Entry type: **Application example**, Entry ID: **19033929**, Entry date: **06/02/2004**

Transferring data with variable message lengths via the TCP protocol

QUESTION:
How can I receive data with variable message lengths as a TCP data stream?

ANSWER:

Behavior of the TCP protocol:
When data is transferred with the TCP, the transfer is made in the form of a data stream. In this case no information is given about the length nor about the beginning and end and so the recipient cannot tell where one message ends in the data stream and where the next begins. Therefore the sender has to define a **message structure** that can be interpreted by the recipient. The message structure can comprise of the **data plus a terminating control character** such as "carriage return" that indicates the end of the message.

The following sample program includes a TCP connection via which the data can be sent with variable message lengths with FC5 to a station and received with FC6.

Sample program:
Transfer of data (with FC5 "AG_SEND" and FC6 "AG_RECV") with variable message lengths via the TCP protocol

General description:
The STEP 7 project includes two S7-300 stations with CPU 315-2DP and CP 343-1 for communication via Industrial Ethernet. The communication basis is a TCP connection between the two stations. If you open the Properties of the TCP connection in NetPro via "Right- click > Object Properties", you can view the block parameter "ID" of the communication function block. This specification is to be noted when calling FC5 and FC6 so that the data can be transferred via the TCP connection.

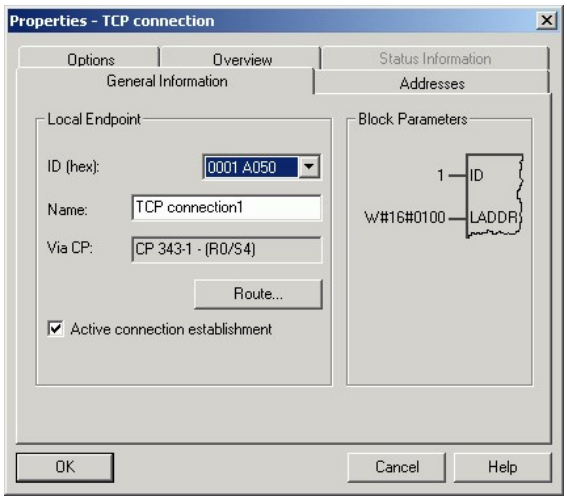


Fig. 1:Properties of the TCP connection

In order to be able to retrieve the messages from the data stream sent the data to be sent must have a specific structure. The message can comprise of the data plus a terminating control character such as "carriage return" so that the recipient can tell where the message ends.

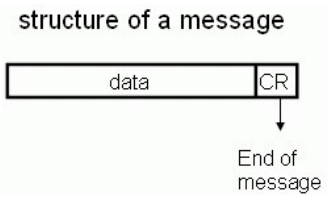


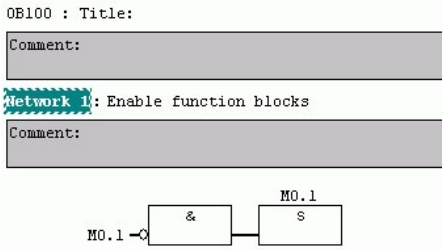
Fig. 2: Structure of a message

Note:
The sample program is based on the message structure shown in Fig. 2, but you can also define your own message structure. If you do want to use a different message structure (e.g. change the end character), then simply change the sample program supplied accordingly to meet your particular requirements.

The data that you want to send with FC5 (AG_SEND) therefore has to be prepared with this predefined structure in a data block (DB221). The data is then receive byte for byte with FC6 (AG_RECV).

Description of the STEP 7 program
The STEP 7 program consists of blocks OB100, OB1, FB100, DB100 (instance DB of FB100), FB102 (multi-instance in DB100), DB221, DB222, FC5, FC6.

- OB100
  OB100 is a startup OB and is run when the CPU is restarted (warm restart). The first communication trigger is issued in this OB with M1.0 and M0.1.



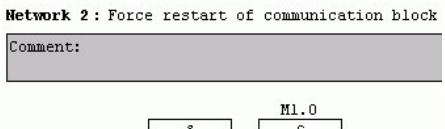


Fig. 3: OB100

- OB1
  OB1 is called cyclically. This OB contains the call of FB100 (instance dB: DB100) with M1.0 and M0.1. Once FB100 has been run M1.0 is reset.

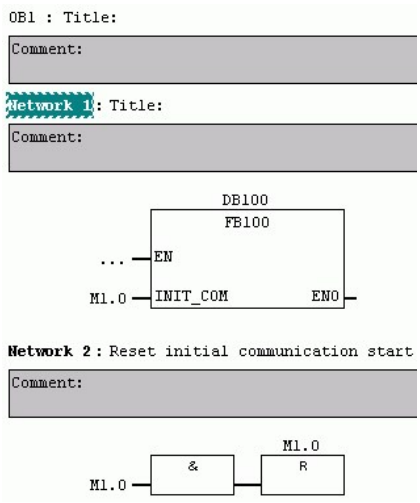


Fig. 4: OB1

- FB100:
  FB100 is called in the OB1 cycle. This FB contains the call of FC5 "AG_SEND" and FB101 "AG_RECV_CR".

  Send block FC5 "AG_SEND"
  FC5 is activated by the input parameter "ACT" when clock marker M10.7 has a rising edge and "SND_BUSY" is not set. "SND_BUSY" is set while the job is running and no function can be triggered during that time.
  This function block is particularly important because the function is asynchronous and can take several cycles. Constant activation of the function without waiting for it to terminate can cause a communication overload.
  The input parameters "ID" and "LADDR" have to taken from the Properties dialog of the TCP connection in NetPro (Fig. 1). In the "SEND" parameter you have to specify the address of the data to be sent (P#DB221.DBX2.0 BYTE 48). For "LEN" you enter the number of bytes to be sent (26). The output parameters "DONE", "ERROR" and "STATUS" are required for job evaluation and are only valid in one and the same cycle.

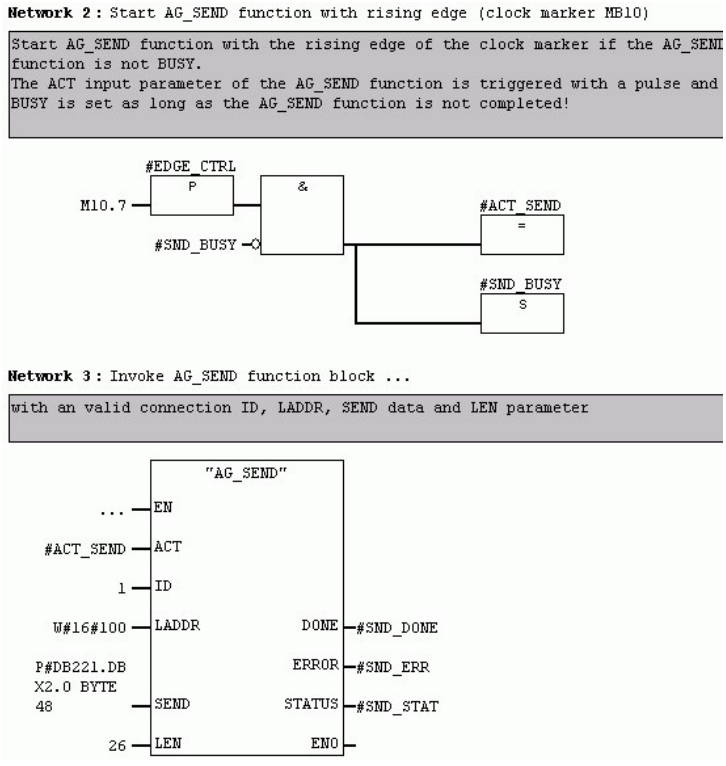
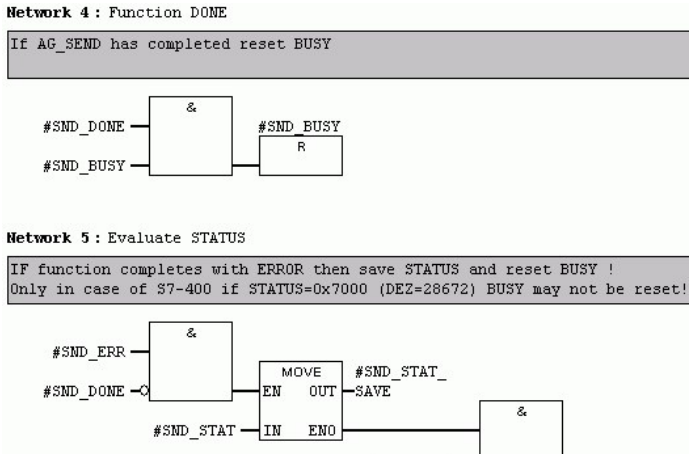


Fig. 5: FB100 - call of FC5

If the block is run through without error, "SND_BUSY" is reset and you can then call FC5 again. If the block is terminated with an error, the status word of the block is saved for error analysis and "SND_BUSY" is also reset.



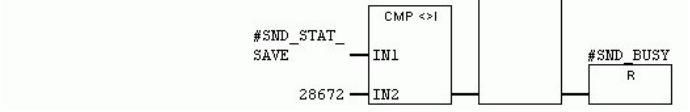


Fig. 6: FB100: evaluation of the FC5 call

Receive block FB102 "AG_RECV_CR"

When you call FB102 "AG_RECV_CR, you must take the parameters "ID" and "LADDR" from the Properties dialog of the connection in NetPro (Fig. 1). In "RECV_BUF" you have to specify the storage location for the data to be received (P#DB222.DBX0.0 BYTE50). The output parameters "NDR", "ERROR" and "STATUS" are required for job evaluation and are only valid in one and the same cycle.

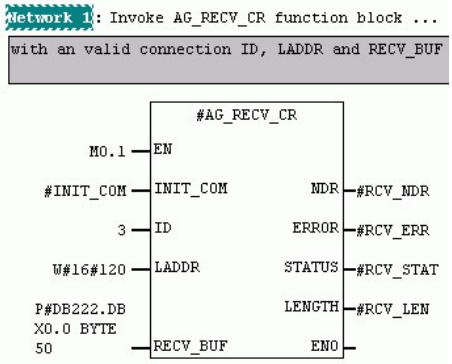


Fig. 7: Call of FB102

If the block is run through without error, the length of the data received is saved. If the block is terminated with an error, the status word of the block is saved for error analysis.

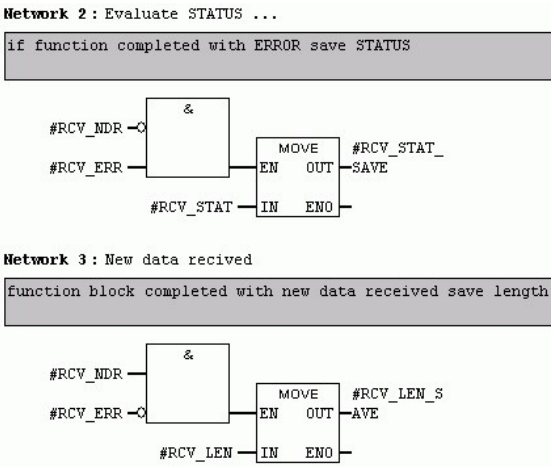


Fig. 8: Evaluation of the FB102 call

- FB102 "AG_RECV_CR"

Initializing:
Initialization steps are executed in Network1 and Network2 of FB102, like resetting the byte counter (#COUNT) to zero and resetting the Error and NDR bit (Out parameter of FB102 "AG_RECV_CR").

Receiving the data:
With FC6 "AG_RECV" the data is received byte for byte. If the function terminates with an error, "#STATUS" is issued as output parameter on FB102 ("AG_RECV_CR").

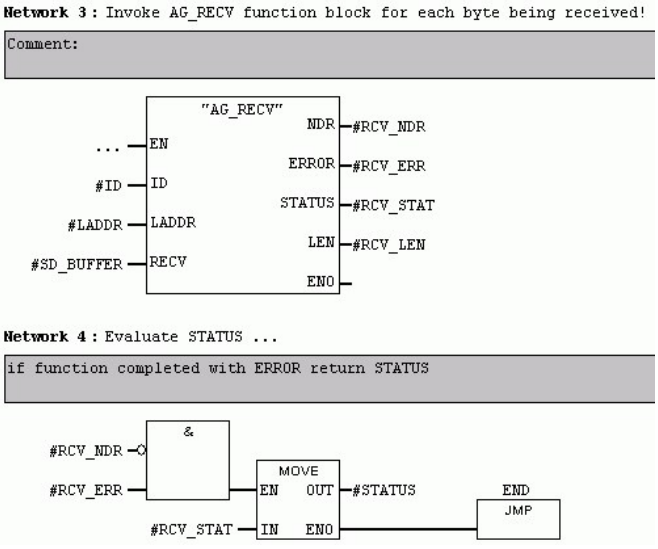


Fig. 9: Receiving data byte for byte

If the data is received without error with FC6, the received byte is saved and the byte offset (#B_OFFSET) is calculated via the "#COUNT" parameter for the ANY Pointer receive area. Then the end character "CR" (carriage return) is checked. The the byte received corresponds to the "carriage return", then the length of the data is forwarded as output parameter "#LENGTH" and the byte counter "#COUNT" is set to zero.

Fig. 10: Determining the byte offset and checking for CR

Creating the receive buffer
The receive buffer (ANY Pointer) specified when FB101 is called is verified with the byte offset determined and saved in a temporary ANY Pointer (VAR_BUF). The byte received is written to the receive buffer via the "BLKMOV" function.

Network 8 : Build ANY-Pointer for DB

Comment:

```
//-----------------------------------------
// allow multiply instances
//-----------------------------------------

      TAR2
      UD    DW#16#FFFFFF
      LAR1  P##RECV_BUF
      +AR1
//-----------------------------------------
// copy ANY-pointer to TEMP BUFFER
//-----------------------------------------

      L     W [AR1,P#0.0]
      T     LW     0

      L     B [AR1,P#2.0]
      T     LB     2
      L     B [AR1,P#3.0]
      T     LB     3
      L     W [AR1,P#4.0]
      T     LW     4
      L     W [AR1,P#6.0]
      T     LW     6
      L     W [AR1,P#8.0]
      L     #B_OFFSET
      +I
      T     LW     8
```

Network 9 : Write next byte into the buffer

Comment:

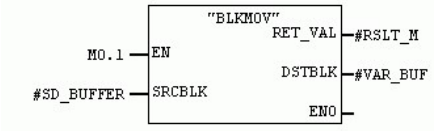


Fig. 11: Creating the receive buffer and saving the byte

If an error occurs while writing the byte to the receive buffer, this is indicated in FB102 via the "#STATUS" and the "#ERROR" bit.

Network 10 : Error occured

Write STATUS=0x80B1 -> Target buffer invalid or not sufficient space!

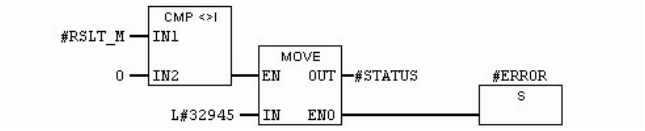


Fig. 12: Error evaluation

- DB221:
  DB221 contains the data structures that are sent with FC5.

Fig. 13: Data structures

- DB222:
  The data received with FC6 is saved in DB222.

The download contains the sample program described. It has been created with STEP 7 V5.3 and SIMATIC NET V6.2.

T_cp_var_cr.exe

Copy the "T_cp_var_cr.exe" file into a separate directory and then start the file with a double-click. The STEP 7 project unpacks automatically with all the subdirectories. You can then use the **SIMATIC Manager** to open and process the project.

Security information

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept. For more information about industrial security, please visit

http://www.siemens.com/industrialsecurity.

› Entry belongs to product tree folder(s):

› This entry is associated to 3 product(s).

Show/Hide picture 1
Show/Hide picture 2
Show/Hide picture 3
Show/Hide picture 4
Show/Hide picture 5
Show/Hide picture 6
Show/Hide picture 7
Show/Hide picture 8
Show/Hide picture 9
Show/Hide picture 10
Show/Hide picture 11