here (Frame length)
       IN20   :=B#16#0         // Not use in this mode (EndChar1)
       IN21   :=B#16#0         // Not use in this mode (EndChar2)
       IN22   :=300            // 300 bytes for the SENDBuffer
       IN23   :=300            // 300 bytes for the RCVBuffer
       IN24   :=0              // Not use
       IN25   :=W#16#0         // Not use
       IN26   :=W#16#0         // Not use
       IN27   :=DW#16#0        // Not use
       RET_VAL:=#RetVal        // RetVal
```

3.5.2 Return value of the SFC245

Value	Description
0	Initialization was done correctly
-1	Not valid COM port number
-2	Not enough S7 memory to create the Buffer, but will be possible if you compress the S7 memory.
-3	Not enough S7 memory to create the Buffer, even if you compress the compress the S7 memory.
-4	Not valid parameter Mode
-5	Not valid interface parameters (Baud rate, Data bits, Stop bit or Parity)
-6	Not valid value in WaitSend or WaitInactiv parameter.
-7	Not valid value in TelCount Parameter
-8	Not valid value in ZVZ or QVZ Parameter
-9	Not valid value in TryToConnect or TryToSend parameter
-10	ASCII – Fixed length: The length of the frame is bigger than the RCV-buffer size
-11	Not valid value in SENDBuffer or RCVBuffer parameter
-12	The total memory of the RCVBuffer and the SENDBuffer is bigger than the 64k Bytes allowed
-13	The SFC was called with the sum of the parameter RCVBuffer and the SENDBuffer different from the first call of the SFC.

Note:

It’s possible to change the driver or the protocol mode of the serial port during execution time, but some rules have to be respected.

1) The buffers size can’t be changed
2) For the PCD to consider the new mode, the SEND or RECEIVE function needs a rising edge on their Enable input.

3.7 SEND a Frame

To send a Frame of character you will use the SFB12. This SFB12 will work for all the ASCII modes, but some different condition has to be respected for each mode.

3.7.1 Parameter SFB 12

Parameter	Description
REQ	With a positive edge, it will start the SEND procedure.
R	With a positive edge, it will cancel and reset the sending.
ID	Serial COM port number.
R_ID	Not use
DONE	SEND procedure is done, **DONE** stay at the value true for one cycle.
ERROR	Error in the send procedure, Output stay at the value true for one cycle.
STATUS	Error code
SD_1	Area of the data to be SEND. This parameter is an ANY pointer, but the length of the any pointer is not considered here. It will be taken from the LEN parameter.
LEN	4’000 bytes maximum can be sent in one time. Of course the SEND buffer size has to be declared equal or higher.

3.7.2 SEND condition

ASCII Mode	**Condition**
Fixed length (5)	The number of character send is determined by the parameter **LEN**.
One end character (6)	The first character transmitted will be the one pointed by SD_1, all character will be send until the END character is found. The END character has to be in the range defined by **LEN**, otherwise nothing will be send.
Two end character (7)	Idem as the mode 6 but with the two end character.
Time Out (8)	The number of character send is determined by the parameter **LEN**. After the last character has been sent, the PCD will wait the **Time Out** time, before to restart a sending (if an other SEND is requested).

There are also some conditions regarding the state machine of the function, see chapter 1.4.

3.7.3 SEND example

```
CALL  SFB   12 , DB12
       REQ   :=M50.0             // request to send
       R     :=M50.1             // transmit reset
       ID    :=W#16#1            // COM port number
       R_ID  :=                  // not needed
       DONE  :=M100.0            // Transmit is done without error
       ERROR :=M100.1            // Error during transmit
       STATUS:=MW102             // Error code
       SD_1  :=P#DB100.DBX 0.0 BYTE 1000 // Data source
       LEN   :=MW104             // Length of byte to be send
```

3.7.4 Return parameter STATUS

Value	Description
-9 (FFF7h)	Telegram can’t be copied
-8 (FFF8h)	Length is too large
-7 (FFF9h)	Unknown type
-6 (FFFAh)	Invalid destination area
-5 (FFFBh)	Invalid length
-4 (FFFCh)	DB is not loaded
-3 (FFFDh)	Invalid DB number
11	Warning: can’t execute the function, because this one is already being executed.

3.8 RECEIVE a Frame

To read the data incoming on the serial port with the ASCII driver, you will use the SFB 13.

3.8.1 Parameter SFB 13

Parameter	Description
EN_R	True, Enable the receiving
ID	Serial port number
R_ID	Not use
NDR	New Data Ready, new data arrived and were transferred to the desired memory area. **NDR** stays at the value TRUE for one cycle.
ERROR	Error appended in the receiving. **Error** stays at the value TRUE for one cycle.
STATUS	Error code
RD_1	Specify the area were the incoming data have to be put. This parameter is an ANY pointer. The number of byte specified in this parameter has to be equal or higher to the number of byte received.
LEN	Number of byte actually in the receiving buffer

3.8.2 RECEIVE condition

ASCII Mode	**Condition**
Fixed length (5)	Signal NDR will be True when the number of characters received are equal to the number specified in the parameter FixedLen (IN19) of the SFC245.
One end character (6)	Signal NDR will be True when the end character will be detected in the receiving buffer and also the Frame length has to be smaller the number of byte specified in the RD_1 parameter.
Two end character (7)	Idem has the mode 6 but with the two end character.
Time Out (8)	Signal NDR will be True when the time between two characters will exceed the time specified in the parameter ZVZ (IN15) of the SFC245.

There are also some conditions regarding the state machine of the function, see chapter 1.4.

3.8.3 Example

```
CALL  SFB   13 , DB13
      EN_R  :=M200.0       // Enable the receiving function
      ID    :=W#16#1       // COM port 1
      R_ID  :=             // Not use
      NDR   :=M200.1       // New data arrived flag
      ERROR :=M200.2       // Error Flag
      STATUS:=MW202        // Error code
      RD_1  :=P#DB200.DBX 0.0 BYTE 2000    // Data destination area
      LEN   :=MW204        // Number of byte received
```

3.8.4 Parameter Status

Value	**Description**
-7 (FFF9h)	Invalid type
-6 (FFFAh)	Invalid destination area
-5 (FFFBh)	Invalid length
-4 (FFFCh)	DB is not loaded
-3 (FFFDh)	Invalid DB number
-2 (FFFEh)	COM port not initialized
-1 (FFFFh)	Invalid COM port
0	OK
4	Length too long compare to the buffer size

3.9 Examples

An example has been made to show how work with the 4 possible drivers.
You download this example from the web address www.sbc-support.ch The file name is DOC_CP44.zip .

You need to retrieve it in order to use it. This is done from the SIMATIC Manager -> Menu **FILE** -> **Retrieve**.

In the project you will find two PLCs.

PCD_SEND	: This PCD will send ASCII frame
PCD_RECEIVE	: This PCD will receive the ASCII frame.

In order to make this demo work correctly, you need to connect the two COM1 of the PCD together, using a PCD2.F120 module.

One PCD will send ASCII frame in a selected mode and the other will receive them in the same mode.

The ASCII mode is selected in the PCD software on the network 1 of the OB100.

Then in both PCD there are FC for each mode:
FC5 : will run on the fixe length mode
FC6 : will run on the one END character mode
FC7 : will run on the two END character mode
FC8 : will run on the TIME out mode.

All the FC are structured in the same way.

SEND_PCD:
Network 1: load length of the frame to send
Network 2: Call the SFB12 (send)
Network 3: Between each SEND of frame wait 2000 cycle (for the example only)
Network 4: Count the number of sent frames and the number of errors.

RECEIVE_PCD:
Network 1: Call the SFB13 (receive)
Network 2: Count the number of sent frames and the number of errors.

The VAT1: shows you how many frame have been SENT or RECEIVED on the PCDs, or also how many error occurred.

4 DK3964

Introduction

The 3964(R) procedure controls data transmission via point-to-point connection between the PCD xx7 and a communication partner. As well as the physical layer (ISO-layer 1), this procedure also incorporate the data-connection layer (ISO-layer2).

Control Characters

During data transmission, the 3964(R) procedure adds control characters to the information data (data-connection layer). These control characters allow the communication partner to check whether the data has arrived complete and without errors.

The 3964(R) procedure analyzes the following control codes:

• **STX**	Start of text
• **DLE**	Data link escape
• **ETX**	End of text
• **BCC**	Block check character (with 3964 (R) only)
• **NAK**	Negative acknowledge

Priority

With the 3964(R) procedure, one of the two communication partners has a priority higher than the other one. This has to be set during the configuration. Like this if both partner begin connection setup at the same time, the partner with the higher priority will be allowed to continue and the other one will defer its request.

Block Checksum

With the 3964R transmission protocol only, data integrity is increased by the additional sending of a block check character (BCC).
But the block check character of the 3964R procedure (EXOR logic operation) cannot detect missing zeros characters, because the character with a value zero does not affect the result of an EXOR operation.

4.1 Initialize the PCD.xx7 for the 3964(R) protocol

In order to use any serial com port of the PCD xx7 you need to initialize and configure it. This is done with the SFC245. To set the protocol on a serial port you need to execute the SFC245 only one time. This SFC is also use to configure the COM port for the other communication protocol as RK512 and ASCII, then some parameter are not needed for the 3964 protocol. The column “Selected Mode” will indicate you, which are the parameters needed for the 3964 protocol.

Name	Para	Type	Possible Values	Selected Mode		Remark
				1	2	
Port	IN0	INT	0…6	■	■	COM port number (see chapter 2.1)
Mode	IN1	INT	1…2	■	■	1 DK 3964 2 DK 3964 R
Baud Rate	IN2	DINT		■	■	Baud rate (see chapter 2.1)
Data Bit	IN3	INT	7…8	■	■	Number of Data bit
Stop Bit	IN4	INT	1…2	■	■	Number of stop bit
Parity	IN5	INT	0…4	■	■	0 None 1 Even 2 Odd 3 Force Low 4 Force High
Control	IN6	INT	0…3	■	■	Interface type 0 RS 232 1 RS485 2 RS422 3 TTY
XON	IN7	BYTE	0…FFh	░	░	Not use
XOFF	IN8	BYTE	0…FFh	░	░	Not use
WaitSend	IN9	WORD	0…FFFEh	░	░	Not use
WaitInactiv	IN10	WORD	0…FFFEh	░	░	Not use
TelCount	IN11	INT	1	░	░	Number of Frame in the buffer
Overwrite	IN12	BOOL	FALSE / TRUE	░	░	FALSE: Can’t overwrite frame TRUE: can overwrite frame in the buffer, but only if TelCount = 1.
DelRxPuffer	IN13	BOOL	FALSE / TRUE	░	░	Not use
DKPriority	IN14	BOOL	FALSE / TRUE	■	■	True for this PCD to be the high priority partner
ZVZ	IN15	WORD	0, 1…FFFEh	■	■	Time out between character, set by step of 10 ms. 0 for the default value of 220 ms.
QVZ	IN16	WORD	0, 1…FFFEh	■	■	Acknowledge Time out, time for waiting an answer, set by step of 10 ms. “0 “for the default value. Default value is 550 ms for the 3964 and 2000 ms for the 3964 R .
TryToConnect	IN17	INT	0…255	■	■	Number of time it will retry to connect to the partner. “0” for the default value of 6 .
TryToSend	IN18	INT	0…255	■	■	Number of retries to send the data. “0” for the default value of 6
FixedLen	IN19	INT	1…1024	░	░	Frame length in receiving
EndChar1	IN20	BYTE	0…255	░	░	End character 1
EndChar2	IN21	BYTE	0…255	░	░	End character 2
SENDBuffer	IN22	INT	0…4000	■	■	Send buffer size (in bytes)
RCVBuffer	IN23	INT	0…4000	■	■	Receive buffer size (in bytes)
Dummy_I0	IN24	INT	0	░	░	Not use
Dummy_W1	IN25	WORD	0	░	░	Not use
Dummy_W2	IN26	WORD	0	░	░	Not use
Dummy_DW1	IN27	DWORD	0	░	░	Not use
RetVal	OUT	WORD		■	■	Result of the operation, see chapter 4.1.2

░ Not needed ■ Required

4.1.1 Example

In this example we configure the COM port1 of the PCD.xx7 to use the DK3964 Protocol. Baud rate 9600, 1 stop bit, no parity. The interface is a RS232. The Time Out between characters (ZVZ) is fixed at 200 ms, Time out for the acknowledge (QVZ) is the default one of 550 ms. The size of the receiving and sending buffer is 4000 bytes. This station has the high priority. The number of try to connect and try to send is set to the default value of 6.

```
CALL  SFC  245
       IN0    :=1           // Serial Port N°1
       IN1    :=1           // DK3964
       IN2    :=L#9600      // Baud Rate
       IN3    :=8           // Data Bit
       IN4    :=1           // Stop Bit
       IN5    :=0           // Parity (None)
       IN6    :=0           // RS232 interface
       IN7    :=B#16#0      // Not use
       IN8    :=B#16#0      // Not use
       IN9    :=W#16#0      // Not use
       IN10   :=W#16#0      // Not use
       IN11   :=1           // TelCount
       IN12   :=FALSE       // Can't Overwrite Frame
       IN13   :=FALSE       // Not use
       IN14   :=TRUE        // DK priority is high for this PCD
       IN15   :=W#16#14     // ZVZ (between character) = 200 ms
       IN16   :=W#16#0      // QVZ (for Acknowledge)default value
       IN17   :=0           // Try to Connect, default value is 6
       IN18   :=0           // Try to send, default value is 6
       IN19   :=0           // Not use
       IN20   :=B#16#0      // Not use
       IN21   :=B#16#0      // Not use
       IN22   :=4000        // 4000 bytes for the SENDBuffer
       IN23   :=4000        // 4000 bytes for the RCVBuffer
       IN24   :=0           // Not use
       IN25   :=W#16#0      // Not use
       IN26   :=W#16#0      // Not use
       IN27   :=DW#16#0     // Not use
       RET_VAL:= 0          // Return value
```

4.1.2 Return value of the SFC245

Value	Description
0	Initialization was done correctly
-1	Not valid COM port number
-2	Not enough S7 memory to create the Buffer, but will be possible if you compress the S7 memory.
-3	Not enough S7 memory to create the Buffer, even if you compress the compress the S7 memory.
-4	Not valid parameter Mode
-5	Not valid interface parameters (Baud rate, Data bits, Stop bit or Parity)
-6	Not valid value in WaitSend or WaitInactiv parameter.
-7	Not valid value in TelCount Parameter
-8	Not valid value in ZVZ or QVZ Parameter
-9	Not valid value in TryToConnect or TryToSend parameter
-10	ASCII – Fixed length: The length of the frame is bigger than the RCV-buffer size
-11	Not valid value in SENDBuffer or RCVBuffer parameter
-12	The total memory of the RCVBuffer and the SENDBuffer is bigger than the 64k Bytes allowed
-13	The SFC was called with the sum of the parameter RCVBuffer and the SENDBuffer different from the first call of the SFC.

<u>Note:</u>

It’s possible to change the communication protocol during execution time, but some rules have to be respected.

1) The buffers size can’t be changed
2) For the PCD to consider the new mode, the SEND or RECEIVE function needs a rising edge on their Enable input.

4.2 Sending data with DK3964(R)

Transmit data via DK3964(R) is done this way:
You can send the data with the system function block BSEND(SFB12) and receive the data at the communication partner with the system function block BRCV(SFB13) .This type of data transmission has the advantage of knowing when all the data have been received, this by checking the NDR (on the receiver) and the DONE (on the sender).

4.2.1 Parameter SFB 12

Parameter	Description
REQ	With a positive edge, it will start the SEND procedure.
R	With a positive edge, it will cancel and reset the sending.
ID	Serial COM port number.
R_ID	Not use
DONE	SEND procedure is done, **DONE** stay at the value true for one cycle.
ERROR	Error in the send procedure, Output stay at the value true for one cycle.
STATUS	Error code
SD_1	Area source of the data to be SEND, this parameter is an ANY pointer, but the length of the any pointer is not considered here, it will be taken from the **LEN** parameter.
LEN	4 KB maximum can be sent in one time, of course the SEND buffer size has to be declared equal or higher.

There are also some conditions regarding the state machine of the function, see chapter 1.4 for details.

4.2.2 SEND example

```
CALL  SFB   12 , DB12
       REQ   :=M50.0            // request to send
       R     :=M50.1            // transmit reset
       ID    :=W#16#1           // COM port number
       R_ID  :=                 // not needed
       DONE  :=M100.0           // Transmit is done without error
       ERROR :=M100.1           // Error during transmit
       STATUS:=MW102            // Error code
       SD_1  :=P#DB100.DBX 0.0 BYTE 1000 // Data source
       LEN   :=MW104            // Length of byte to be send
```

4.2.3 Return parameter STATUS

Value	Description
-9 (FFF7h)	Telegram can't be copied
-8 (FFF8h)	Length is too large
-7 (FFF9h)	Unknown type
-6 (FFFAh)	Invalid destination area
-5 (FFFBh)	Invalid length
-4 (FFFCh)	DB is not loaded
-3 (FFFDh)	Invalid DB number
1	Communication problems (check the transmission line)
5	Reset request received
11	Warning: can't execute the function, because this one is already being executed.

<u>Note:</u>

In the standard DK3964 protocol is a second possibility to send data to a communication partner, which is equipped with a Receiving mailbox.
On the PCD.xx7 this function is not implemented on the 3964(R) protocol, but you can get it using the RK512 protocol which is implemented (see chapter 5).

4.3 Receiving data with DK 3964(R)

Receiving data with the DK3964(R) is done through the SFB13. The COM port of the PCD has to be initialized and configured first with the SFC245.
Receiving data is handled by the PCD as soon as the SFB13 is executed. Then the user can control when new data arrived with the **NDR** parameter.

4.3.1 Parameter SFB 13

Parameter	Description
EN_R	True, Enable the receiving
ID	Serial port number
R_ID	Not use
NDR	New Data Ready, new data arrived and were transferred to the desired memory area. **NDR** stays at the value TRUE for one cycle.
ERROR	Error appended in the receiving. **Error** stays at the value TRUE for one cycle.
STATUS	Error code
RD_1	Specify the area were the incoming data have to be put. This parameter is an ANY pointer. The number of byte specified in this parameter has to be equal or higher to the number of byte received.
LEN	Number of byte actually in the receiving buffer

There are also some conditions regarding the state machine of the function, see chapter 1.4 for details.

4.3.2 Example

```
CALL  SFB   13 , DB13
      EN_R  :=M200.0       // Enable the receiving function
      ID    :=W#16#1       // COM port 1
      R_ID  :=             // Not use
      NDR   :=M200.1       // New data arrived flag
      ERROR :=M200.2       // Error Flag
      STATUS:=MW202        // Error code
      RD_1  :=P#DB200.DBX 0.0 BYTE 2000    // Data destination area
      LEN   :=MW204        // Number of byte received
```

4.3.3 Parameter Status

Value	**Description**
-7 (FFF9h)	Invalid type
-6 (FFFAh)	Invalid destination area
-5 (FFFBh)	Invalid length
-4 (FFFCh)	DB is not loaded
-3 (FFFDh)	Invalid DB number
-2 (FFFEh)	COM port not initialized
-1 (FFFFh)	Invalid COM port
0	OK
1	Communication problems (check the transmission line)
4	Length too long compare to the buffer size

4.4 Examples

An example has been made to show how work the DK3964 protocol.
You download this example from the web address www.sbc-support.ch The file name is DOC_CP44.zip .

You need to retrieve it in order to use it. This is done from the SIMATIC Manager -> Menu **FILE** -> **Retrieve**.

In the project you will find two PLC.

PCD_SEND	: This PCD will send data with the 3964
PCD_RECEIVE	: This PCD will receive data with the 3964.

In order to make this demo work correctly, you need to connect the COM1 of the two PCD together, using a PCD2.F120 module.

One PCD will send data in a selected mode and the other will receive them in the same mode.

The 3964 mode is selected in the PCD software on the network 1 of the OB100.

Then in both PCD there are FC for each mode:
FC1 : will run on the 3964
FC2 : will run on the 3964R

All the FC are structured in the same way.

SEND_PCD:
Network 1: load length of the frame to send
Network 2: Call the SFB12 (send)
Network 3: Between each SEND of Frame wait 2000 cycle (for the example only)
Network 4: Count the number of sent frame and the number of error.

RECEIVE_PCD:
Network 1: Call the SFB13 (receive)
Network 2: Count the number of sent frame and the number of error.

The only difference between the 3964 and the 3964R is on the configuration, in the OB100.

The VAT1: show you how many frame have been SENT or RECEIVED on the PCDs, or also how many error occurred.

5 RK512

Introduction

The RK512(R) computer connection controls data transmission via point-to-point connection between the PCD xx7 and a communication partner. As well as the physical layer (ISO-layer 1) and the data-connection layer (ISO-layer2), the RK512 does also the transport layer (ISO layer 4).
The RK512 computer connection also offers higher data integrity and better addressing (because the RK512 uses the 3964(R) protocol for data transport).

Further processing in the communication partner is ensured, because the RK512 interpreter checks the additional length specification in the header and, after storing the data in the destination data block of the communication partner, generates a message frame acknowledging the success or the failure of the data transmission.

Functionality

There are two specific functionalities to the RK512(R) protocol. Those functionalities are the commands SEND and GET.

- The SEND functionality sends a command message frame with user data, and the communication partner replies with a response message frame without user data (acknowledge).
- The GET functionality sends a command message frame without data (request), and the communication partner replies with a response message frame with user data.

If the volume of the data exchanged exceeds 128 bytes, SEND and GET messages frames are automatically accompanied by continuation message frame.

Each message frame of the RK512(R) begins with a header. It can contain message frame Ids, information on the data destination or source and an error number.

5.1 Initialize the PCD.xx7 for the RK512(R) protocol

In order to use any serial com port of the PCD xx7 you need to initialize and configure it. This is done with the SFC245. To set the protocol on a serial port you need to execute the SFC245 only one time.

This SFC is also use to configure the COM port for the other communication protocol as 3964 and ASCII, then some parameters are not needed for the RK512 protocol. The column “Selected Mode” will indicate you, which are the parameters needed for the RK512 protocol.

Name	Para	Type	Possible Values	Selected Mode		Remark
				3	4	
Port	IN0	INT	0…6	■	■	COM port number (see chapter 2.1)
Mode	IN1	INT	3…4	■	■	3 RK 512 4 RK 512 R
Baud Rate	IN2	DINT		■	■	Baud rate (see chapter 2.1)
Data Bit	IN3	INT	7…8	■	■	Number of Data bit
Stop Bit	IN4	INT	1…2	■	■	Number of stop bit
Parity	IN5	INT	0…4	■	■	0 None 1 Even 2 Odd 3 Force Low 4 Force High
Control	IN6	INT	0…3	■	■	Interface type 0 RS 232 1 RS485 2 RS422 3 TTY
XON	IN7	BYTE	0…FFh	░	░	Not use
XOFF	IN8	BYTE	0…FFh	░	░	Not use
WaitSend	IN9	WORD	0…FFFEh	░	░	Not use
WaitInactiv	IN10	WORD	0…FFFEh	░	░	Not use
TelCount	IN11	INT	1	░	░	Number of Frame in the buffer
Overwrite	IN12	BOOL	FALSE / TRUE	░	░	FALSE: Can’t overwrite frame TRUE: can overwrite frame in the buffer, but only if TelCount = 1.
DelRxPuffer	IN13	BOOL	FALSE / TRUE	░	░	Not use
DKPriority	IN14	BOOL	FALSE / TRUE	■	■	True for this PCD to be the high priority partner
ZVZ	IN15	WORD	0, 1…FFFEh	■	■	Time out between character, set by step of 10 ms. 0 for the default value of 220 ms.
QVZ	IN16	WORD	0, 1…FFFEh	■	■	Acknowledge Time out, time for waiting an answer, set by step of 10 ms. “0 “for the default value. Default value is 550 ms for the RK512 and 2000 ms for the RK512R.
TryToConnect	IN17	INT	0…255	■	■	Number of time it will retry to connect to the partner. “0” for the default value of 6 .
TryToSend	IN18	INT	0…255	■	■	Number of time it will retry to send the data. “0” for the default value of 6
FixedLen	IN19	INT	1…1024	░	░	Frame length in receiving
EndChar1	IN20	BYTE	0…255	░	░	End character 1
EndChar2	IN21	BYTE	0…255	░	░	End character 2
SENDBuffer	IN22	INT	0…4000	■	■	Send buffer size (in bytes)
RCVBuffer	IN23	INT	0…4000	■	■	Receive buffer size (in bytes)
Dummy_I0	IN24	INT	0	░	░	Not use
Dummy_W1	IN25	WORD	0	░	░	Not use
Dummy_W2	IN26	WORD	0	░	░	Not use
Dummy_DW1	IN27	DWORD	0	░	░	Not use
RetVal	OUT	WORD		■	■	Result of the operation, see chapter 4.1.2

░ Not needed ■ Required

5.1.1 Example

In this example we configure the COM port1 of the PCD.xx7 to use the RK512R protocol. Baud rate 9600, 1 stop bit, no parity. The interface is a RS232. The Time Out between characters (ZVZ) is fixed at 200 ms, Time out for the acknowledge (QVZ) is the default one of 2000 ms. The size of the receiving and sending buffer is 4000 bytes. This station has the high priority. The number of try to connect and try to send is set to the default value of 6.

```
CALL  SFC  245
       IN0    :=1           // Serial Port N°1
       IN1    :=4           // RK512R
       IN2    :=L#9600      // Baud Rate
       IN3    :=8           // Data Bit
       IN4    :=1           // Stop Bit
       IN5    :=0           // Parity (None)
       IN6    :=0           // RS232 interface
       IN7    :=B#16#0      // Not use
       IN8    :=B#16#0      // Not use
       IN9    :=W#16#0      // Not use
       IN10   :=W#16#0      // Not use
       IN11   :=1           // TelCount
       IN12   :=FALSE       // Can't Overwrite Frame
       IN13   :=FALSE       // Not use
       IN14   :=TRUE        // DK priority is high for this PCD
       IN15   :=W#16#14     // ZVZ (between character) = 200 ms
       IN16   :=W#16#0      // QVZ (Acknowledge) default value
       IN17   :=0           // Try to Connect, default value is 6
       IN18   :=0           // Try to send, default value is 6
       IN19   :=0           // Not use
       IN20   :=B#16#0      // Not use
       IN21   :=B#16#0      // Not use
       IN22   :=4000        // 4000 bytes for the SENDBuffer
       IN23   :=4000        // 4000 bytes for the RCVBuffer
       IN24   :=0           // Not use
       IN25   :=W#16#0      // Not use
       IN26   :=W#16#0      // Not use
       IN27   :=DW#16#0     // Not use
       RET_VAL:=MW240       //
```

5.1.2 Return value of the SFC245

Value	Description
0	Initialization was done correctly
-1	Not valid COM port number
-2	Not enough S7 memory to create the Buffer, but will be possible if you compress the S7 memory.
-3	Not enough S7 memory to create the Buffer, even if you compress the compress the S7 memory.
-4	Not valid parameter Mode
-5	Not valid interface parameters (Baud rate, Data bits, Stop bit or Parity)
-6	Not valid value in WaitSend or WaitInactiv parameter.
-7	Not valid value in TelCount Parameter
-8	Not valid value in ZVZ or QVZ Parameter
-9	Not valid value in TryToConnect or TryToSend parameter
-10	ASCII – Fixed length: The length of the frame is bigger than the RCV-buffer size
-11	Not valid value in SENDBuffer or RCVBuffer parameter
-12	The total memory of the RCVBuffer and the SENDBuffer is bigger than the 64k Bytes allowed
-13	The SFC was called with the sum of the parameter RCVBuffer and the SENDBuffer different from the first call of the SFC.

Note:

It’s possible to change the communication protocol during execution time, but some rules have to be respected.

1) The buffers size can’t be changed
2) For the PCD to consider the new mode, the SEND, RECEIVE or GET function needs a rising edge on their Enable input.

5.2 Sending data with RK512(R)

When the COM port is configured in the RK512 mode, there are two possible way to transmit data between the PCD and a communication partner.

1) In the sender side you will use the SFB12 (SEND) and in the other side you will need the SFB13 (RECEIVE), it’s like the DK3964 (to use this way, refer to the chapter 4.2 Sending data with DK3964).
2) In the sender side you will use the SFB12 (SEND), but you will send it to a mailbox of the communication partner (receiver), then the communication partner will interpret the incoming data. If the receiver is a PCD.xx7 this one doesn’t need any program, just the SFC245 has to be run once to configure the serial com in RK512. The following chapter will document this possibility.

5.2.1 Synchronization

As the receiving PCD (receiver) doesn’t need to be programmed, all the operations of writing data are totally transparent for this system. This could lead to problems, if synchronization between the two systems is needed. To avoid this problem a function in the protocol is available. In fact there is the possibility to use a flag in the receiver system which signals the new incoming data. This flag is called the IPC. **IPC Flag** (Inter-Processor Communication Flag)
This flag is located on the receiver but has to be configured in the sender side, through the parameter R_ID. The IPC flag has two functions:

1) To indicate that new data are on the mailbox (the flag is set)
2) To prevent the overwriting of the data in the mailbox.

In fact, when using the IPC flag, data send to the partner will set a flag in the receiver and won’t be possible to send again data to this partner if the previous flag is not reset.

5.2.2 Parameter SFB 12

Parameter	Description
REQ	With a positive edge, it will start the SEND procedure.
R	With a positive edge, it will cancel and reset the sending.
ID	Serial COM port number.
R_ID	Data destination (Mailbox) and IPC flag specification.
DONE	SEND procedure is done, **DONE** stay at the value true for one cycle.
ERROR	Error in the send procedure, Output stay at the value true for one cycle.
STATUS	Error code
SD_1	Area source of the data to be SEND, this parameter is an ANY pointer, but the length of the any pointer is not considered here. It will be taken from the **LEN** parameter.
LEN	4 KB maximum can be sent in one time, of course the SEND buffer size has to be declare equal or higher.

There are also some conditions regarding the state machine of the function, see chapter 1.4 for details.

5.2.3 R_ID parameter

Byte	Description	
Byte 0 { Bit 0…3 }	0 → Destination DX	1 → Destination DB
Byte 0 { Bit 4…7 }	Not use (0H)	Bit number of the IPC flag
Byte 1	Not use (00H)	Byte number of the IPC flag (between 1..254)
Byte 2	Not use (00H)	Offset 0..255 (word)
Byte 3	Not use (00H)	DB-Number (1..255)

Byte 0: DX or DB
DX mode is use for communication partner, which does not support the MailBox system, the whole R_ID = 0h.
DB mode needs the communication partner to support the mailbox system.
Byte 2 and 3: Destination Offset and DB-Number
This indicates where the Mailbox on the communication partner is. The mailbox has to be in a Data Block.
Byte 2 specifies the offset inside the data block in WORD (value between 0..255).
Byte 3 specifies the DB number (value between 1..255)

5.2.4 Examples

In this example the R_ID is configure in the following way:

- Send to a mailbox, of the DB33 (21h), with offset of 2 words (02h)
- Use the IPC flag M30.3, which is the M30 (1Eh) , bit 3 of the communication partner.

```
CALL  SFB   12 , DB12
       REQ   :=M300.0
       R     :=M300.1
       ID    :=W#16#1
       R_ID  :=DW#16#311E0221
       DONE  :=M300.2
       ERROR :=M300.3
       STATUS:=MW302
       SD_1  :=P#DB100.DBX 0.0 BYTE 200
       LEN   :=MW304
```

5.2.5 Return parameter STATUS

Value	Description
-9 (FFF7h)	Telegram can't be copied
-8 (FFF8h)	Length is too large
-7 (FFF9h)	Unknown type
-6 (FFFAh)	Invalid destination area
-5 (FFFBh)	Invalid length
-4 (FFFCh)	DB is not loaded
-3 (FFFDh)	Invalid DB number
1	Communication problems (check the transmission line)
2	Function can't be executed due to the communication partner
5	Reset request received
9	Can't write data in the partner, because it's lock by IPC
10	Data couldn't be written in the partner station due to missing mailbox area.
11	Warning: can't execute the function, because this one is already being executed.

Note:
With this function, you can read data from a DB and write it directly to a DB of the communication partner. It's only from DB to DB.

5.3 Receiving data with RK512(R)

There is two ways to receive data in the RK512 mode.

- An active way which consist of running the SFB13 (BRCV), and the sender side as to use the SFB12 in the right mode. This is like the DK3964(R) mode, see chapter 4.3.
- A passive mode, where the PLC will receive data without doing anything. You just need to configure at the beginning the COM port to the right protocol. Depending on the choice made by the sender, you can be notified that new data arrived with an IPC flag (Inter-Processor Communication flag).

How to proceed :

1) call the SFC245 to configure the COM port
2) If you use the IPC flag: a rising edge of the IPC flag mean that new data arrived in the DB.
3) If you use the IPC flag: reset the IPC flag to allow new incoming data.

5.4 Fetching data with RK512(R)

The GET functionality allows you to fetch data from a communication partner (server) without needing to program anything to this last one (need just to be configured).
The protocol will handle all necessary operations to get the data, just execute the SFB14.

5.4.1 Synchronization

As the partner PCD (server) doesn't need to be programmed. All the operations of reading data are totally transparent for this system. This could lead to problems if synchronization between the two systems is needed. To avoid this problem a function in the protocol is available. In fact there is the possibility to use a flag, which signals when data are read. This flag is called the IPC.
IPC Flag (Inter-Processor Communication Flag)
This flag is located on the server, but has to be configured in the fetching side, through the parameter ID of the SFB14. The IPC flag has two functions:

1) To indicate that data have read (the flag is set).
2) To avoid the reading of old data.

In fact, when using the IPC flag, data read from the partner will set a Flag in the server and won't be possible to read again data to this server if the previous Flag is not reset.

5.4.2 Parameter SFB 14

Parameter	Description
REQ	Positive edge start to fetch data
ID	Serial COM port number of the xx7 and information about the IPC Flag (See 5.5.2)
NDR	New Data Ready. When the operation of fetching data is complete, this output stays at level high for one cycle.
ERROR	Error during operation. This output stay at level high for one cycle.
STATUS	Error code
ADDR_1	Source area of the data inside the communication partner.
ADDR_2	Not use
ADDR_3	Not use
ADDR_4	Not use
RD_1	Destination area of the data. It's an ANY pointer type. The length of the data as to be the same as the source.
RD_2	Not use
RD_3	Not use
RD_4	Not use

5.4.3 ID parameter

Bit	Description
0…3	Serial COM port of the PCD.xx7
4…7	Bit number of the IPC Flag
8…15	Byte number of the IPC Flag (1..254)

5.4.4 Call Example

In this example we use the IPC flag, which is the M40.4, which is embedded in the parameter ID as: M40 => 28h, .4 => 4h, COM1 => 1h, all together => 2841h.

```
CALL  SFB   14 , DB14
      REQ   :=M400.0                  // Request to GET new data
      ID    :=W#16#2841               // COM Port =1 ; IPC Flag is M40.4
      NDR   :=M400.2                  // New Data Received
      ERROR :=M400.3                  // Error during the operation
      STATUS:=MW402                   // Error code
      ADDR_1:=P#DB200.DBX 0.0 BYTE 100 // Source of the Data
      ADDR_2:=
      ADDR_3:=
      ADDR_4:=
      RD_1  :=P#DB100.DBX 0.0 BYTE 100 // Destination of the Data
      RD_2  :=
      RD_3  :=
      RD_4  :=
```

5.4.5 Parameter Status

Value	Error	Description
-11 (FFF5h)	1	The actual mode of the serial COM Port is not RK512
-10 (FFF6h)	1	Invalid parameter ADDR_1
-2 (FFFEh)	1	COM Port not initialized
-1 (FFFFh)	1	Invalid COM Port number
0	0	OK
1	1	Communication problems (check the transmission line)
2	1	Negative Answer from the Communication Partner
9	1	Can access to the data, because locked by IPC Flag
10	1	Invalid parameter RD_1

5.5 Examples

An example has been made to show how work the RK512 protocol.
You download this example from the web address www.sbc-support.ch The file name is DOC_CP44.zip .

You need to retrieve it in order to use it. This is done from the SIMATIC Manager -> Menu **FILE** -> **Retrieve**.

In the project you will find two PLC.

PCD_SEND
FC3 : This FC3 will send data to mailbox of the communication partner (PCD_RECEIVE) using the RK512 protocol.
FC4 : This FC4 will fetch data from a DB the communication partner (PCD_RECEIVE) using the RK512 protocol.

PCD_RECEIVE
FC3 : This FC3 will take care of the IPC flag, by counting the number of transmission and resetting the IPC flag.
FC4 : This FC3 will take care of the IPC flag, by counting the number of fetch done and resetting the IPC flag.

In order to make this demo work correctly, you need to connect the COM1 of the two PCD together, using a PCD2.F120 module.

One PCD will send data in a selected mode and the other will receive them in the same mode.

The RK512(R) mode is selected in the PCD software on the network 1 of the OB100.

SEND_PCD:
Network 1: load length of the frame to send
Network 2: Build parameter R_ID, with the IPC flag.
Network 3: Call the SFB12 (send)
Network 4: Between each SEND of frame wait 2000 cycle (for the example only)
Network 5: Count the number of sent frames and the number of errors.

RECEIVE_PCD:
Network 1: Count the number of transactions and reset the IPC flag.

The only difference between the RK512 and the RK512R is on the configuration which is done in the OB100.

The VAT1: shows you how many frame have been SENT or RECEIVED on the PCDs, or also how many error occurred.

6 RK512 (multi-point communication)

Introduction

The RK512(R) computer connection procedure controls data transmission normally via point-to-point connection between the PCD xx7 and a communication partner. But it’s also possible to have multi-point communication using only PCD.xx7 as communication with the RS485 interface.

This multi-point network is based on a Master-Slave communication, where only one master is allowed. The master is writing or pulling the data to or from the slave systems.

How does the multi-point work?

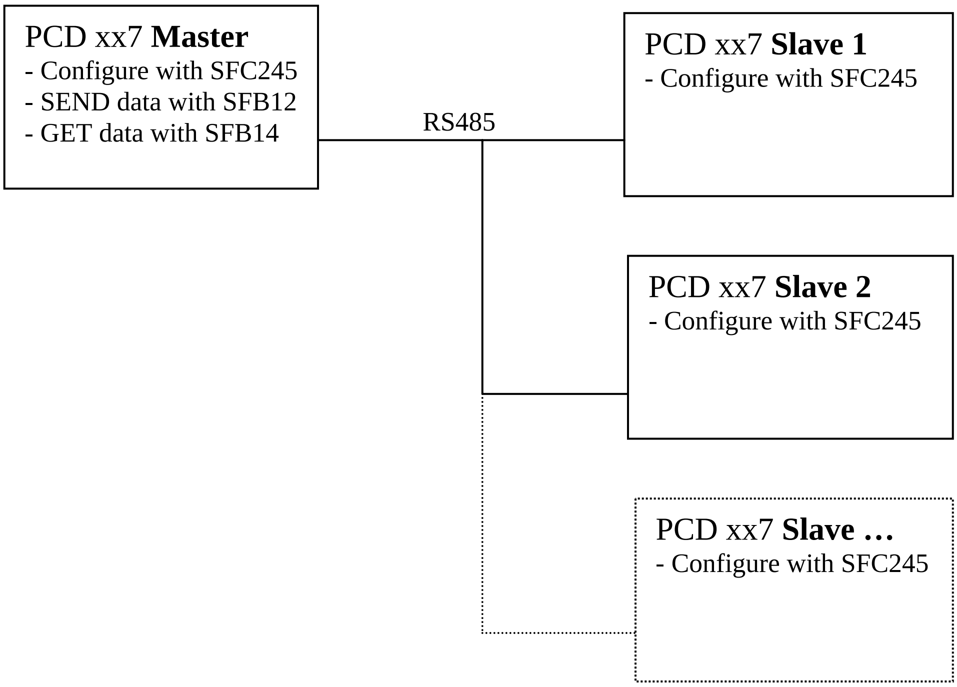
It uses a specific function of the DUART, called the Multi DROP mode.
The multi DROP mode is a different way of using the PARITY bit.
In fact the PARITY bit is not used as a parity bit any more, its new meaning is a slave address flag. In fact when the Master is opening the communication with a slave, it sends the address of the concerned slave, and force the PARITY bit to low. For all the other data, the PARITY bit will be forced to high.

Functionality

In the RK512 multi-point mode, two functionalities can be used by the master only. Theses functionalities are the commands SEND and GET.

- SEND (SFB12): will allow the master to write in a Data Block of any Slave.
- GET (SFB14): will allow the Master to read a Data Block of any Slave.

Structure



This multi-point is based on RS485 network (details about the correct wiring can be found on the manual 26/740 E).
Only **one** master system is allowed in the network.

The master system is configured as master by executing the SFC245 with the right parameter (DKPriority=True). The slaves system will be configured as Slave (DKPriority=False) with their address (IN25= own_address) by executing the SFC245.

Then the master is the only system which needs to be program for the data exchange, in fact the master will SEND or GET data from the slaves system one by one. Only one Slave can be accessed at time and also only one job can be proceed at time.

6.1 Initialize the PCD.xx7 for the RK512(R) multi-point

In order to use any serial com port of the PCD xx7 you need to initialize and configure it. The SFC245 configure and set the protocol on a serial port. The SFC245 needs to be executed only once. It can be called in the OB100.

This SFC is also used to configure the COM port for the other communication protocol as 3964 and ASCII, then some parameter are not needed for the RK512 protocol. The column “Selected Mode” will indicate you, which are the parameters needed for the RK512 multi-point mode.

Name	Para	Type	Possible Values	Selected Mode		Remark
				3	4	
Port	IN0	INT	1,3,5,6	■	■	COM port number (see chapter 2.1)
Mode	IN1	INT	9	■	■	3 RK 512 4 RK 512 R
Baud Rate	IN2	DINT		■	■	Baud rate (see chapter 2.1)
Data Bit	IN3	INT	7…8	■	■	Number of Data bit
Stop Bit	IN4	INT	1…2	■	■	Number of stop bit
Parity	IN5	INT	5	■	■	5 Multi-point
Control	IN6	INT	1	■	■	Interface type 1 RS485
XON	IN7	BYTE	0…FFh	░	░	Not use
XOFF	IN8	BYTE	0…FFh	░	░	Not use
WaitSend	IN9	WORD	0…FFFEh	░	░	Not use
WaitInactiv	IN10	WORD	0…FFFEh	░	░	Not use
TelCount	IN11	INT	1	░	░	Number of Frame in the buffer
Overwrite	IN12	BOOL	FALSE / TRUE	░	░	FALSE: Can’t overwrite frame TRUE: can overwrite frame in the buffer, but only if TelCount = 1.
DelRxPuffer	IN13	BOOL	FALSE / TRUE	░	░	Not use
DKPriority	IN14	BOOL	FALSE / TRUE	■	■	True if this PCD is the Master
ZVZ	IN15	WORD	0, 1…FFFEh	■	■	Time out between character, set by step of 10 ms. 0 for the default value of 220 ms.
QVZ	IN16	WORD	0, 1…FFFEh	■	■	Acknowledge Time out, time for waiting an answer, set by step of 10 ms. “0 “for the default value. Default value is 550 ms for the RK512 and 2000 ms for the RK512 R .
TryToConnect	IN17	INT	0…255	■	■	Number of time it will retry to connect to the partner. “0” for the default value of 6 .
TryToSend	IN18	INT	0…255	■	■	Number of time it will retry to send the data. “0” for the default value of 6
FixedLen	IN19	INT	1…1024	░	░	Frame length in receiving
EndChar1	IN20	BYTE	0…255	░	░	End character 1
EndChar2	IN21	BYTE	0…255	░	░	End character 2
SENDBuffer	IN22	INT	0…32768	■	■	Send buffer size (in bytes)
RCVBuffer	IN23	INT	0…32768	■	■	Receive buffer size (in bytes)
Dummy_I0	IN24	INT	0	░	░	Not use
Dummy_W1	IN25	WORD	0	▥	▥	Own network address
Dummy_W2	IN26	WORD	0	▨	▨	Use only for the Master: If the slave address is on a DB, then DB number in this parameter.
Dummy_DW1	IN27	DWORD	0	■	■	Pointer of the Slave address
RetVal	OUT	WORD		■	■	Result of the operation, see chapter 6.1.2

░ Not needed ■ Required ▨ Only Master ▥ Only Slave

6.1.1 Example Master system configuration

In this example we configure the COM port1 of the PCD.xx7 to use the RK512R protocol. Baud rate 9600, 8 Data bits, 1 stop bit. The interface has to be RS485. The Time Out between characters (ZVZ) is fixed at 200 ms, Time out for the acknowledge (QVZ) is the default one of 2000 ms. The size of the receiving and sending buffer is 4000 bytes. This station is the MASTER on the network. The number of try to connect and try to send is set to the default value of 6.
This has to be executed only once, it could be executed in the OB100.

```
L     P#M 20.0             // Address of the Slave (destination)
T     #DestPointer         // will be in the MB20

L     0                    // Slave address is in Flag MB20 not
T     #DBNumber            // in a DB, so DBNumber=0

CALL  SFC  245
       IN0    :=1           // Serial Port N°1
       IN1    :=4           // RK512R
       IN2    :=L#9600      // Baud Rate
       IN3    :=8           // Data Bit
       IN4    :=1           // Stop Bit
       IN5    :=9           // Multi-Point communication
       IN6    :=1           // RS485 interface
       IN7    :=B#16#0      // Not use
       IN8    :=B#16#0      // Not use
       IN9    :=W#16#0      // Not use
       IN10   :=W#16#0      // Not use
       IN11   :=1           // TelCount
       IN12   :=FALSE       // Can't Overwrite Frame
       IN13   :=FALSE       // Not use
       IN14   :=TRUE        // This system is the network MASTER
       IN15   :=W#16#14     // ZVZ ( between character) = 200 ms
       IN16   :=W#16#0      // QVZ (Acknowledge) default value
       IN17   :=0           // Try to Connect, default value is 6
       IN18   :=0           // Try to send, default value is 6
       IN19   :=0           // Not use
       IN20   :=B#16#0      // Not use
       IN21   :=B#16#0      // Not use
       IN22   :=4000        // 4000 bytes for the SENDPuffer
       IN23   :=4000        // 4000 bytes for the RCVPuffer
       IN24   :=0           // Not use
       IN25   :=w#16#0      // Not use
       IN26   :=#DBnumber   // DB number in which the address is
       IN27   :=#destpointer// Pointer of the Slave address
       RET_VAL:=MW240       // Return value
```

6.1.2 Example Slave system configuration

The communication properties will be the same as the master system (6.1.3), but it will be configure as a Slave and the address will be **4** .
This has to be executed only once, it could be executed in the OB100.

```
L     4                    // Preset
T     #myadr               // address 4

L     P#M 20.0             // Not relevant for the slave system
T     #DestPointer         // just has to be a pointer

L     0                    // Not relevant for the slave system
T     #DBNumber

CALL  SFC  245
       IN0    :=1            // Serial Port N°1
       IN1    :=4            // RK512R
       IN2    :=L#9600       // Baud Rate
       IN3    :=8            // Data Bit
       IN4    :=1            // Stop Bit
       IN5    :=9            // Multi-Point communication(withIPC)
       IN6    :=1            // RS485 interface
       IN7    :=B#16#0       // Not use
       IN8    :=B#16#0       // Not use
       IN9    :=W#16#0       // Not use
       IN10   :=W#16#0       // Not use
       IN11   :=1            // TelCount
       IN12   :=FALSE        // Can't Overwrite Frame
       IN13   :=FALSE        // Not use
       IN14   :=FALSE        // This system is a SLAVE
       IN15   :=W#16#14      // ZVZ ( between character) = 200 ms
       IN16   :=W#16#0       // QVZ (Acknowledge) default value
       IN17   :=0            // Try to Connect, default value is 6
       IN18   :=0            // Try to send, default value is 6
       IN19   :=0            // Not use
       IN20   :=B#16#0       // Not use
       IN21   :=B#16#0       // Not use
       IN22   :=4000         // 4000 bytes for the SENDPuffer
       IN23   :=4000         // 4000 bytes for the RCVPuffer
       IN24   :=0            // Not use
       IN25   :=#myadr       // Own address on the network
       IN26   :=#DBnumber    // DB number in which the address is
       IN27   :=#destpointer// Pointer of the Slave address
       RET_VAL:=MW240        // Return value
```

6.1.3 Return value of the SFC245

Value	Description
0	Initialization was done correctly
-1	Not valid COM port number
-2	Not enough S7 memory to create the buffer, but will be possible if you compress the S7 memory.
-3	Not enough S7 memory to create the buffer, even if you compress the compress the S7 memory.
-4	Not valid parameter Mode
-5	Not valid interface parameters (Baud rate, Data bits, Stop bit or Parity)
-6	Not valid value in WaitSend or WaitInactiv parameter.
-7	Not valid value in TelCount parameter
-8	Not valid value in ZVZ or QVZ parameter
-9	Not valid value in TryToConnect or TryToSend parameter
-10	ASCII – Fixed length: The length of the frame is bigger than the RCV-buffer size
-11	Not valid value in SENDBuffer or RCVBuffer parameter
-12	The total memory of the RCVBuffer and the SENDBuffer is bigger than the 64k Bytes allowed
-13	The SFC was called with the sum of the parameter RCVBuffer and the SENDBuffer different from the first call of the SFC.
-14	Own address not valid
-15	DB number for the Salve address is not valid
-16	Not valid pointer type
-17	Not valid offset in the pointer

Note:

6.2 Sending data with RK512(R) in multi-point

The SFB12(SEND) sends data, using it in the same way as the RK512 protocol. As we are in a multi-point configuration, we need to specify with which system we would like to make connection. To do it, you just need to write the integer value of the slave address on the Memory Byte selected for this purpose (by the SFC245; parameter IN27). This address will be taken in consideration as soon as you call the SFB12.
The master will then send the data and write them directly to the destination area of the slave, not programming is needed to receive the data in the Slave PCD, only the COM port needs to be initialize with the SFC245.

6.2.1 Parameter SFB 12

Parameter	Description
REQ	With a positive edge, it will start the SEND procedure.
R	With a positive edge, it will cancel and reset the sending.
ID	Serial COM port number.
R_ID	Data destination (Mailbox) and IPC flag specification.
DONE	SEND procedure is done, **DONE** stay at the value true for one cycle.
ERROR	Error in the send procedure, output stays at the value true for one cycle.
STATUS	Error code
SD_1	Area source of the data to be SEND, this parameter is an ANY pointer, but the length of the any pointer is not considered here, it will be taken from the **LEN** parameter.
LEN	4 KB maximum can be sent in one time, of course the SEND buffer size has to be declared equal or higher.

6.2.2 R_ID parameter:

Byte	Description	
Byte 0 { Bit 0…3 }	0 → Destination DX	1 → Destination DB
Byte 0 { Bit 4…7 }	Not use (0H)	Bit number of the IPC flag
Byte 1	Not use (00H)	Byte number of the IPC flag (between 1..254)
Byte 2	Not use (00H)	Offset 0..255 (word)
Byte 3	Not use (00H)	DB-Number (1..255)

Byte 0: DX or DB
DX mode is use for communication partner, which does not support the MailBox system, the whole R_ID = 0h.
DB mode needs the communication partner to support the Mailbox system.

IPC Flag (Inter-Processor Communication Flag)
This flag is located on the communication partner and its functions are two:

1) To indicate that new data are on the mailbox
2) To prevent the overwriting of the data in the mailbox

More details are described in the chapter 5.2.1

Byte 2 and 3: Destination Offset and DB-Number
This indicates where the mailbox on the communication partner is. The mailbox has to be in a Data Block.
Byte 2 specify the offset inside the data block in WORDs (value between 0..255).
Byte 3 specify the DB number (value between 1..255)

6.2.3 Example

We will connect and send data to the Slave address **4** => MB20
The R_ID is configured in the following way:

- Send to a mailbox, of the DB33 (21h), with offset of 2 words (02h)
- Use the IPC flag, which is the M30 (1Eh), bit 3 of the communication partner.

```
      L     4               // SLAVE address ; Destination
      T     MB    20        // is loaded in the address byte

 CALL  SFB   12 , DB12
        REQ   :=M300.0
        R     :=M300.1
        ID    :=W#16#1
        R_ID  :=DW#16#311E0221
        DONE  :=M300.2
        ERROR :=M300.3
        STATUS:=MW302
        SD_1  :=P#DB100.DBX 0.0 BYTE 200
        LEN   :=MW304
```

6.2.4 Return parameter STATUS

Value	Description
-9 (FFF7h)	Telegram can’t be copied
-8 (FFF8h)	Length is too large
-7 (FFF9h)	Unknown type
-6 (FFFAh)	Invalid destination area
-5 (FFFBh)	Invalid length
-4 (FFFCh)	DB is not loaded
-3 (FFFDh)	Invalid DB number
1	Communication problems (check the transmission line)
2	Function can’t be executed due to the communication partner
5	Reset request received
9	Can’t write data in the partner, because it’s lock by IPC
10	Data couldn’t be written in the partner station due to missing mailbox area.
11	Warning: can’t execute the function, because this one is already being executed.

Note:
With this function, you can read data from a DB and write it directly to a DB of the communication partner. It’s only from DB to DB.

6.3 Fetching data with RK512(R) in multi-point

In this multi-point configuration, the slave systems can’t send data. The master is fetching the Data from the slave.

To fetch the data the master system will use the SFB14 (GET). On the slave side no programming is needed. The protocol will handle all the operation, but the COM port has just to be configured with the SFC245 to support the RK512 multi-point (Slave).
Before to fetch data from a slave the communication with this slave has to be opened, so you need to specify which slave you like to get connected with. This is done by writing the integer value of the slave address on the memory byte selected for this purpose (by the SFC245; parameter IN27). This address will be taken in consideration as soon as you call the SFB14.

6.3.1 Parameter SFB 14

Parameter	**Description**
REQ	Positive edge starts to fetch data
ID	Serial COM port number of the xx7 and information about the IPC Flag (See 5.5.2)
NDR	New Data Ready. When the operation of fetching data is completed, this output stays at level high for one cycle.
ERROR	Error during operation. This output stays at level high for one cycle.
STATUS	Error code
ADDR_1	Source area of the data inside the communication partner.
ADDR_2	Not use
ADDR_3	Not use
ADDR_4	Not use
RD_1	Destination area of the data. It’s an ANY pointer type, the length of the data as to be the same as the source.
RD_2	Not use
RD_3	Not use
RD_4	Not use

6.3.2 ID parameter

Bit	**Description**
0…3	Serial COM port of the PCD.xx7
4…7	Bit number of the IPC Flag
8…15	Byte number of the IPC Flag (1..254)

Description details can be found in chapter 5.4.1

6.3.3 Call Example

We will connect and get data from the Slave address **3** => MB20
The ID is configure in the following way:

- Communication port = 1
- Use the IPC flag, which is the M40 (28h) , bit 4 of the communication partner.

```
L      3                        // Connect to SLAVE address = 3
T      MB    20                 // MB20 was configure by SFC245

CALL  SFB   14 , DB14
     REQ   :=M400.0             // Request to GET new data
     ID    :=W#16#2841          // COM Port =1 ; IPC Flag is M40.4
     NDR   :=M400.2             // New Data Received
     ERROR :=M400.3             // Error during the operation
     STATUS:=MW402              // Error code
     ADDR_1:=P#DB200.DBX 0.0 BYTE 100 // Source of the Data
     ADDR_2:=
     ADDR_3:=
     ADDR_4:=
     RD_1  :=P#DB100.DBX 0.0 BYTE 100 // Destination of the Data
     RD_2  :=
     RD_3  :=
     RD_4  :=
```

6.3.4 Parameter Status

Value	Error	Description
-11 (FFF5h)	1	The actual mode of the serial COM Port is not RK512
-10 (FFF6h)	1	Invalid parameter ADDR_1
-2 (FFFEh)	1	COM Port not initialized
-1 (FFFFh)	1	Invalid COM Port number
0	0	OK
1	1	Communication problems (check the transmission line)
2	1	Negative Answer from the Communication Partner
9	1	Can access to the data, because locked by IPC Flag
10	1	Invalid parameter RD_1

6.4 Examples

An example has been made to show how work the RK512 protocol in multi-point. You download this example from the web address www.sbc-support.ch The file name is DOC_CP44.zip .

You need to retrieve it in order to use it. This is done from the SIMATIC Manager -> Menu **FILE** -> **Retrieve** .

In the project you will find three PLC. One MASTER and two SLAVES.

PCD_SEND (Master)
FC9 : This FC9 will fetch (GET) data from the two Slave system (PCD_RECEIVE) using the RK512 protocol multi-point.
Get data from the slave address 2, then wait 2000 cycles (optional), then GET data from the Slave system address 1, then wait 2000 cycle and restart with Slave address 2.
FC10: Does the same thing, but doesn't use the IPC flag.

PCD_RECEIVE_or_SLAVE1 (address = 1)
FC9 : This FC9 will take care of the IPC Flag, by counting the number of transmission and resetting the IPC flag.

SLAVE2 (address = 2)
FC9 : This FC9 will take care of the IPC Flag, by counting the number of transmission and resetting the IPC flag.

The master system (PCD_SEND) will select a slave (in this example by putting the address of the slave in the MB20) and then fetch data from it. On the other side the slave system checks when data have been taken from it with the IPC flag and will count it.

SEND_PCD (master):
Network 1: Build parameter ID, with the IPC flag and COM port number.
Network 2: Call the SFB14 (GET)
Network 3: Wait 2000 cycle between each GET and then change slave address
Network 4: Count the number of sent frames and the number of errors.

RECEIVE_PCD_SLAVE:
Network 1: Count the number of transactions and reset the IPC flag.

In order to make this demo work correctly, you need to connect the COM1 of the two PCD together, using a PCD2.F110 module.

In the OB100 of each system you will need to select the demo mode 9 for the multi-point with IPC, to run the multi-point demo software.

7 Transparent mode

Introduction

The transparent mode is very close to the ASCII-Fixed length driver, but has more flexibility regarding the length. In fact, for each transmission or receiving, the telegram length can be defined.
The other characteristic is that the buffers are more transparent to the user program, especially the receiving buffer. You can know anytime how many byte are in it.

There are two way to work ins transparent mode, each of them have their own characteristics.

- **SAIA-SFC (NOT for PCD2.M487):**

This mode is the one by default. You can use it through the SFC240, 241, 242 and 243. It’s possible with it to know at any time the status of the receiving, sending buffer and if an overflow occurred. But the maximum length of the telegram is then 128 bytes, which is the size of the receive- and send buffer.

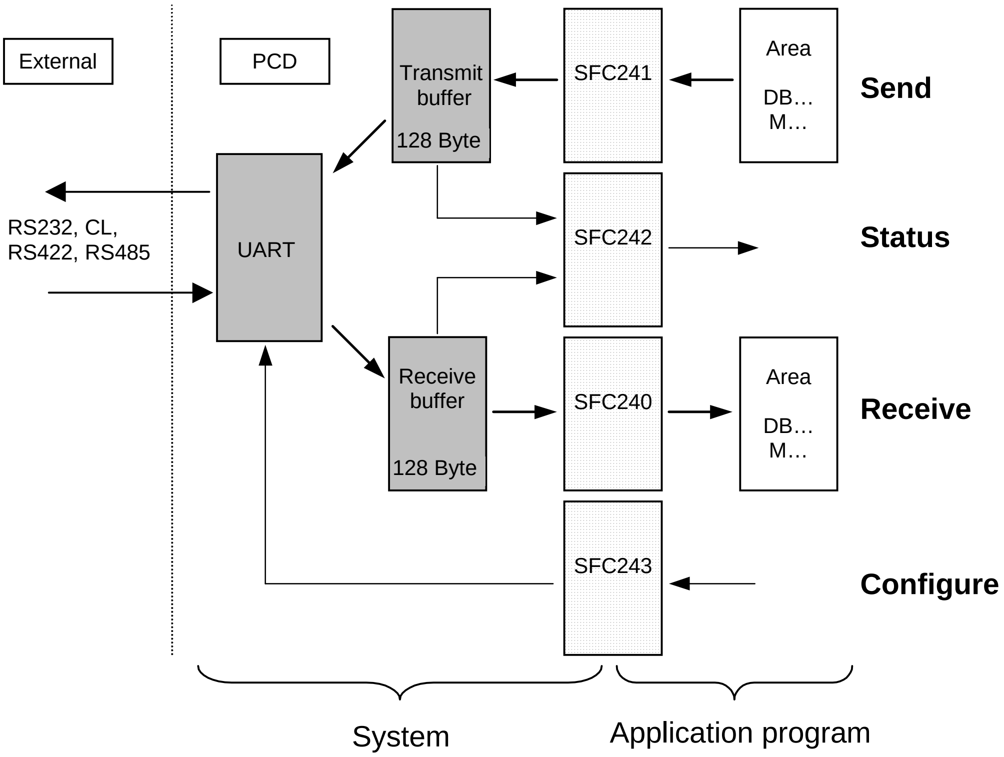
- **CP-SFB:**

This mode has to be configured with the SFC245. It’s only possible to know the status of the receiving buffer. The maximum length of the telegram is depending of the buffer size you configured (up to 4Kb).

Warning
It’s possible to switch from the SAIA-SFC mode to the CP-SFB mode, but the way back is not possible. So after using the CP-SFB on a serial port, it’s not allowed to use the SAIA-SFC.

7.1 SAIA-SFC for the transparent mode

Structure



The UART (Universal Asynchronous Receiver Transmitter) is the interface between the transmission line and the transmit- or receive buffer of a PCD. Data is transferred between the UART and the transmit- or receive buffer via the system program. Data transfer between the transmit- or receive buffer and the S7 memory of the PCD is done by the application program with the help of the specific SFC. This structure exists for each serial com port.

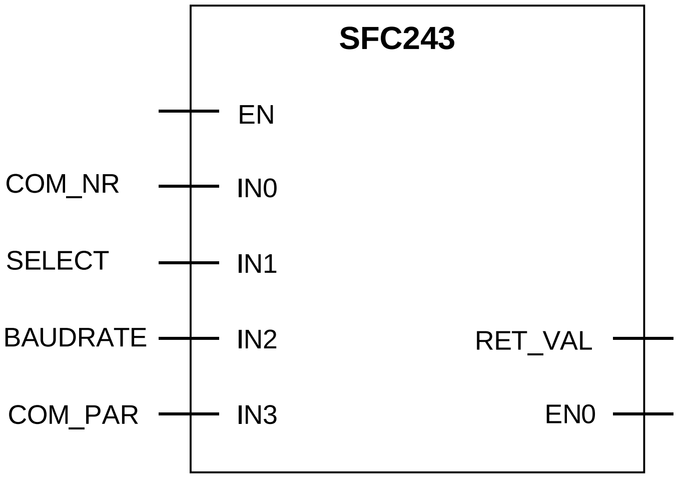
To interface the system with the application program are 4 system functions, which are:

▪ SFC243	Configure the serial port
▪ SFC242	Inquiry the status of the serial port
▪ SFC241	Transmit Data
▪ SFC240	Receive Data

All SFC are described in details in the following chapter.

7.1.1 Configure and Initialise serial COM with SFC 243 "COM_INIT"

When the "COM_INIT" SFC is called, the specified serial interface is initialized.



Parameters

Parameter	Declaration	Memory range	Description
COM_NR	IN / BYTE	E,A,M,D,L,Const.	Interface number 1..5

SELECT	IN / BYTE	E,A,M,D,L,Const.	Interface mode
			RS232 = 0
			RS485 = 1
			RS422 = 2
			CL 20 mA = 3

BAUDRATE	IN / DINT	E,A,M,D,L,Const.	Baud rate
COM_PAR	IN / WORD	E,A,M,D,L,Const.	Initialisation parameters for the interface (see below)
RET_VAL	OUT / WORD	E,A,M,D,L	Error information

Error information

Error code (W#16#....)	Description
0000	no error
00FE	incorrect interface number or incorrect initialisation values

Meaning of “COM_PAR”

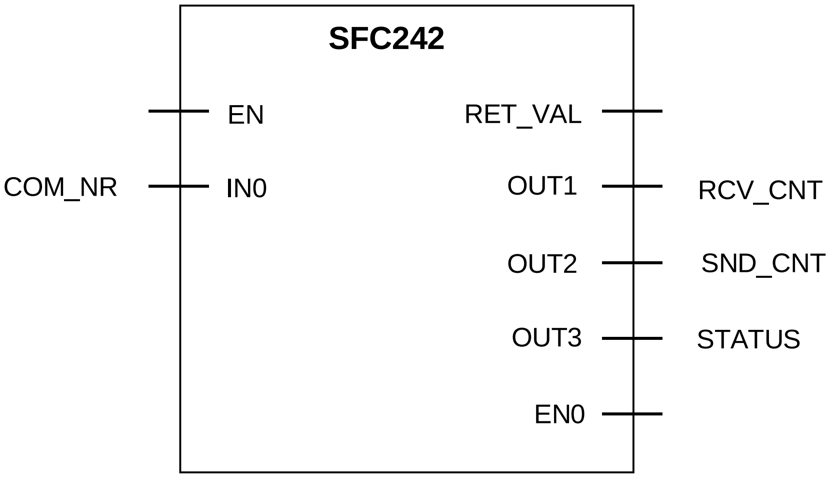
Bits 1..0	Number of data bits (00 = 5, 01 = 6, 10 = 7, 11 = 8)
Bits 4..2	Parity (000 = even, 001 = odd, 010 = force low, 011 = force high, 10x =no)
Bit 5	Stop bits, 0 => 1 stop bit, 1 => 2 stop bits

Permitted baud rates: 300[1], 600[1], 1200, 2400, 4800, 9600, 19200[2], 38400[2])

[1]) Not for PCD2.M487
[2]) 38400 for COM1 only or with CDB for COM2..5 (chapter 2.1)

7.1.2 Status of serial COM with SFC 242 "COM_STAT"

When the "COM_STAT" SFC is called, the status of the specified serial interface is returned.



Parameters

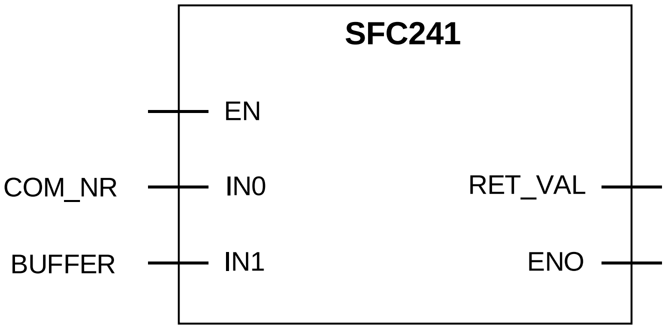
Parameter	Declaration	Memory range	Description
COM_NR	IN / BYTE	E,A,M,D,L,Const.	Interface number 1…5
RET_VAL	OUT / WORD	E,A,M,D,L	Error information
RCV_CNT	OUT / WORD	E,A,M,D,L	Number of bytes in receive buffer
SND_CNT	OUT / WORD	E,A,M,D,L	Number of bytes in transmit buffer
STATUS	OUT / WORD	E;A;M;D;L	Status Bit 0 = 1 → Receive buffer overrun Bit 1 = 1 → Interface error

Error information

Error code (W#16#....)	Description
0000	no error
00FE	incorrect interface number

7.1.3 Transmit with SFC 241 "COM_SEND"

When the "COM_SEND" SFC is called, the number of bytes indicated is transferred from the specified buffer into the transmit buffer. The transmission itself takes place in the background.



Parameters

Parameter	Declaration	Memory range	Description
COM_NR	IN / BYTE	E,A,M,D,L,Const	Interface number 1…5
BUFFER	IN / ANY		Pointer to data source (128 Byte)
RET_VAL	OUT / WORD	E,A,M,D,L	Error information

Error information

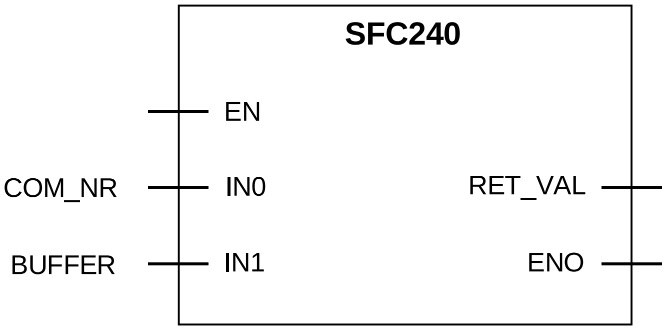
Error code (W#16#....)	Description
0000	no error
00FE	incorrect interface number
0001	insufficient space in transmit buffer

Example:
Send on the serial COM 1, 50 bytes of the DB100, starting at the byte DBB0.
Return error code on the Flag Word MW1040.

```
CALL  SFC  241
  IN0    :=B#16#1
  IN1    :=P#DB100.DBX 0.0 BYTE 50
  RET_VAL:=MW1040
```

7.1.4 RECEIVE with SFC 240 "RCV_COM"

When the "RCV_COM" SFC is called, the number of bytes indicated is transferred from the receive buffer to the buffer specified.



Parameters

Parameter	Declaration	Memory range	Description
COM_NR	IN / BYTE	E,A,M,D,L,Const	Interface number 1…5
BUFFER	IN / ANY		Pointer to destination of data (128 Byte)
RET_VAL	OUT / WORD	E,A,M,D,L	Error information

Error information

Error code (W#16#....)	Description
0000	no error
00FE	incorrect interface number
0001	insufficient bytes in receive buffer

Example:

Transfer from the receive buffer of the serial COM 1, 50 bytes to the DB33, starting at the byte DBB0. Return error code on the Flag Word MW240.

```
CALL  SFC  240
  IN0    :=B#16#1
  IN1    :=P#DB33.DBX 0.0 BYTE 50
   RET_VAL:=MW240
```

7.1.5 Example of communication with SAIA-SFC

An example has been made to show how work in the transparent mode.
You download this example from the web address www.sbc-support.ch The file name is DOC_CP44.zip .

You need to retrieve it in order to use it. This is done from the SIMATIC Manager -> Menu **FILE** -> **Retrieve** .

In the project two PLC are concerned by this example:

PCD_SEND	: This PCD will send 50 characters per time
PCD_RECEIVE	: This PCD will receive the characters.

In order to make this demo work correctly, you need to connect the two COM1 of the PCD together, using a PCD2.F120 module.

One PCD will send frame of 50 characters and the other will wait to receive those 50 characters in its buffer and read it in one time.

The transparent mode is selected in the PCD software on the network 1 of the OB100.

Then in both PCD there is the FC240, which will run for the transparent mode.

The FC240 is structured as follow.

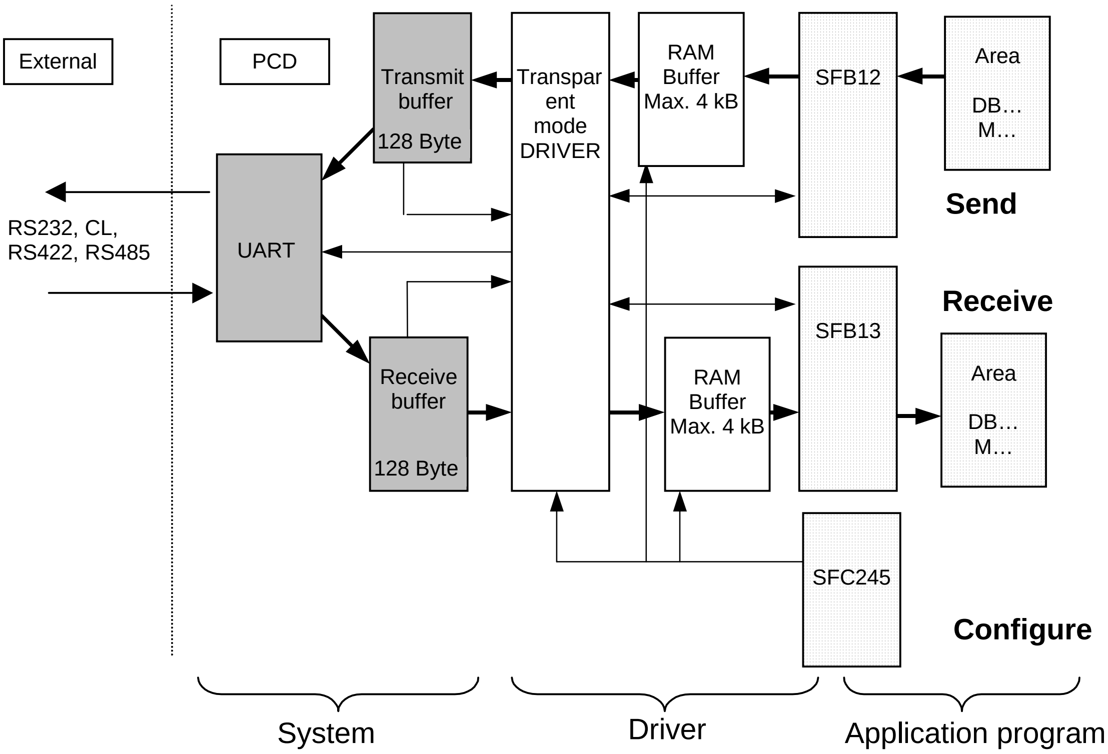
SEND_PCD:
Network 1: Manage the actual step status, SEND or WAIT_TO_SEND.
Network 2: Call the SFC241 to SEND.
Network 3: WAIT 2000 cycles before sending again.

RECEIVE_PCD:
Network 1: Wait to have 50 bytes in the Receive buffer and then read those 50 bytes. Also count of many time the receive buffer had 50 characters.

This example just shows one possible way to use the SFC. There are other possibilities to structure the communication, but this depending on the application.

7.2 CP-SFB for the transparent mode

Structure



The UART (Universal Asynchronous Receiver Transmitter) is the interface between the transmission line and the transmit- or receive buffer of the PCD. Data are transferred between the UART and the transmit or receive buffer via the system program. Now in this transparent mode is a second level of receive- and transmit buffer and the size of those buffer can be configured up to 4 kB. The driver of the transparent mode will take care of the data transfer between the buffers. Then there are two SFB, which link the application software and the driver. This structure exists for each serial com port.

So to interface the system with the application program, there are 2 System Functions Blocks and one SFC to configure it, which are:

SFC245	Configure and initialise the serial port
SFB12	Transmit Data
SFB13	Receive Data

All these functions are described in details in the following chapter.

7.2.1 SFC245 Configure and initialise the serial COM and DRIVER

To use the transparent mode with the SFB, you need to initialize and configure the COM port of the PCD. This is done with the SFC245. To set a mode, you need to execute the SFC245 only one time.
This SFC is also use to configure the COM port for the other communication protocol as RK512 and DK3964. Then some parameters are not needed for the Transparent mode. The darkened areas are not necessary for the Transparent mode.

Name	Para	Type	Possible Values	Remark
Port	IN0	INT	0…6	COM port number (see chapter 2.1)
Mode	IN1	INT	0	0: Transparent mode
Baud Rate	IN2	DINT		Baud rate (see chapter 2.1)
Data Bit	IN3	INT	7…8	Number of Data bit
Stop Bit	IN4	INT	1…2	Number of stop bit
Parity	IN5	INT	0…4	0 None 1 Even 2 Odd 3 Force Low 4 Force High
Control	IN6	INT	0…3	Interface type 0 RS 232 1 RS485 2 RS422 3 TTY
XON	IN7	BYTE	0…FFh	Not use
XOFF	IN8	BYTE	0…FFh	Not use
WaitSend	IN9	WORD	0…FFFEh	Not use
WaitInactiv	IN10	WORD	0…FFFEh	Not use
TelCount	IN11	INT	1	Number of Frame in the buffer
Overwrite	IN12	BOOL	FALSE / TRUE	FALSE: Can't overwrite frame TRUE: can overwrite frame in the buffer, but only if TelCount = 1.
DelRxPuffer	IN13	BOOL	FALSE / TRUE	Not use
DKPriority	IN14	BOOL	FALSE / TRUE	Not use
ZVZ	IN15	WORD	0, 1…FFFEh	Time out set in ms, set to 0 for the default value of 4 ms.
QVZ	IN16	WORD	0, 1…FFFEh	Not use
TryToConnect	IN17	INT	0…255	Not use
TryToSend	IN18	INT	0…255	Not use
FixedLen	IN19	INT	1…1024	Frame length in receiving
EndChar1	IN20	BYTE	0…255	End character 1
EndChar2	IN21	BYTE	0…255	End character 2
SENDBuffer	IN22	INT	0…4000	Send buffer size, depending on the frame length you need to send. (in bytes)
RCVBuffer	IN23	INT	0…4000	Receive buffer size, depending on the frame length you need to receive. (in bytes)
Dummy_I0	IN24	INT	0	Not use
Dummy_W1	IN25	WORD	0	Not use
Dummy_W2	IN26	WORD	0	Not use
Dummy_DW1	IN27	DWORD	0	Not use
RetVal	OUT	WORD		Result of the operation, see chapter 7.2.3

▨ Not needed □ Required

Permitted baud rates: 300[1], 600[1], 1200, 2400, 4800, 9600, 19200[2], 38400[2])

[1]) Not for PCD2.M487
[2]) 38400 for COM1 only or with CDB for COM2..5 (chapter 2.1)

7.2.2 Example

In this example we configure the COM port1 of the PCD.xx7 to use the Transparent mode. Baud rate 9600,8 Data bits, 1 stop bit, no parity. The interface is a RS232. The size of the receiving- and sending buffer is 300 bytes.

```
CALL  SFC  245
       IN0    :=1              // Serial Port N°1
       IN1    :=0              // Transparent
       IN2    :=L#9600         // Baud Rate
       IN3    :=8              // Data Bit
       IN4    :=1              // Stop Bit
       IN5    :=0              // Parity (None)
       IN6    :=0              // RS232 interface
       IN7    :=B#16#0         // Not use
       IN8    :=B#16#0         // Not use
       IN9    :=W#16#0         // Not use
       IN10   :=W#16#0         // Not use
       IN11   :=1              // TelCount
       IN12   :=FALSE          // Can’t Overwrite Frame
       IN13   :=FALSE          // Not use
       IN14   :=FALSE          // Not use
       IN15   :=W#16#0         // Not use
       IN16   :=W#16#0         // Not use
       IN17   :=0              // Not use
       IN18   :=0              // Not use
       IN19   :=0              // Not use
       IN20   :=B#16#0         // Not use
       IN21   :=B#16#0         // Not use
       IN22   :=300            // 300 bytes for the SEND Buffer
       IN23   :=300            // 300 bytes for the RCV Buffer
       IN24   :=0              // Not use
       IN25   :=W#16#0         // Not use
       IN26   :=W#16#0         // Not use
       IN27   :=DW#16#0        // Not use
       RET_VAL:=#RetVal        // RetVal
```

7.2.3 Return value of the SFC245

Value	Description
0	Initialization was done correctly
-1	Not valid COM port number
-2	Not enough S7 memory to create the Buffer, but will be possible if you compress the S7 memory.
-3	Not enough S7 memory to create the Buffer, even if you compress the compress the S7 memory.
-4	Not valid parameter mode
-5	Not valid interface parameters (Baud rate, Data bits, Stop bit or Parity)
-6	Not valid value in WaitSend or WaitInactiv parameter.
-7	Not valid value in TelCount parameter
-8	Not valid value in ZVZ or QVZ parameter
-9	Not valid value in TryToConnect or TryToSend parameter
-10	ASCII – Fixed length: The length of the frame is bigger than the RCV-buffer size
-11	Not valid value in SENDBuffer or RCVBuffer parameter
-12	The total memory of the RCVBuffer and the SENDBuffer is bigger than the 64k Bytes allowed
-13	The SFC was called with the sum of the parameter RCVBuffer and the SENDBuffer different from the first call of the SFC.

Note:

It’s possible to change the driver or the protocol mode of the serial port during execution time, but some rules have to be respected.

1) The buffers size can’t be changed
2) For the PCD to consider the new mode, the SEND or RECEIVE function need a rising edge on their Enable input.

7.3 SEND data

To send a pack of data (byte, character) you will use the SFB12.

7.3.1 Parameter SFB 12

Parameter	Description
REQ	With a positive edge, it will start the SEND procedure.
R	With a positive edge, it will cancel and reset the sending.
ID	Serial COM port number.
R_ID	Not use
DONE	SEND procedure is done, **DONE** stays at the value TRUE one cycle.
ERROR	Error during send operation, **ERROR** stays at the value TRUE one cycle.
STATUS	Error code
SD_1	Area of the data to be SEND, this parameter is an ANY pointer, but the length of the any pointer is not considered here. It will be taken from the LEN parameter.
LEN	Number of byte to be SENT, 4’000 bytes maximum can be sent in one time, of course the SEND buffer size has to be declared equal or higher.

There are also some conditions regarding the state machine of the function, see chapter 1.4.

7.3.2 SEND example

```
CALL  SFB   12 , DB12
       REQ   :=M50.0             // request to send
       R     :=M50.1             // transmit reset
       ID    :=W#16#1            // COM port number
       R_ID  :=                  // not needed
       DONE  :=M100.0            // Transmit is done without error
       ERROR :=M100.1            // Error during transmit
       STATUS:=MW102             // Error code
       SD_1  :=P#DB100.DBX 0.0 BYTE 1000 // Data source
       LEN   :=MW104             // Length of byte to be send
```

7.3.3 Return parameter STATUS

Value	Description
-9 (FFF7h)	Telegram can’t be copied
-8 (FFF8h)	Length is too large
-7 (FFF9h)	Unknown type
-6 (FFFAh)	Invalid destination area
-5 (FFFBh)	Invalid length
-4 (FFFCh)	DB is not loaded
-3 (FFFDh)	Invalid DB number
11	Warning: can’t execute the function, because this one is already being executed.

7.4 RECEIVE a Frame

To read the data incoming on the serial port using the Transparent mode Driver, you will use the SFB 13.

7.4.1 Parameter SFB 13

Parameter	Description
EN_R	True, Enable the receiving
ID	Serial port number
R_ID	Not use
NDR	New Data Ready, new data arrived and were transferred to the desired memory area. **NDR** stays at the value TRUE for one cycle.
ERROR	Error appended in the receiving. **Error** stays at the value TRUE for one cycle.
STATUS	Error code
RD_1	Specifies the area were the incoming data have to be put. This parameter is an ANY pointer. The number of byte specified in this parameter has to be equal or higher to the number of byte received.
LEN	This is an IN/OUT parameter, so the input value can be modified by the function. The value of LEN when calling the function is the number of byte to read from the COM port. The value return in LEN after execution of the function, is the number of byte left in the receive buffer.

There are also some conditions regarding the state machine of the function, see chapter 1.4 .

7.4.2 Example

```
    L       200               // Number of byte to read
    T       MW204

CALL  SFB   13 , DB13
       EN_R  :=M200.0   // Enable the receiving function
       ID    :=W#16#1   // COM port 1
       R_ID  :=         // Not use
       NDR   :=M200.1   // New data arrived flag
       ERROR :=M200.2   // Error Flag
       STATUS:=MW202    // Error code
       RD_1  :=P#DB200.DBX 0.0 BYTE 2000    // Data destination area
       LEN   :=MW204    // Number of byte received
```

7.4.3 Parameter Status

Value	**Description**
-7 (FFF9h)	Invalid type
-6 (FFFAh)	Invalid destination area
-5 (FFFBh)	Invalid length
-4 (FFFCh)	DB is not loaded
-3 (FFFDh)	Invalid DB number
-2 (FFFEh)	COM port not initialized
-1 (FFFFh)	Invalid COM port
0	OK
4	Length too long compare to the buffer size

7.5 Examples

An example has been made to show how work with the 4 possible drivers.
You download this example from the web address www.sbc-support.ch The file name is DOC_CP44.zip .

You need to retrieve it in order to use it. This is done from the SIMATIC Manager -> Menu **FILE** -> **Retrieve** .

In the project you will find two PLC.

PCD_SEND	: This PCD will send data in Transparent mode
PCD_RECEIVE	: This PCD will receive the data.

In order to make this demo work correctly, you need to connect the two COM1 of the PCD together, using a PCD2.F120 module.

One PCD will send Data in Transparent mode and the other will receive them in the same mode.

The ASCII mode is selected in the PCD software on the network 1 of the OB100.

Then in both PCD there is the FC0. There are structured as follow:

SEND_PCD:
Network 1: load length of the frame to send
Network 2: Call the SFB12 (send)
Network 3: Wait 2000 cycle between each SEND of Frame (for the example only)
Network 4: Count the number of sent frames and the number of errors.

RECEIVE_PCD:
Network 1: Call the SFB13 (receive)
Network 2: Count the number of sent frames and the number of errors.

The VAT1: show you how many frames have been SENT or RECEIVED on the PCDs, or also how many errors occurred.