Description Data Exchange Controller – PC

General

Each command is transmitted as a single entity from PC to controller.
A command starts with <STX>, the controller answers within less than 100 ms with <DLE> or <NAK>.
In case of <NAK> or no answer the computer has to repeat the start procedure with <STX>.
In case of <DLE> the computer sends the data block to be transmitted, followed by a check byte. The check byte represents an XOR link of all data bytes sent. This check byte is followed by <DLE>. The data block is then terminated by <ETX>.

The controller answers with <ACK> when no error is detected. I f the command is not recognized the controller sends <NAK>. This exception has to be handled by the calling PC program.

The data block always consists of

- the device number (#n)
- a command, a request, or a message

Commands are led by (!), requests are led by (?), messages are led by (:)
Data blocks do neither contain spaces nor control codes. Valid characters are ASCII characters from 21h to 7Eh. <ESC> is an exception. It is sent to terminate a previous command currently executed.
Any other character within a data block is interpreted by the controller as an error and will result in sending a <NAK>.

There are limited commands (single steps) and commands with no time limit (all others). Non-limited means it cannot be predicted when execution of the command is finished.

If the end position of an axis is reached, a non-limited command will be terminated. In case of the HOME command the controller waits for the HOME RETURN command (!HR)

Available Baud Rates

.....

Table

.....

Examples

Example 1:

Drive device 1 to position + 1234,49

Computer	Controller	
<STX>		StartOfText
	<DLE>	DataLinkEscape
#1		Device 1
!GF+01.234,49		GotoAbsPos +1.234,49 (fast speed)
(BCC)		Hardware BCC → Test byte
<DLE>		DataLinkEscape
	#1	Device 1
	:M	Motor of device 1 is active
	(BCC)	Hardware BCC → Test byte
	<DLE>	DataLinkEscape
	<ETX>	EndOfText

Number Values

# (n)	Device number
(Step value)	(<sign><Full steps><Micro steps>) → -30.000,00 to +30.000,00 The relation between full step/micro step is determined by the thread slope of the spindle used.

(Key)	Number of key on keyboard (1 ... 8)
(Device)	Number of device (motor; 1 ... 8)

Messages

Controller → computer

:E+	:E-	Mechanical end position reached
:H+	:H-	Home function active
:M		Motor is active
:<ESC>		Feed back: previous command interrupted

Error Codes

F01	-	Invalid character detected
F02	-	BCC not correct
F03	-	Device number not valid
F04	-	Command string for reset not correct
F05	-	Command string for single step not correct
F06	-	Command string for fast speed not correct
F07	-	Command string for slow speed not correct
F08	-	Command string for Home not correct
F09	-	Command string for key lock/unlock not correct
F0A	-	Command string for Piezo on/off not correct
F0B	-	Command string for device on/off not correct
F0C	-	Command string for relative distance not correct
F0D	-	Command string for absolute distance not correct
F0E	-	Command code not recognized
F0F	-	Command code not recognized
F10	-	String for request not correct
F11	-	"!" or "?" missing
F12	-	"#" missing
F13	-	Digit expected at this position
F14	-	ditto
F15	-	"+" or "-" missing
F16	-	Decimal point missing
F17	-	Value larger than 30000.00
F18 .... F1D		Internal communication codes of controller (to be neglected)

Explanation of BCC and ACK/NAK

(BCC)	Hardware BCC (BlockCheckCharacter) XOR link of all data block characters (without <STX>): Both the sent and received BCCs consist of two bytes: (HiNibble (BCC) + 30h) and (LowNibble (BCC) +30h)
<ACK/NAK>	Acknowledge: no error detected NotAcknowledge: error detected

Status request

On a status request the controller answers with one data block. According to the current situation (busy/idle) this data block is variable and is not predictable in length. The controlling computer software must scan these data and evaluate the information therein. Since the state of the manipulation system is highly variable neither the position nor the occurrence of certain status information can be guaranteed.

Examples for Status data blocks

#1: E+L+P+01.234,49	Device 1: End position (clock wise) reached / keys of keypad locked / current position: +1234,49
#3: H-L-MP+12.345,49	Device 3: Home function (counter clock wise) in progress / keys unlocked / Motor is running / current position: +12345,49

More Examples

Example 2: Drive Device 5 to Home Position CW (clock wise)

Computer	Controller	
<STX>		StartOfText
	<DLE>	DataLinkEscape
#5		Device 5
!H+		Go to home CW
(BCC)		Hardware BCC
<DLE>		DataLinkExchange
<ETX>		EndOfText
	<ACK>	Acknowledge
	<STX>	StartOfText
<DLE>		DataLinkExchange
	#5	Device 5
	:M	Motor is running
	(BCC)	Hardware BCC
	<DLE>	DataLinkExchange
	<ETX>	EndOfText
<ACK>		Acknowledge

Example 2: Get Current Position of Device 3

Computer	Controller	
<STX>		StartOfText
	<DLE>	DataLinkEscape
#3		Device 3
?P		Get Position
(BCC)		Hardware BCC
<DLE>		DataLinkExchange
<ETX>		EndOfText
	<ACK>	Acknowledge
	<STX>	StartOfText
<DLE>		DataLinkExchange
	#3	Device 3
	:P+00012,34	Position of device 3: 12,34
	(BCC)	Hardware BCC
	<DLE>	DataLinkExchange
	<ETX>	EndOfText
<ACK>		Acknowledge

More Info about Commands

Status Information

The computer can get status information or position of any device at any time. The reply contains all information about the actual state of this particular device.

Timeout

The controller monitors all incoming bytes using a serial timeout. This time is 100 ms, i.e. unless the next byte is received within 100 ms the data received so far are discarded, and the controller waits for the next command starting with <STX>.

Length of Commands

An entire command may not exceed 24 bytes. This means the 24th byte at last has to be <ETX>, otherwise the command is not recognized. The controller will then wait for the next command starting with <STX>.

Interface Parameters

9 different Baud rates can be set: 38400; 19200; 9600; 4800; 2400; 1200; 600; 300; 110 Baud.
Parity can be set odd or even.
There are 8 data bits and 1 stop bit (cannot be changed).

Command Parameter List

!F+	Fast move CW (clockwise)
!F-	Fast move CCW (counterclockwise)
!S+	Slow move CW (clockwise)
!S-	Slow move CCW (counterclockwise)
!E+	Single step CW (clockwise)
!E-	Single step CCW (counterclockwise)
!A	Stop movement
!H+	Home CW (clockwise)
!H-	Home CCW (counterclockwise)
!HR	Home return
!GF (number of steps)	GotoAbsPos; fast move
!GS (number of steps)	GotoAbsPos; slow move
!EF (number of steps)	RelDistance; fast move
!ES (number of steps)	RelDistance; slow move
!@S	Reset counter
!L+	Lock keypad
!L-	Unlock keypad
!V+	Switch on current for device
!V-	Switch off current for device
!<ESC>	Interrupt function in process
!Z+	Step output high
!Z-	Step output low
!O (Sub speed)	Sub speed; fast move
!U (Sub speed)	Sub speed; slow move
?Z	Get state of device
?P	Get position of device

Calculate Negative Number of Steps

1

2: yy <=0 calculation finished
3: yy >=0 increment number of full steps by 1
4: calculate micro steps

$$50 - \frac{yy}{2} * 100 = \text{(number of micro steps)}$$

e.g. GotoAbsPos; fast move –2,567 mm

- 2567 µm

---------- = -513,40 full steps, increment by 1: -514 full steps

5 µm

$$50 - \frac{40}{2} * 100 = 30 \text{ micro steps}$$

Additional Commands:

Setting ramp-length:

Command: #N!RU
Parameter: <00000..65535> [5byte]
Remark: It is possible to set the length of the start- and stop-ramp (only both together).
The appropriate command is: #N!RU followed by the ramp-length in milliseconds.
The minimal length is approximately 200 milliseconds und the maximum length is 65535 milliseconds.
Example:

Set ramp-length of device 3 to 1200 milliseconds

computer	controller	
<STX>		StartOfText
	<DLE>	DataLinkEscape
#3		Device 3
!RU 01200		Ramp-Length is 1200 milliseconds
(BCC)		Hardware BCC → Test byte
<DLE>		DataLinkEscape
	#3	Device 3
	:M	Motor of device 1 is active
	(BCC)	Hardware BCC → Test byte
	<DLE>	DataLinkEscape
	<ETX>	EndOfText

Set positioning velocity slow:

Command: #N!UX

Parameter: micro steps per second (150 to approximately 20000)
Remark: Resolution of micro steps is 256.
e.g.: You choose 12800 microsteps per second. As one fullstep is 256 microsteps and one turn is 100 fullsteps you got 0.5 rps.

Set positioning velocity high:

Command: #N!OX

Parameter: Fullsteps per second

Remark: One fullstep is 256 micro Steps.
There are approximately 150 different possible velocities. The absolute value (in rps) from each of them depends on the shape of the acceleration-ramp and the achievable maximal velocity. The system will automatically choose the closest match.
e.g.: You choose 1500 fullsteps/second. As one turn is 100 fullsteps you got 15 rps.
Possible values with this ramp are: 14.85 rps and 15.1 rps so the system will choose 15.1 rps cause it's closest to 15 rps.
Again: Absolute values depend on the used ramp and can't be predicted.

Goto absolute position variable fast:

Command: #N!GX

Parameter: same as GotoAbsPosition: <sign><fullsteps><ustep>

Remark: The preset velocity from ‘Set positioning velocity high’ will be used

Goto relative position variable fast:

Command: #N!DX

Parameter: same as GotoRelativePosition: <sign><fullsteps><ustep>

Remark: The preset velocity from ‘Set positioning velocity high’ will be used

Goto absolute position variable slow:

Command: #N!GY

Parameter: same as GotoAbsPosition <sign><fullsteps><ustep>

Remark: The preset velocity from ‘Set positioning velocity slow’ will be used

Goto relative position variable slow:

Command: #N!DY

Parameter: same as GotoRelativePosition <sign><fullsteps><ustep>

Remark: The preset velocity from ‘Set positioning velocity slow’ will be used