Application Note

ASCII

Data Flow Control in the cyclic Process Data

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC100213AN04EN | Revision 4 | English | 2015-01 | Released | Public

Table of Contents

1 Introduction

1.1 Introduction to this manual

This manual describes the data flow control between a PLC and a gateway to send and receive ASCII data via the gateway using a serial connection.

The ASCII protocol for serial data transfer is used on netTAP NT 50, netTAP NT 100 and netBRICK NB 100 gateway devices.

1.1.1 List of revisions

Index	Date	Chapter	Revision
1	2010-02-19	All	Created
2	2010-03-18	2.5	Description expanded in section *Initialization of the ASCII communication*: Pair of bits for send respectively receive an ASCII telegram and *Relationship between the used operating mode and the used* pairs of bits
3	2011-05-30	1.1.2	Section *Reference to firmware and software*: netTAP NT 50 and netBRICK NB 100 added.
4	2015-01-25	All	Completely revised.

Table 1: List of revisions

1.1.2 Reference to firmware and software

Firmware

Firmware File	Firmware Version
N5ASCxxx.NXF repectively N5xxxASC.NXF netTAP NT 50 firmware with ASCII protocol	from 1.0.x.x
NTxxxASC.NXF netTAP NT 100 firmware with ASCII protocol	from 1.3.x.x
NBASCxxx.NXF netBRICK NB 100 Firmware with ASCII protocol	from 1.4.x.x

Table 2: Reference to firmware

Software

Software	Software Version
SYCONnet netX setup.exe	from 1.310.x.x

Table 3: Reference to software

1.1.3 Conventions in this manual

Operation instructions, a result of an operation step or notes are marked as follows:

Operation Instructions:

➢ <instruction>

Or

1. <instruction>
2. <instruction>

Results:

➩ <result>

Notes:

Important: <important note>

Note: <note>

<note, where to find further information>

1.2 Legal notes

1.2.1 Copyright

1.2.2 Important notes

The user manual, accompanying texts and the documentation were created for the use of the products by qualified experts, however, errors cannot be ruled out. For this reason, no guarantee can be made and neither juristic responsibility for erroneous information nor any liability can be assumed. Descriptions, accompanying texts and documentation included in the user manual do not present a guarantee nor any information about proper use as stipulated in the contract or a warranted feature. It cannot be ruled out that the user manual, the accompanying texts and the documentation do not correspond exactly to the described features, standards or other data of the delivered product. No warranty or guarantee regarding the correctness or accuracy of the information is assumed.

We reserve the right to change our products and their specification as well as related user manuals, accompanying texts and documentation at all times and without advance notice, without obligation to report the change. Changes will be included in future manuals and do not constitute any obligations. There is no entitlement to revisions of delivered documents. The manual delivered with the product applies.

Hilscher Gesellschaft für Systemautomation mbH is not liable under any circumstances for direct, indirect, incidental or follow-on damage or loss of earnings resulting from the use of the information contained in this publication.

1.2.3 Exclusion of liability

The software was produced and tested with utmost care by Hilscher Gesellschaft für Systemautomation mbH and is made available as is. No warranty can be assumed for the performance and flawlessness of the software for all usage conditions and cases and for the results produced when utilized by the user. Liability for any damages that may result from the use of the hardware or software or related documents, is limited to cases of intent or grossly negligent violation of significant contractual obligations. Indemnity claims for the violation of significant contractual obligations are limited to damages that are foreseeable and typical for this type of contract.

It is strictly prohibited to use the software in the following areas:

- for military purposes or in weapon systems;
- for the design, construction, maintenance or operation of nuclear facilities;
- in air traffic control systems, air traffic or air traffic communication systems;
- in life support systems;
- in systems in which failures in the software could lead to personal injury or injuries leading to death.

We inform you that the software was not developed for use in dangerous environments requiring fail-proof control mechanisms. Use of the software in such an environment occurs at your own risk. No liability is assumed for damages or losses due to unauthorized use.

1.2.4 Warranty

Although the hardware and software was developed with utmost care and tested intensively, Hilscher Gesellschaft für Systemautomation mbH does not guarantee its suitability for any purpose not confirmed in writing. It cannot be guaranteed that the hardware and software will meet your requirements, that the use of the software operates without interruption and that the software is free of errors. No guarantee is made regarding infringements, violations of patents, rights of ownership or the freedom from interference by third parties. No additional guarantees or assurances are made regarding marketability, freedom of defect of title, integration or usability for certain purposes unless they are required in accordance with the law and cannot be limited. Warranty claims are limited to the right to claim rectification.

1.2.5 Export regulations

The delivered product (including the technical data) is subject to export or import laws as well as the associated regulations of different counters, in particular those of Germany and the USA. The software may not be exported to countries where this is prohibited by the United States Export Administration Act and its additional provisions. You are obligated to comply with the regulations at your personal responsibility. We wish to inform you that you may require permission from state authorities to export, re-export or import the product.

1.2.6 Registered trademarks

Windows® 2000, Windows® XP, Windows® Vista and Windows® 7 are registered trademarks of Microsoft Corporation.

All other mentioned trademarks are property of their respective legal owners.

2 ASCII data flow control with handshake

2.1 Cyclic process data and acyclic ASCII data

The superordinated control unit (PLC) communicates with the netTAP and netBRICK gateway devices via an I/O protocol (PROFIBUS, PROFINET, DeviceNet etc.) and exchanges cyclically process input data and process output data with these devices.

However, ASCII devices operate as a rule acyclically, which means command oriented and provide data on request only or when data changes.

A control mechanism is necessary to link the cyclic and acyclic communication types together in order to transfer data consistently and reliable between the systems. The control mechanism (handshake for data flow control) is embedded in the cyclic process input data and process output data by the gateway device. This means that the gateway device and the PLC use special data structures inside the cyclic data to build up an acyclic working data channel to the ASCII device(s).

2.2 ASCII telegram structure and SYCON.net

The cyclic I/O protocol to the PLC is standardized by specifications. The protocol sequence of ASCII devices vary greatly. Therefore, the structure of the ASCII protocol has to be configured using SYCON.net configuration software during commissioning. Properties like data direction, start character, end character, data count and timeout values etc. are defined during the device configuration.

Definition and configuration of the ASCII telegram structure is not part of this document. At this point we assume that the ASCII telegram structure is already configured correctly in the gateway device and telegrams can be sent to and received from the ASCII device.

This document describes the transfer (data flow control with handshake) of the ASCII data received and sent only from and to the PLC within the cyclic process data.

2.3 Functional units in the gateway

The gateway has two functional units internally used for communication. The first unit executes the cyclic protocol and its data. The second unit executes the data of the acyclic ASCII protocol.

Both functional units are link together in the gateway by a third functional unit via I/O data buffers. Data coming from the PLC (named output as a rule) are copied into the OUT buffer of the ASCII protocol unit. In the opposite direction, the ASCII protocol unit writes data into the IN buffer and the third unit copies them from this IN butter into OUT buffer of the OUT buffer of the cyclic protocol unit. Data are sent to the PLC (named input data as a rule) from the OUT buffer of the cyclic protocol unit. Hence, the PLC and the ASCII protocol unit have a strong linkage and can exchange data.

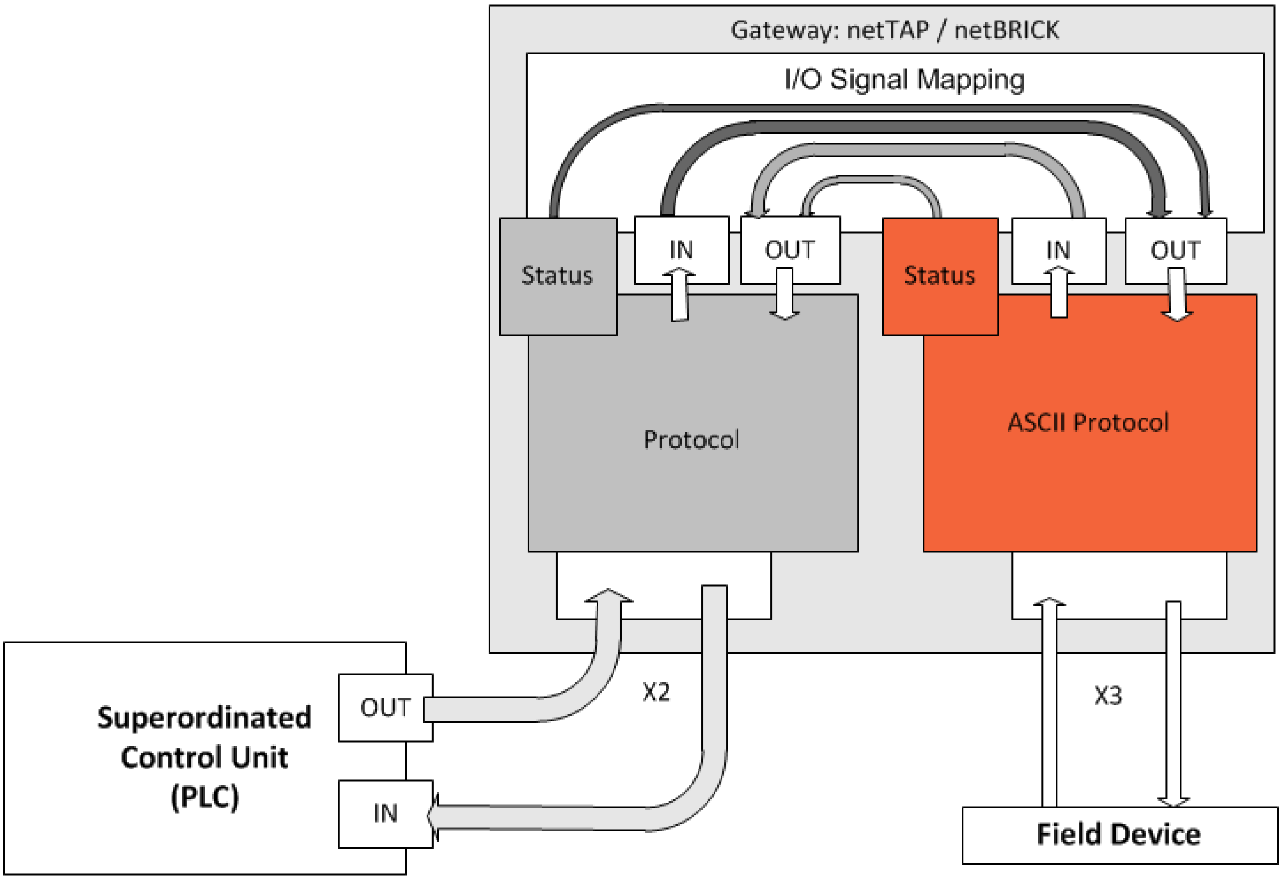

Figure 1: Principal mapping of the data between the protocols

2.4 Data transfer between PLC and ASCII

The transfer mechanism integrated in the cyclic process data contains for each data direction of at least one status-/synchronization register (the handshake flag), a field for the data count, two error registers and the ASCII user data.

2.4.1 Data structure (OUT) – PLC writes

Data type	Signal	Signal for mapping in SYCON.net
32 bit (4 byte)	Output synchronization register Structure, see Table 7: Send synchronization register on page 13.	“Application handshake flags“
32 bit (4 byte)	Amount of ASCII user data counted in bytes to be sent	“Byte count of OutData”
8 bit array[0 … 511] or 8 bit array[0 … 1023]	ASCII user data to be sent	“OutData.UNSIGNED8_0000” “OutData.UNSIGNED8_0001” … “OutData.UNSIGNED8_0511” … “OutData.UNSIGNED8_1023”

Table 4: Data transfer structure: PLC writes ASCII data

2.4.2 Data sructure (IN) – PLC reads

Data type	Signal	Signal for mapping in SYCON.net
32 bit (4 byte)	Input synchronization register Structure, see Table 8: Receive synchronization register on page 15.	“Protocol handshake flags”
32 bit (4 byte)	Amount of ASCII user data received counted in bytes to be receved from the PLC	“Byte count of InData”
32 bit (4 byte)	Error register about receive errors. Error codes are described in section *Error codes* on page 20.	“Error code in case of receive error”
32 bit (4 byte)	Error register about transmit errors. Error codes are described in section *Error codes* on page 20	“Error code in case of transmit error”
8 bit array[0 … 511] or 8 bit array[0 … 1023]	ASCII user data received	“InData.UNSIGNED8_0000” “InData.UNSIGNED8_0001” … “InData.UNSIGNED8_0511” … “InData.UNSIGNED8_1023”

Table 5: Data transfer structure: PLC reads ASCII data

2.5 Initialization of the ASCII communication

The data transfer is between the control unit (PLC) and the ASCII protocol unit is organized by synchronization registers within the I/O data of the cyclic process data.

The basic idea of this method is: for each triggered action, a pair of bits is used in both synchronization registers. The state of the specifies when an action is triggered and when the action is finished. Several actions can be triggered simultaneously using several pairs of bits.

One bit of a bit pair is used to request an action (command CMD) and the other bit of this bit pair is used to acknowledge the action (acknowledge ACK). The CMD bit is always located in the synchronization register written by the PLC, the ACK bit is always located in the synchronization register read by the PLC. On the basis of the state of both bits of a bit pair, the PLC can detect weather and action is currently running of is the action already is acknowledged. It is essential: one action is requested by setting the CMD bit unequal to the ACK bit. The other side acknowledges this request by setting the ACK bit equal to the CMD bit. Both bits have the state 0 and 0 after a reset, an action is trigged then by inverting the CMD bit. The bits have the state 1 and 0 and are unequal and an action is active. Is the action finished, the ACK bit has to be made equal and the state of the bits changes to 1 and 1. Both bits are equal and the action is finished.

Sending ASCII data

One pair of bits is used to send an ASCII data. The PLC controls that an ASCII telegram has to be send. The bits of this pair of bits have the name APP_HS_TX_CMD and PROT_HS_TX_ACK and are described in the following sections:

- APP_HS_TX_CMD is described in Table 7: Send synchronization register on page 13
- PROT_HS_TX_ACK is described in Table 8: Receive synchronization register on page 15
- The usage is described in section Sending ASCII data on page 17

Receiving ASCII data

An other pair of bits is used to receive an ASCII data. The PLC can detect that an ASCII telegram was received. The bits of this pair of bits have the name APP_HS_RX_ACK and PROT_HS_RX_CMD and are described in the following sections:

- APP_HS_RX_ACK is described in Table 7: Send synchronization register on page 13
- PROT_HS_RX_CMD is described in Table 8: Receive synchronization register on page 15
- The usage is described in section Receiving ASCII data on page 19

Four ASCII operating modes

One operation mode has to be set by the configuration software SYCON.net for the ASCII protocol unit. The ASCII protocol unit offers four operating modes.

The following table shows the relationship between the four operating modes and the used pair of bits, which have to be operated or monitored by the PLC.

Operating Mode	Pair of bits	Description
'Receive Only' Mode	APP_HS_TX_CMD and PROT_HS_TX_ACK	Sending ASCII data on page 17
'Send Only' Mode	APP_HS_RX_ACK and PROT_HS_RX_CMD	Receiving ASCII data on page 19
Client Mode (first send, then receive)	APP_HS_TX_CMD and PROT_HS_TX_ACK APP_HS_RX_ACK and PROT_HS_RX_CMD	Sending ASCII data on page 17 Receiving ASCII data on page 19
Server Mode (first receive, then send)	APP_HS_RX_ACK and PROT_HS_RX_CMD APP_HS_TX_CMD and PROT_HS_TX_ACK	Receiving ASCII data on page 19 Sending ASCII data on page 17

Table 6: Relationship between the used operating mode and the used pairs of bits

2.5.1 Structure of the synchronization register in the I/O data

2.5.1.1 Send synchronization register of the PLC

Structure of the send synchronization register (coming from the PLC, going to the ASCII protocol unit).

Bit no.	Name and description
0	**APP_HS_TX_CMD** The PLC has to change this bit as soon as a new send command has to be activated. Before the PLC changes this bit, the PLC has to write the ASCII user data into the send user data structure. The ASCII protocol unit checks this bit continuously and activates exactly one send command unit this command is acknowledged. The ASCII protocol unit changes the **PROT_HS_TX_ACK** bit as soon as the ASCII user data are sent to acknowledge this command.
1	**APP_HS_RX_ACK** The PLC has to change this bit as soon as the receive command of the ASCII protocol unit was finished by the PLC (acknowledge from the PLC). Before the PLC changes this bit, the PLC has to read and evaluate the ASCII user data within the receive data structure. The ASCII protocol unit checks this bit continuously. Not before the last receive command is acknowledged by the PLC, the ASCII protocol unit then provides the next ASCII receive data within the receive data structure and indicated this with the **PROT_HS_RX_CMD** bit in the receive synchronization register.
2 … 5	Do not use. Reserved.
6	**APP_HS_TX_ENABLE_CMD** The PLC has to set this bit as soon as the send mode of the ASCII protocol unit has to be activated and switched on. The ASCII protocol unit acknowledges the enabling of the send mode by setting the **PROT_HS_TX_ENABLE_ACK** bit within the receive synchronization register. Clearing this bit deactivates the send mode.
7	**APP_HS_RX_ENABLE_CMD** The PLC has to set this bit as soon as the receive mode of the ASCII protocol unit has to be activated and switched on. The ASCII protocol unit acknowledges the enabling of the receive mode by setting the **PROT_HS_RX_ENABLE_ACK** bit within the receive synchronization register. Clearing this bit deactivates the receive mode.
8 … 31	Do not use. Reserved.

Table 7: Send synchronization register

Note: The ASCII protocol unit activates the send and receive mode only, if the **APP_HS_TX_ENABLE_CMD** bit and the **APP_HS_RX_ENABLE_CMD** bit are set. The PLC has to set both bits initially and leave both bits set in order ASCII communication can take place.

2.5.1.2 Receive synchronization register of the PLC

Structure of the receive synchronization register (coming from th ASCII protocol unit, going to the PLC).

Bit no.	Name and description
0	**PROT_HS_TX_ACK** The ASCII protocol unit changes this bit as soon as a send command from the PLC is finished by the ASCII protocol unit. The ASCII protocol unit acknowledges the send command. Before the ASCII protocol unit changes this bit the ASCII protocol unit has read the ASCII user data from the receive structure and has been sent the data. The PLC has to check this bit continuously. Not before the last send command is acknowledged by the ASCII protocol unit, the PLC then can provide the next ASCII send data within the send data structure and indicated this with the **PROT_HS_TX_CMD** bit in the send synchronization register.
1	**PROT_HS_RX_CMD** The ASCII protocol unit changes this bit as soon as new reveice command has to be indicated. Before the ASCII protocol unit changes this bit, the ASCII protocol unit has written the ASCII user data into the receive user data structure. The PLC has to check this bit continuously. Exactly one receive command is active until it is acknowledged. As soon as the PLC has evaluated the ASCII user data, the PLC has to change the **PROT_HS_RX_ACK** bit in the send synchronization register to acknowledge this command.
2	Do not use. Reserved.
3	**PROT_HS_RUN_IND** The ASCII protocol unit signals 'Ready' and the signals the end of its initialization. This bit indicates the PLC in general and at any time that the ASCII protocol unit is ready and active.
4	**PROT_HS_TX_ERROR_IND** The ASCII protocol unit sets this bit if a send error was detected. Together with this bit an error code is provided in the “Error code in case of send error” register. The error codes are described in section *Error codes* on page 20. The ASCII protocol unit clears this bit as soon as a as the next telegram was sent without error.
5	**PROT_HS_RX_ERROR_IND** The ASCII protocol unit sets this bit if a receive error was detected. Together with this bit an error code is provided in the “Error code in case of receive error” register. The error codes are described in section *Error codes* on page 20. The ASCII protocol unit clears this bit as soon as a as the next telegram was received without error.
6	**PROT_HS_TX_ENABLE_ACK** The ASCII protocol unit set this bit to indicate that the send mode is active. The PLC has to activate the send mode by setting the **APP_HS_TX_ENABLE_CMD** bit in the send synchronization register. If the bit is cleared the send mode is disabled.

Bit no.	Name and description
7	**PROT_HS_RX_ENABLE_ACK** The ASCII protocol unit set this bit to indicate that the receive mode is active. The PLC has to activate the receive mode by setting the **APP_HS_RX_ENABLE_CMD** bit in the send synchronization register. If the bit is cleared the receive mode is disabled.
8 … 31	Do not use. Reserved.

Table 8: Receive synchronization register

2.5.2 Initializing of the communication between the PLC and the ASCII protocol unit

This section describes the values of the synchronization register and the sequence directly after a device reset.

Note: In the following tables an "x" marks an undefined bit position and an "X" marks a defined but not relevant bit position.

Step	Action: **Start of the communication, initialization is done by the PLC.**	Send synchronization register (bits 0-7 only) 7 6 5 4 3 2 1 0	Receive synchronization register (bits 0-7 only) 7 6 5 4 3 2 1 0
0	Situation directly after a device reset. The ASCII protocol unit signals 'not ready'.	0 0 x x x x 0 0	0 0 0 0 0 0 0 0
1	The ASCII protocol unit signals 'ready' after finish of the device initialization. The PLC has to check weather this bit is set and then can activate further actions.	0 0 x x x x 0 0	0 0 0 0 1 0 0 0
2	The PLC activates the send and receive mode of the ASCII protocol unit by setting bit 6 and 7.	1 1 x x x x 0 0	0 0 0 0 1 0 0 0
3	The ASCII protocol unit acknowledges that send and receive mode is active. The communication from and to the PLC can start.		1 1 0 0 1 0 0 0

Table 9: ASCII – Initializing of the communication

Note: The communication can start any time after bit 6 and 7 is set (in the receive synchronization register).

2.5.3 Normal operating mode between PLC and ASCII protocol unit

This section describes the values of the synchronization register for send and receive mode. The initialization is already finished.

Note: In the following tables an "x" marks an undefined bit position and an "X" marks a defined but not relevant bit position.

2.5.3.1 Sending ASCII data

Step	Action: The PLC sends ASCII data	Send synchronization register (bits 0-7 only) 7 6 5 4 3 2 1 0	Receive synchronization register (bits 0-7 only) 7 6 5 4 3 2 1 0
0	Status of the send and receive synchronization register after initialization is finished. Bit 0 of both registers have value 0 → No send command is active and the PLC can activate a command.	1 1 x x x x X **0**	1 1 x x 1 x X **0**
1	The PLC writes the ASCII data to be sent and the user data count into the send data structure. The PLC inverts the value of bit 0 to activate the send command by changing the value of bit 0 from 0 to 1.	1 1 x x x x X **1**	
2	The ASCII protocol unit detects that bit 0 is unequal (value of bit 0 in the send synchronization register is 1 and value of bit 0 in receive synchronization register is 0). A send command is active. The ASCII protocol unit reads the data count and the ASCII send data from the send data structure and sends the data.	1 1 x x x x X **1**	1 1 x x 1 x X **0**
3	After the ASCII data are sent from the ASCII protocol unit, the ASCII protocol unit inverts the value of bit 0 in the receive synchronization register to make it the same as value of bit 0 in the send synchronization register. Both bits have value 1 now and thus are equal. The last send command is finished and acknowledged. No send command is active.		1 1 x x 1 x X **1**
4	The PLC detects the change of the value in bit 0 in the receive synchronization register which means the send command is finished and acknowledged. New ASCII send data can be transferred to the ASCII protocol unit.	1 1 x x x x X **1**	1 1 x x 1 x X **1**
5	New ASCII data are to be sent. The PLC writes the ASCII data to be sent and the user data count into the send data structure. The PLC inverts the value of bit 0 to activate the send command by changing the value of bit 0 from 1 to 0.	1 1 x x x x X **0**	
6	The ASCII protocol unit detects that bit 0 is unequal (value of bit 0 in the send synchronization register is 0 and value of bit 0 in receive synchronization register is 1). A send command is active. The ASCII protocol unit reads the data count and the ASCII send data from the send data structure and sends the data.	1 1 x x x x X **0**	1 1 x x 1 x X **1**
7	After the ASCII data are sent from the ASCII protocol unit, the ASCII protocol unit inverts the value of bit 0 in the receive synchronization register to make it the same as value of bit 0 in the send synchronization register. Both bits have value 0 now and thus are equal. The last send command is finished and acknowledged. No send command is active.		1 1 x x 1 x X **0**

Step	Action: The PLC sends ASCII data	Send synchronization register (bits 0-7 only) 7 6 5 4 3 2 1 0	Receive synchronization register (bits 0-7 only) 7 6 5 4 3 2 1 0
8	The PLC detects the change of the value in bit 0 in the receive synchronization register which means the send command is finished and acknowledged. New ASCII send data can be transferred to the ASCII protocol unit.	1 1 x x x x X **0**	1 1 x x 1 x X **0**

Table 10: Sequence in the synchronization register for sending ASCII data

2.5.3.2 Receiving ASCII data

Step	**Action:** **The PLC receives ASCII data**	**Send synchronization register (bits 0-7 only)** 7 6 5 4 3 2 1 0	**Receive synchronization register (bits 0-7 only)** 7 6 5 4 3 2 1 0
0	Status of the send and receive synchronization register after initialization is finished. Bit 1 of both registers have value 0 → No receive command is active and the ASCII protocol unit can activate a command.	1 1 x x x x **0** X	1 1 x x 1 x **0** X
1	The ASCII protocol unit has received data and has written these data and the data count into the receive data structure. The ASCII protocol unit inverts the value of bit 1 to activate the receive command by changing the value of bit 1 from 0 to 1.		1 1 x x 1 x **1** X
2	The PLC detects that value of bit 1 of the receive synchronization register and value of bit 1 in the send synchronization register is unequal. A receive command is active. The PLC reads the data count and the ASCII receive data from the receive data structure and the received data and evaluates them.	1 1 x x x x **0** X	1 1 x x 1 x **1** X
3	After the PLC has evaluated the received ASCII data, the PLC inverts the value of bit 1 in the send synchronization register to make it the same as value of bit 1 in the receive synchronization register. Both bits have value 1 now and thus are equal. The last receive command is finished and acknowledged. No receive command is active.	1 1 x x x x **1** X	
4	The ASCII protocol unit detects the change of the value in bit 1 in the send synchronization register which means the receive command is finished and acknowledged. New ASCII receive data can be transferred to the PLC.	1 1 x x x x **1** X	1 1 x x 1 x **1** X
5	The ASCII protocol unit has received new ASCII data. The ASCII protocol unit writes the ASCII data to be received from the PLC and the user data count into the receive data structure. The ASCII protocol unit inverts the value of bit 1 to activate the receive command by changing the value of bit 1 from 1 to 0.		1 1 x x 1 x **0** X
6	The PLC detects that bit 1 is unequal (value of bit 1 in the send synchronization register is 0 and value of bit 1 in receive synchronization register is 1). A (new) receive command is active. The PLC reads the data count and the ASCII receive data from the receive data structure and the received data and evaluates them.	1 1 x x x x **1** X	1 1 x x 1 x **0** X
7	After the PLC has evaluated the received ASCII data, the PLC inverts the value of bit 1 in the send synchronization register to make it the same as value of bit 1 in the receive synchronization register. Both bits have value 0 now and thus are equal. The last receive command is finished and acknowledged. No receive command is active.	1 1 x x x x **0** X	
8	The ASCII protocol unit detects the change of the value in bit 1 in the send synchronization register which means the receive command is finished and acknowledged. New ASCII receive data can be transferred to the PLC.	1 1 x x x x **0** X	1 1 x x 1 x **0** X

Table 11: Sequence in the synchronization register for receiving ASCII data

3 Error codes

Error codes	Name and description
00000000 hex 0 dez	TLR_S_OK No error
C0000007 hex 3221225479 dez	TLR_E_INVALID_PACKET_LEN Invalid packet length
C0000180 hex 3221225857 dez	TLR_E_BUS_OFF The ASCII stack has been stopped by the host application.
C07E 0001 hex 3229483009 dez	TLR_E_ASCII_COMMAND_INVALID Invalid command received.
C07E0002 hex 3229483010 dez	TLR_E_ASCII_STACK_DATA_SIZE_INVALID Invalid data length
C07E 0003 hex 3229483011 dez	TLR_E_ASCII_STACK_BUSY The serial line is busy. Any sending or receiving is in progress
C07E0004 hex 3229483012 dez	TLR_E_ASCII_STACK_PACKET_TOO_LONG ASCII sends: Requested packet is too long for sending ASCII receives: The received telegram is longer than expected.
C07E0005 hex 3229483013 dez	TLR_E_ASCII_STACK_DATA_OVERLAPPED ASCII receives. A new telegram has been received via the serial line before the previous was acknowledged by the superordinated control unit. The previous data in the internal buffer of the ASCII protocol has been overwritten.
C07E0006 hex 3229483014 dez	TLR_E_ASCII_STACK_RESPONSE_TIMEOUT Answer time limit has been exceeded.
C07E0007 hex 3229483015 dez	TLR_E_ASCII_STACK_WAITING_RESPONSE The ASCII protocol expects a response via the serial line, e. g. a receive of data from the serial interface is expected.
C07E0008 hex 3229483016 dez	TLR_E_ASCII_STACK_LED_NOT_SUPPORTED LED not supported
C07E0009 hex 3229483017 dez	TLR_E_ASCII_STACK_MSG_MODE_DISABLED Message mode is deactivated

Table 12: Error codes of the ASCII protocol unit

4 Appendix

4.1 List of Figures

4.2 List of Tables

4.3 Contacts

Headquarters

Germany
Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-Mail: info@hilscher.com

Support
Phone: +49 (0) 6190 9907-99
E-Mail: de.support@hilscher.com

Subsidiaries

China
Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-Mail: info@hilscher.cn

Support
Phone: +86 (0) 21-6355-5161
E-Mail: cn.support@hilscher.com

France
Hilscher France S.a.r.l.
69500 Bron
Phone: +33 (0) 4 72 37 98 40
E-Mail: info@hilscher.fr

Support
Phone: +33 (0) 4 72 37 98 40
E-Mail: fr.support@hilscher.com

India
Hilscher India Pvt. Ltd.
New Delhi - 110 025
Phone: +91 11 40515640
E-Mail: info@hilscher.in

Italy
Hilscher Italia srl
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-Mail: info@hilscher.it

Support
Phone: +39 02 25007068
E-Mail: it.support@hilscher.com

Japan
Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-Mail: info@hilscher.jp

Support
Phone: +81 (0) 3-5362-0521
E-Mail: jp.support@hilscher.com

Korea
Hilscher Korea Inc.
Suwon, 443-734
Phone: +82 (0) 31-695-5515
E-Mail: info@hilscher.kr

Switzerland
Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-Mail: info@hilscher.ch

Support
Phone: +49 (0) 6190 9907-99
E-Mail: ch.support@hilscher.com

USA
Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-Mail: info@hilscher.us

Support
Phone: +1 630-505-5301
E-Mail: us.support@hilscher.com