

mechOnics ag

CF30 Programming Interface

USER'S GUIDE

Table of Contents

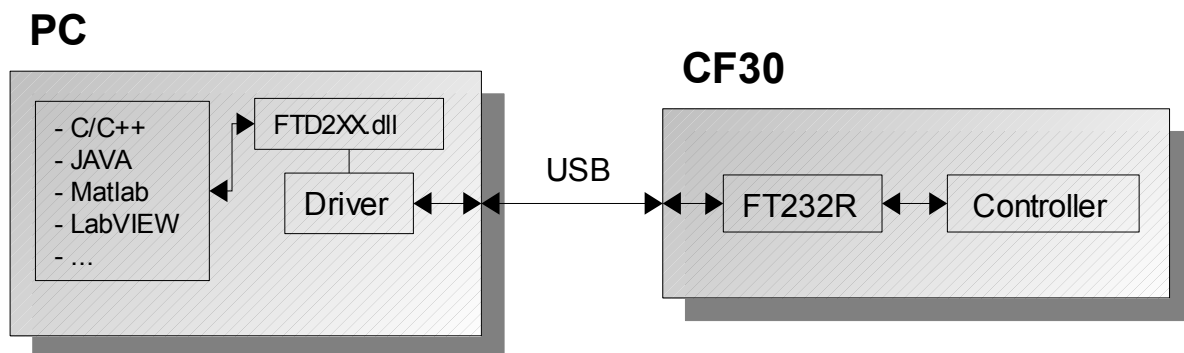
1 INTRODUCTION.....	3
2 INSTALLATION.....	4
3 INTERFACE FOR YOUR OWN CONTROL SOFTWARE.....	5
3.1 LOAD THE FTD2XX.DLL (EXPLICIT OR IMPLICIT)	5
3.2 SCAN FOR CONNECTED DEVICES.....	5
3.3 OPEN THE DEVICE.....	5
3.4 CONFIGURE THE DEVICE.....	5
3.5 SEND AN ASCII COMMAND.....	6
3.6 ASCII COMMANDS.....	6
4 DATA FORMAT OF THE ASCII COMMANDS.....	7
4.1 SPEED COMMAND: PV.....	7
4.2 SINGLE STEP COMMAND: PS.....	8
4.3 MOVE RELATIVE TO CURRENT POSITION : PM.....	9
4.4 MOVE TO ABSOLUTE POSITION : PA.....	10
4.5 CHANGE THE RAMP VALUE (SET RAMP): SR.....	11
4.6 SET COUNTER: SC.....	12
4.7 STOP FAST: SF.....	13
4.8 READ THE RAMP VALUE (GET RAMP): GR.....	14
4.9 GET COUNTER: GC.....	15
5 BLOCK DIAGRAM OF THE INTERFACE	16
6 DATA FORMAT OF THE BINARY COMMANDS.....	17
7 THE BIT SHIFTING SCHEME.....	18
7.1 BINARY COMMANDS OVERVIEW.....	20
7.2 BINARY SINGLE STEP COMMANDS.....	21
7.3 BINARY SPEED COMMAND.....	22
7.4 BINARY MOVE RELATIVE TO CURRENT POSITION.....	24
7.5 BINARY MOVE TO ABSOLUTE POSITION.....	26
7.6 BINARY GET COUNTER AND VELOCITY.....	28

1 Introduction

The CF30 controller USB interface is realized using the FT232R USB interface chip of the company FTDI. (www.ftdichip.com)

The interface uses:

1. ASCII string commands.
2. Commands in binary format.



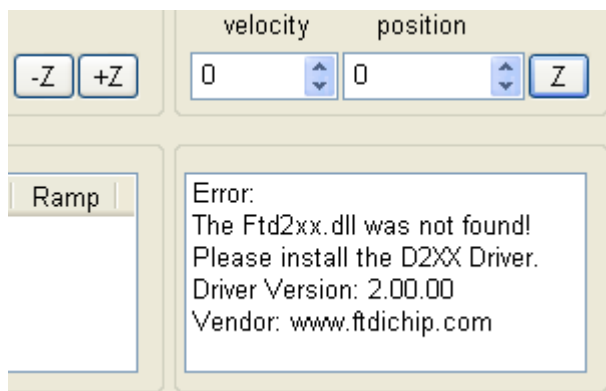
Main Features:

- Interface driver for Windows, Linux ...
- Simple ASCII string commands.
- Binary command. (faster)
- Virtual COM ports.
- Read command with timeout (non blocking mode)
- Search commands for FTDI chips.
- EEPROM Programming Interface Functions.
- Parallel connections to more than one device.

2 Installation

First you have to install the driver that isn't part of the CF30setup.msi windows installer. FTDI provides two types of drivers for Windows: a D2XX direct driver and a virtual COM port (VCP) driver. The combined driver model (CDM) installs both parts at the same time. An application accesses the FTDI devices through the ftd2xx.dll. Download the zip file (CDM 2.00.00.zip) if this file is not part of the software CD distributed with the controller and follow the windows installation instruction. The installer registers the self-registering ftd2xx.dll. At least driver version 2.00.00 is necessary.

- 1 Plug in the CF30 controller.
- 2 Install the FTDI driver for the FT232R interface chip.
Version 2.00.00 or newer.
- 3 Install the CF30 application (not necessary).



If there is no valid FTDI Driver on your PC, the “CF30.exe” program start with an error message.

3 Interface for your own control software

The “D2XX Programmer's Guide” describes the use of the driver - the open, write and read functions.

This guide describes the use and format of the ASCII string commands.

How to proceed:

3.1 Load the *ftd2xx.dll* (explicit or implicit)

3.2 Scan for connected devices

```
DWORD dwNumDevs;
int iCount;
FT_STATUS ftStatus;
FT_DEVICE_LIST_INFO_NODE *devInfo ;
//
// create the device information list
ftStatus = FT_CreateDeviceInfoList(&dwNumDevs);
if (ftStatus == FT_OK){
    if(dwNumDevs > 0){
        devInfo = new FT_DEVICE_LIST_INFO_NODE[dwNumDevs];
        if(devInfo){
            ftStatus = FT_GetDeviceInfoList(devInfo,&dwNumDevs);
            if (ftStatus == FT_OK){
                for(iCount = 0; iCount < (int) dwNumDevs; iCount++)
                {
                    m_i64ComboBoxDeviceIDs[iCount] =((__int64) devInfo[iCount].SerialNumber;
                }
            }
        }
    }
}
....
```

3.3 Open the device

```
ftStatus = FT_OpenEx(devInfo[iCount].SerialNumber,FT_OPEN_BY_SERIAL_NUMBER,&m_ftHandle);
```

3.4 Configure the device

```
ftStatus = FT_SetDivisor(m_ftHandle,5); // 600000 bit/s
ftStatus = FT_SetDataCharacteristics(m_ftHandle,FT_BITS_8,FT_STOP_BITS_1,FT_PARITY_NONE);
ftStatus = FT_ResetDevice(m_ftHandle);
ftStatus = FT_SetFlowControl(m_ftHandle,FT_FLOW_DTR_DSR,'0','1');
ftStatus = FT_SetTimeouts(m_ftHandle,100,100);

// Baud rate serial = 3 MHz / divisor
// 3000000/5 = 600000 bits/s
```

Baud rate: Only 600000 bit/s is a valid value.

Flow control: DTR/DSR handshaking

3.5 Send an ASCII command

```
int iVel ; // [-32767...32767]
char szBufferWrite[40];
char szBufferRead [40];
StringCbPrintfA(szBufferWrite,sizeof(szBufferWrite),"PV,1,%d\r", iVel );

ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );// _TX - out // _RX - in
ftstatus = FT_Write(m_ftHandle,szBufferWrite, (DWORD) strlen(szBufferWrite),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,12,&dwBytesReturned);
```

3.6 ASCII commands

Single step axis = 1 , direction: positive:

```
ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );// _TX - out // _RX - in
ftstatus = FT_Write(m_ftHandle,"PS,1,1\r",(DWORD) strlen("PS,1,1\r"),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,12,&dwBytesReturned);
```

Single step axis = 1 , direction: negative:

```
ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );// _TX - out // _RX - in
ftstatus = FT_Write(m_ftHandle,"PS,1,0\r",(DWORD) strlen("PS,1,0\r"),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,12,&dwBytesReturned);
```

Set ramp value (SR) :

```
StringCbPrintfA(szBufferWrite,sizeof(szBufferWrite),"SR,1,%d\r", (BYTE) dwSteady);
ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );// _TX - out // _RX - in
ftstatus = FT_Write(m_ftHandle,szBufferWrite,(DWORD) strlen(szBufferWrite),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,12,&dwBytesReturned);
```

ramp value: [0..32]

close device:

```
ftstatus = FT_Close (m_ftHandle);
```

Please note:

The ASCII string isn't a "null-terminated string" the last character is "carriage return" CR. Each instruction causes an answer from the CF30 controller. For each type of command the length of the answer is constant . Don't send new commands before the answer of the previous command has arrived. The close command "FT_Close(m_ftHandle)" doesn't stop any movement this command only close an open device. The FT232R uses flow control. (DTR/DSR)

4 Data format of the ASCII commands

4.1 Speed command: PV

position	0	1	2	3	4	5..10	11
character	P	V	,	x	,	yyyyyy	CR
ASCII (dec)	80	86	44		44		13

x – axis [1..3]

y – velocity [-32767...32767] ; velocity = 0 stops the movement.

CR – carriage return

Example:

character	P	V	,	1	,	0	\r					
ASCII (Dec)	80	86	44	49	44	48	13					
character	P	V	,	1	,	-	3	2	7	6	7	\r
ASCII (Dec)	80	86	44	49	44	45	51	50	55	54	55	13
character	P	V	,	1	,	3	2	7	6	7	\r	
ASCII (Dec)	80	86	44	49	44	51	50	55	54	55	13	

Send a speed command with velocity = 0 to stop the movement, or send the stop command.

The PV command starts the movement without timeout. Use it with care. The movement doesn't stop automatically. The FT_Close() function doesn't stop the movement.

C/C++:

```
StringCbPrintA(szBufferWrite,sizeof(szBufferWrite),"PV,1,%d\r", iVel );  
ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );  
ftstatus = FT_Write(m_ftHandle,szBufferWrite,(DWORD) strlen(szBufferWrite),&dwBytesWritten);  
ftstatus = FT_Read (m_ftHandle,szBufferRead,12,&dwBytesReturned);
```

response format:

position	0	1	2	3	4	5..11	11
character	P	V		x		00000 ¹	CR
ASCII (dec)	80	86	32		32		13

¹ Velocity: [-32767...+32767]: 00100, -00100, 00000, -32767, 32767. The value has leading zeros.

4.2 Single step command: PS

position	0	1	2	3	4	5	6
character	P	S	,	x	,	y	CR
ASCII (dec)	80	83	44		44		13

x – axis [1..3]

y – direction :

[1 positive direction]

[0 negative direction]

CR – carriage return

Example:

character	P	S	,	1	,	1	\r					
ASCII (Dec)	80	83	44	49	44	49	13					
character	P	S	,	1	,	0	\r					
ASCII (Dec)	80	83	44	49	44	48	13					
character	P	S	,	2	,	1	\r					
ASCII (Dec)	80	83	44	49	44	48	13					

After sixty-four single steps there is one full step done and the position will be held without voltage.

C/C++:

```
ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX); // _TX - Out // _RX - In
ftstatus = FT_Write(m_ftHandle,"PS,1,0\r",(DWORD) strlen("PS,1,0\r"),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,12,&dwBytesReturned);
```

response format:

position	0	1	2	3	4	5..10	11
character	P	S		x		00000 ²	CR
ASCII (dec)	80	83	32		32		13

² Velocity, number with leading zeros its a setpoint value

4.3 move relative to current position : PM

position	0	1	2	3	4	5	6..11	7,12..15,20	8..21
character	P	M	,	x	,	yyyyyy	,	zzzzzzzzzz	CR
ASCII (dec)	80	77	44		44				13

x – axis [1..3]
 y – velocity [-32767...32767]
 x – counter [-9999999... 9999999]
 CR – carriage return

Example:

character	P	M	,	1	,	1	0	,	1	0	\r						
ASCII (Dec)	80	77	44	49	44	49	48	44	49	48	13						
character	P	M	,	1	,	-	3	2	7	6	7	,	-	1	0	0	\r
ASCII (Dec)	80	77	44	49	44	45	51	50	55	54	55	44	45	49	48	48	13
character	P	M	,	1	,	3	2	7	6	7	,	1	0	0	\r		
ASCII (Dec)	80	77	44	49	44	51	50	55	54	55	44	49	48	48	13		

C/C++:

```

StringCbPrintfA(szBufferWrite,sizeof(szBufferWrite),"PM,%d,%d,%d\r",axis,(short)iVel,LCnt);
ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );
ftstatus = FT_Write(m_ftHandle,szBufferWrite,(DWORD) strlen(szBufferWrite),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,24,&dwBytesReturned);
  
```

response format:

position	0	1	2	3	4	5..10	11	12..22	23
character	P	M		x		yyyyyy		zzzzzzzzzzzz	CR
ASCII (dec)	80	77	32		32		32		13

4.4 move to absolute position : PA

position	0	1	2	3	4	5	6..11	7,12..15,20	8..21
character	P	A	,	x	,	yyyyyy	,	zzzzzzzzzz	CR
ASCII (dec)	80	65	44		44				13

x – axis [1..3]
 y – velocity [-32767...32767]
 x – counter [-9999999... 9999999]
 CR – carriage return

Example:

character	P	A	,	1	,	1	0	,	1	0	\r						
ASCII (Dec)	80	65	44	49	44	49	48	44	49	48	13						
character	P	A	,	1	,	-	3	2	7	6	7	,	-	1	0	0	\r
ASCII (Dec)	80	65	44	49	44	45	51	50	55	54	55	44	45	49	48	48	13
character	P	A	,	1	,	3	2	7	6	7	,	1	0	0	\r		
ASCII (Dec)	80	65	44	49	44	51	50	55	54	55	44	49	48	48	13		

C/C++:

```

StringCbPrintfA(szBufferWrite,sizeof(szBufferWrite),"PA,%d,%d,%d\r",axis,(short)iVel,LCnt);
ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );// _TX - Out // _RX - In
ftstatus = FT_Write(m_ftHandle,szBufferWrite,(DWORD) strlen(szBufferWrite),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,22,&dwBytesReturned);
  
```

response format:

position	0	1	2	3	4	5..10	11	12..22	23
character	P	A		x		yyyyyy		zzzzzzzzzzzz	CR
ASCII (dec)	80	65	32		32		32		13

4.5 Change the ramp value (set ramp): SR

position	0	1	2	3	4	5	6..8
character	S	R	,	x	,	yyy	CR
ASCII (dec)	83	82	44		44		13

x – axis [1..3]
 y – ramp value [0..255] only values between [0..32] are valid. All values above 32 are set to 32.
 CR – carriage return

Example

character	S	R	,	1	,	0	\r					
ASCII (Dec)	83	82	44	49	44	48	13					
character	S	R	,	1	,	3	2	\r				
ASCII (Dec)	83	82	44	49	44	51	50	13				
character	S	R	,	2	,	2	3	2	\r			
ASCII (Dec)	83	82	44	50	44	50	51	50	13			

C/C++:

```

StringCbPrintfA(szBufferWrite,sizeof(szBufferWrite),"SR,1,%d\r", (BYTE)LCnt);
ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );/
ftstatus = FT_Write(m_ftHandle,szBufferWrite,(DWORD) strlen(szBufferWrite),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,iAnswer12,&dwBytesReturned);
  
```

response format:

position	0	1	2	3	4	5...10	11
character	S	R		x		yyyyyy	CR
ASCII (dec)	83	82	32		32		13

4.6 Set counter: SC

position	0	1	2	3	4	5..13	6..14
character	S	C	,	x	,	zzzzzzzzzz	CR
ASCII (dec)	83	67	44		44		13

x – axis [1..3]
z – counter [-9999999... 9999999]
CR – carriage return

Example:

character	S	C	,	1	,	0	\r									
ASCII (Dec)	83	67	44	49	44	48	13									
character	S	C	,	1	,	3	2	\r								
ASCII (Dec)	83	67	44	49	44	51	50	13								
character	S	C	,	2	,	-	9	2	2	3	2	2	3	2	\r	
ASCII (Dec)	83	67	44	50	44	45	57	50	50	51	50	50	51	50	13	

C/C++;

```
StringCbPrintfA(szBufferWrite,sizeof(szBufferWrite),"SC,1,%d\r", LCnt);
ftstatus = FT_Purge(m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );
ftstatus = FT_Write(m_ftHandle,szBufferWrite,(DWORD) strlen(szBufferWrite),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,15,&dwBytesReturned);
```

response format:

position	0	1	2	3..13	14
character	S	C		yyyyyyyyyyyyyy	CR
ASCII (dec)	83	67	32		13

4.7 Stop fast: SF

position	0	1	2	3	4
character	S	F	,	x	CR
ASCII (dec)	83	70	44		13

x decimal	binary	Axis 1	Axis 2	Axis 3
0	000	-	-	-
1	001	stop	-	-
2	010	-	stop	-
3	011	stop	stop	-
4	100	-	-	stop
5	101	stop	-	stop
6	011	-	stop	stop
7	111	stop	stop	stop

Its possible to stop a single axis or all axes at the same time with one command.

Example:

character	S	F	,	1	\r										
ASCII (Dec)	83	70	44	49	13										
character	S	F	,	3	\r										
ASCII (Dec)	83	70	44	51	13										
character	S	F	,	7	\r										
ASCII (Dec)	83	70	44	55	13										

C/C++;

```
StringCbPrintfA(szBufferWrite,sizeof(szBufferWrite),"SF,7\r");
ftstatus = FT_Purge(m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );
ftstatus = FT_Write(m_ftHandle,szBufferWrite,(DWORD) strlen(szBufferWrite),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,5,&dwBytesReturned);
```

response format:

position	0	1	2	3	4
character	S	F		y	CR
ASCII (dec)	83	70	32		13

4.8 Read the ramp value (get ramp): GR

position	0	1	2	3	4
character	G	R	,	x	CR
ASCII (dec)	71	82	44		13

x – axis [1..3]
CR – carriage return

Example:

character	G	R	,	1	\r
ASCII (Dec)	71	82	44	49	13
character	G	R	,	3	\r
ASCII (Dec)	71	82	44	51	13
character	G	R	,	2	\r
ASCII (Dec)	71	82	44	50	13

C/C++;

```
StringCbPrintfA(szBufferWrite,sizeof(szBufferWrite),"GR,1\r", (BYTE)LCnt);  
ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX );// _TX - Out // _RX - In  
ftstatus = FT_Write(m_ftHandle,szBufferWrite,(DWORD) strlen(szBufferWrite),&dwBytesWritten);  
ftstatus = FT_Read (m_ftHandle,szBufferRead,12,&dwBytesReturned);
```

response format:

position	0	1	2	3	4	5..10	11
character	G	R		x		yyyyyy	CR
ASCII (dec)	71	82	32		32		13

4.9 Get counter: GC

position	0	1	2	3	4
character	G	C	,	x	CR
ASCII (dec)	71	67	44		13

x - axis [1..3]
CR - carriage return

Example:

character	G	C	,	1	\r
ASCII (Dec)	71	67	44	49	13
character	G	C	,	3	\r
ASCII (Dec)	71	67	44	51	13
character	G	C	,	2	\r
ASCII (Dec)	71	67	44	50	13

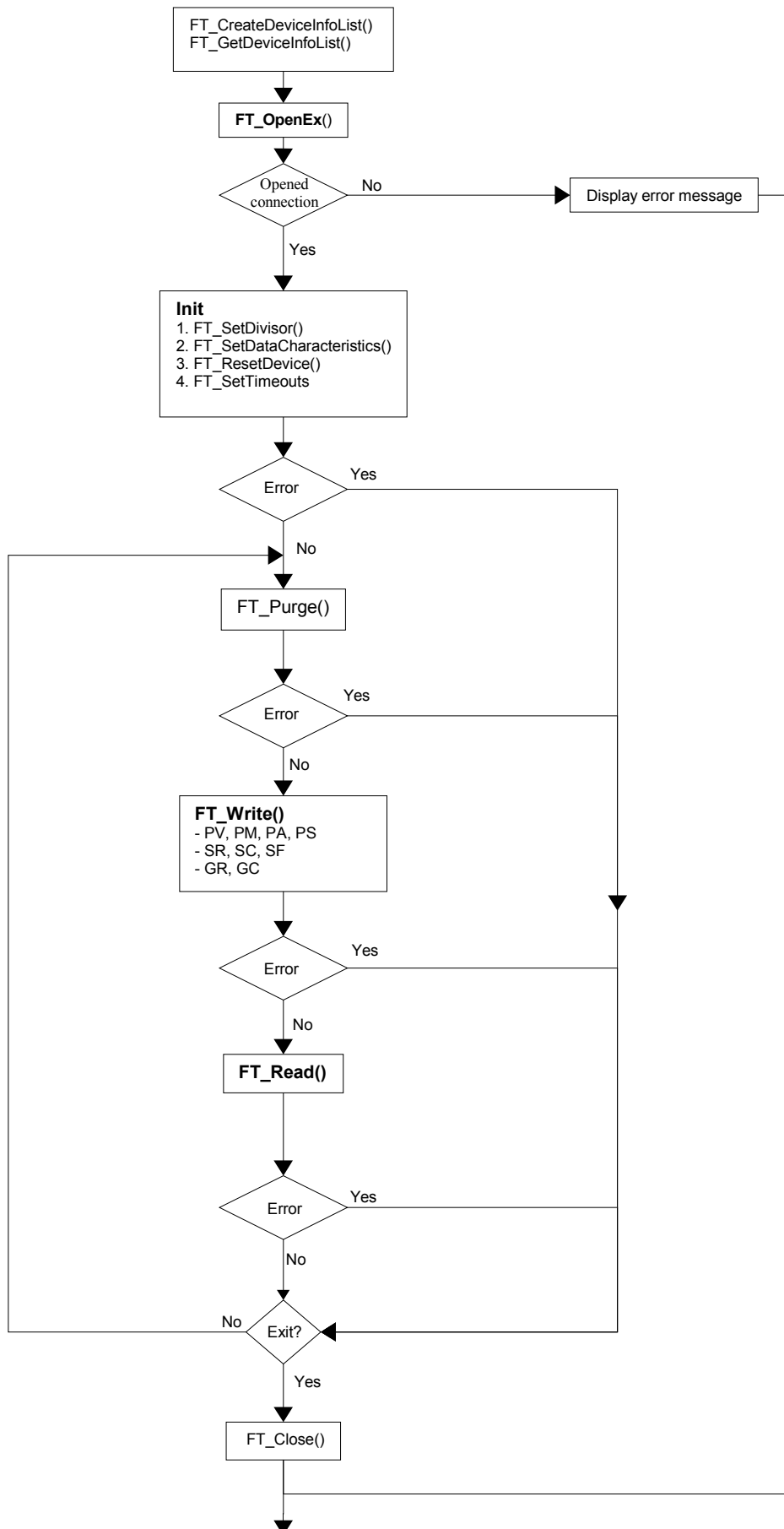
C/C++;

```
ftstatus = FT_Purge (m_ftHandle, FT_PURGE_TX | FT_PURGE_RX); // _TX - Out // _RX - In
ftstatus = FT_Write(m_ftHandle,"GC,1\r",(DWORD) strlen("GC,1\r"),&dwBytesWritten);
ftstatus = FT_Read (m_ftHandle,szBufferRead,17,&dwBytesReturned);
```

response format:

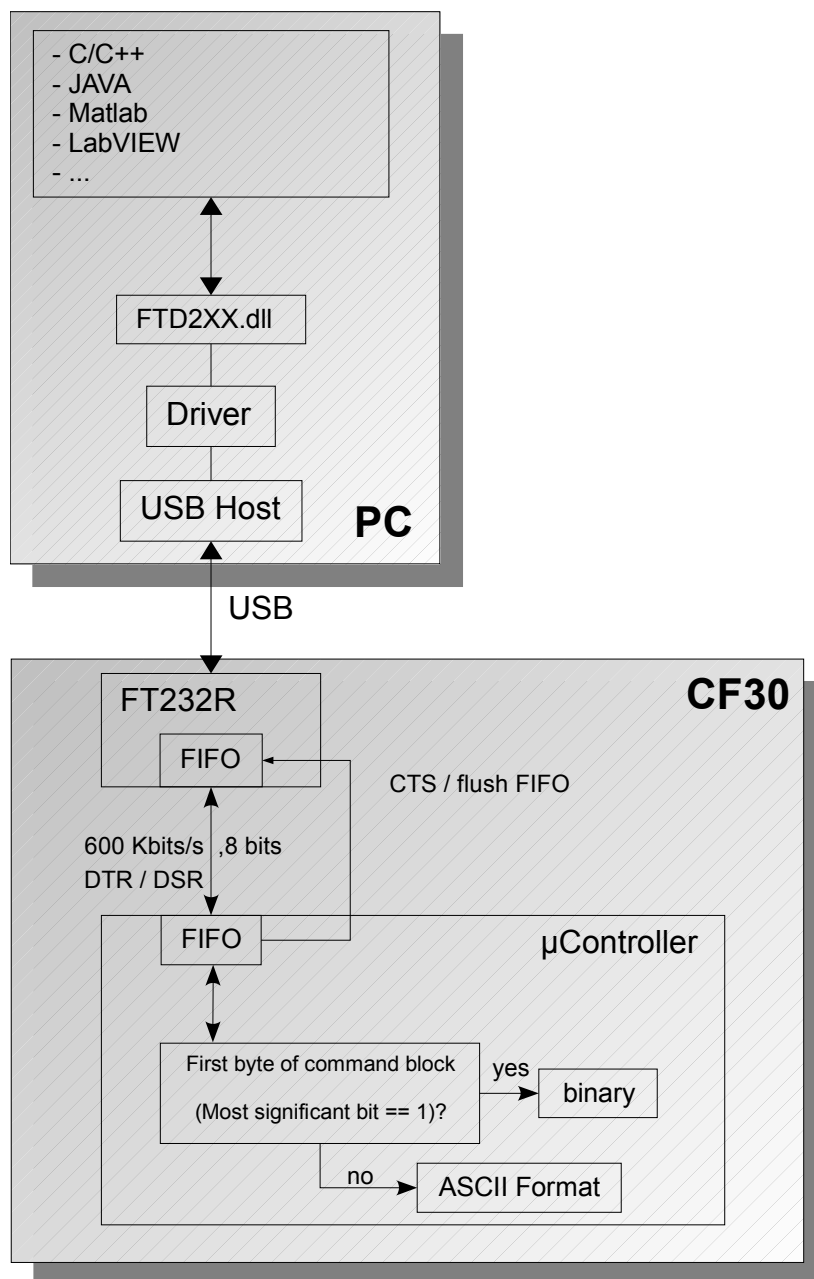
position	0	1	2	3	4	5..15	16
character	G	C		x		yyyyyyyyyyyyyy	CR
ASCII (dec)	71	67	32		32		13

5 Block diagram of the interface



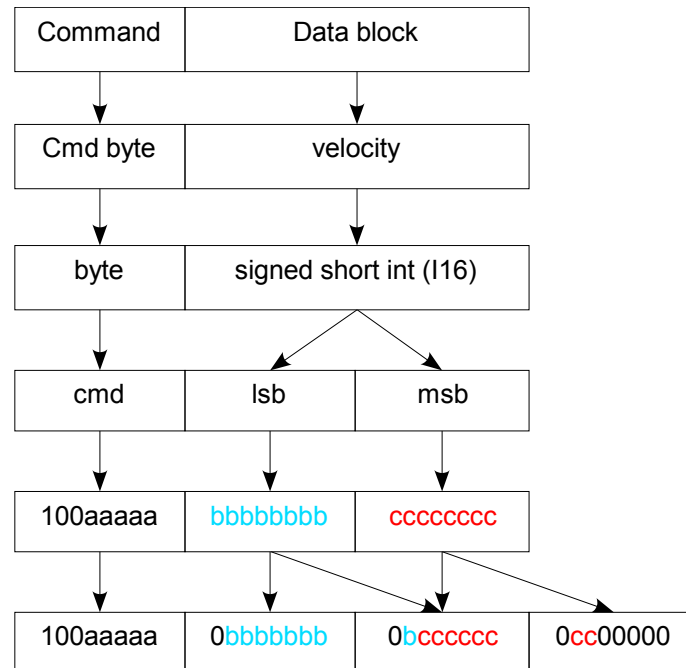
6 Data Format of the binary commands

Binary commands are shorter, therefore the transmitting time is shorter. Additionally the processing time is further minimized. If you use the ASCII Format to send commands to the CF30, only characters are used with binary values are less than 128 and the most significant bit of each byte is not set. If a new block of data reaches the μ controller of the CF30, the program checks the first bit of the first incoming byte. If the most significant bit of a byte is set, the byte is recognised as command byte. In binary mode every data bit could be set or not. We use logical bit shifting and only the seven least significant bits of the data bytes. The most significant bits of the data bytes are cleared (not set). Therefore it is possible to mix binary and ASCII commands. The binary commands are nearly two times faster.



7 The bit shifting scheme

In binary mode there are commands with the length of one, four and eight bytes. The step command is a single byte command. The command with the length of four bytes sends one command byte and three bytes for the velocity value.



C/C++:

```

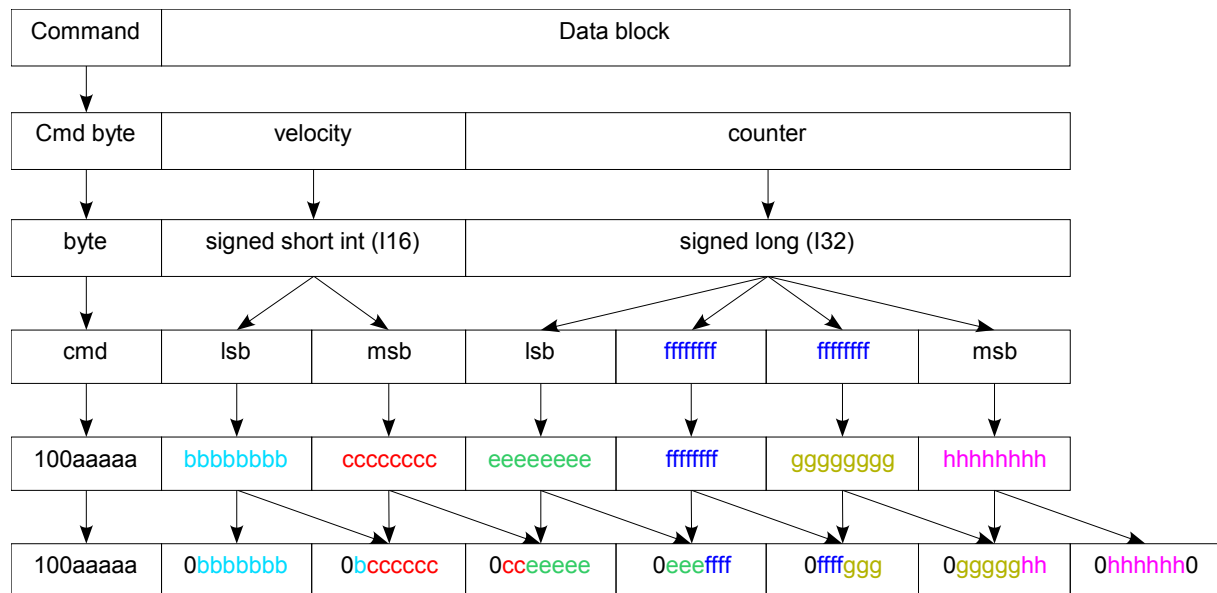
    szBufferWrite[0] = (char) 0x86;
    m_ShiftBufferVelocity((unsigned char*) &szBufferWrite[1], iVel);
    ftstatus = FT_Write(m_ftHandle, szBufferWrite, 4, &dwBytesWritten);

void CF30Dialog::m_ShiftBufferVelocity(unsigned char* buff, int iVel)
{
    short I16Vel;
    I16Vel = (short) iVel;

    buff[0] = * ((char*) &I16Vel);
    buff[1] = * ((char*) &I16Vel + 1);

    buff[3] = 0x00;
    buff[2] = (buff[1] & 0x03) << 5;
    buff[1] = ((buff[1] & 0xFC) >> 2) | ((buff[0] & 0x01) << 6);
    buff[0] = ((buff[0] & 0xFE) >> 1);
}
  
```

The command with the length of eight bytes sends one command byte and seven bytes for the velocity value and the counter value.



C/C++ code sample

```

szBufferWrite[0] = (char) (0x8B + axis);

m_ShiftBufferVelocityAndCounter((unsigned char*) &szBufferWrite[1], iVel, LCnt);

ftstatus = FT_Write(m_ftHandle, szBufferWrite, 8, &dwBytesWritten);

void CF30Dialog::m_ShiftBufferVelocityAndCounter(unsigned char* buff, int iVel, int icnt)
{
    short I16Vel;
    I16Vel = (short) iVel;

    buff[0] = * ((char*) &I16Vel);
    buff[1] = * ((char*) &I16Vel + 1);

    buff[2] = * ((char*) &icnt);
    buff[3] = * ((char*) &icnt + 1);
    buff[4] = * ((char*) &icnt + 2);
    buff[5] = * ((char*) &icnt + 3);

    buff[7] = 0x00;
    buff[6] = ((buff[5] & 0x3f) << 1);
    buff[5] = ((buff[5] & 0xC0) >> 6) | ((buff[4] & 0x1f) << 2);
    buff[4] = ((buff[4] & 0xE0) >> 5) | ((buff[3] & 0x0f) << 3);
    buff[3] = ((buff[3] & 0xF0) >> 4) | ((buff[2] & 0x07) << 4);
    buff[2] = ((buff[2] & 0xF8) >> 3) | ((buff[1] & 0x03) << 5);
    buff[1] = ((buff[1] & 0xFC) >> 2) | ((buff[0] & 0x01) << 6);
    buff[0] = ((buff[0] & 0xFE) >> 1);
}

```

7.1 Binary commands overview:

	cmd byte	data bytes , bit 7 = 0								Nr.	Explanation
PS	10000000								X ₊	0	Single step x axis positive
	10000001								X ₋	1	Single step x axis negativ
	10000010								Y ₊	2	Single step y axis positive
	10000011								Y ₋	3	Single step y axis negativ
	10000100								Z ₊	4	Single step z axis positive
	10000101								Z ₋	5	Single step z axis negativ
PV	10000110	V ₁	V ₂	V ₃					V _x	6	Speed command x axis
	10000111	V ₁	V ₂	V ₃					V _y	7	Speed command y axis
	10001000	V ₁	V ₂	V ₃					V _z	8	Speed command z axis
PM	10001001	V ₁	V ₂	V ₃ /P ₁	P ₂	P ₃	P ₄	P ₅	x	9	Move x axis (relative)
	10001010	V ₁	V ₂	V ₃ /P ₁	P ₂	P ₃	P ₄	P ₅	y	10	Move y axis (relative)
	10001011	V ₁	V ₂	V ₃ /P ₁	P ₂	P ₃	P ₄	P ₅	z	11	Move z axis (relative)
PA	10001100	V ₁	V ₂	V ₃ /P ₁	P ₂	P ₃	P ₄	P ₅	x	12	Move x axis (absolute)
	10001101	V ₁	V ₂	V ₃ /P ₁	P ₂	P ₃	P ₄	P ₅	y	13	Move y axis (absolute)
	10001110	V ₁	V ₂	V ₃ /P ₁	P ₂	P ₃	P ₄	P ₅	z	14	Move z axis (absolute)
GC	10001111								x	15	Get counter & velocity x
	10010000								y	16	Get counter & velocity y
	10010001								z	17	Get counter & velocity z

7.2 Binary single step commands

command byte	hexadecimal		Explanation
10000000	0x80	X ₊	Single step in positive direction (x axis)
10000001	0x81	X ₋	Single step in negative direction (x axis)
10000010	0x82	Y ₊	Single step in positive direction (y axis)
10000011	0x83	Y ₋	Single step in negative direction (y axis)
10000100	0x84	Z ₊	Single step in positive direction (z axis)
10000101	0x85	Z ₋	Single step in negative direction (z axis)

Example:

character	0x80
character	0x82
character	0x83

After sixty-four single steps there is one full step done and the position will be held without voltage.

C/C++:

```
ftstatus = FT_Write(m_ftHandle,"x80",1,&dwBytesWritten);  
ftstatus = FT_Read (m_ftHandle,szBufferRead,1,&dwBytesReturned);
```

response format:

position	0
hexadecimal	0x80
decimal	128

7.3 Binary speed command

command byte	hexadecimal	bits	bits	bits	axis
10000110	0x86	0bbbbbbb	0bcccccc	0cc00000	x
10000111	0x87	0bbbbbbb	0bcccccc	0cc00000	y
10001000	0x88	0bbbbbbb	0bcccccc	0cc00000	z

Example:

axis: x

velocity(dec): 25000

velocity(hex): 0x61A8

velocity(bin): 01100001 10101000

	cmd	lsb	msb	
hexadecimal	0x86	0xA8	0x61	
binary	10000110	01010100	01100001	
shifted	10000110	00101010	00011000	00100000

axis: z

velocity(dec): -25000

velocity(hex): 0x9E58

velocity(bin): 10011110 01011000

(The **two's complement** of a binary number)

	cmd	lsb	msb	
hexadecimal	0x88	0x58	0x9E	
binary	10001000	01011000	10011110	
shifted	10001000	00101100	00100111	01000000

axis: y

velocity(dec): -1

velocity(hex): 0xFFFF

velocity(bin): 11111111 11111111

(The **two's complement** of a binary number)

	cmd	lsb	msb	
hexadecimal	0x87	0xFF	0xFF	
binary	10000111	11111111	11111111	
Shifted (bin)	10000111	01111111	01111111	01100000
Shifted (hex)	0x87	0x7F	0x7F	0x60

C/C++:

```
ftstatus = FT_Write(m_ftHandle, "\x87\x7f\x7f\x60", 4, &dwBytesWritten);  
ftstatus = FT_Read (m_ftHandle, szBufferRead, 5, &dwBytesReturned);
```

response format:

position	0	1	2	3	4
hexadecimal	0x87	0x01	0x02	0x03	0x00
hexadecimal	0x87	0x00030201 (I32)			
decimal	cmd	197121 (dec)			
decimal	cmd	counter/position value			

Remarks:

The bytes of the response aren't shifted. The first byte is the command byte and the next bytes are four bytes of an unsigned integer value, the least significant byte was sent first.

7.4 Binary move relative to current position

cmd byte	(hex)	bits	bits	bits	bits	bits	bits	bits	axis
10001001	0x89	0bbbbbbb	0bcccccc	0ceeeeee	0eeefffff	0ffffggg	0ggggggh	0hhhhh0	x
10001010	0x8A	0bbbbbbb	0bcccccc	0ceeeeee	0eeefffff	0ffffggg	0ggggggh	0hhhhh0	y
10001011	0x8B	0bbbbbbb	0bcccccc	0ceeeeee	0eeefffff	0ffffggg	0ggggggh	0hhhhh0	z

Example:

axis: x

velocity(dec): 25000

velocity(hex): 0x61A8

velocity(bin): 01100001 10101000

delta position(dec): 12000

delta position(hex): 0x00002EE0

delta position(bin): 00000000 00000000 00101110 11100000

	cmd	lsb	msb	lsb			msb	
hex	0x86	0xA8	0x61	0xE0	0x2E	0x00	0x00	
bin	10000110	01010100	01100001	11100000	00101110	00000000	00000000	
shifted	10000110	00101010	00011000	00111100	00000010	01110000	00000000	00000000

axis: z

velocity(dec): -25000

velocity(hex): 0x9E58

velocity(bin): 10011110 01011000

delta position(dec): -12000

delta position(hex): 0xFFFFD120 (The **two's complement** of a binary number)

delta position(bin): 11111111 11111111 11010001 00100000

	cmd	lsb	msb	lsb			msb	
hex	0x8B	0x58	0x9E	0x20	0xD1	0x00	0x00	
bin	10001011	01011000	10011110	00100000	11010001	11111111	11111111	
shifted	10001011	00101100	00100111	01000100	00001101	00001111	01111111	01111110
hex	0x8B	0x2C	0x27	0x44	0x0D	0x0F	0x7F	0x7E

C/C++:

```
fstatus = FT_Write(m_ftHandle, "\x8B\x2C\x27\x44\x0D\x0F\x7F\x7E", 8, &dwBytesWritten);
fstatus = FT_Read(m_ftHandle, szBufferRead, 5, &dwBytesReturned);
```


response format:

position	0	1	2	3	4
hexadecimal	0x8B	0x01	0x02	0x03	0x00
hexadecimal	0x8b	0x00030201 (I32)			
decimal	cmd	197121 (dec)			
decimal	cmd	counter/position value			

Remarks:

The bytes of the response aren't shifted. The first byte is the command byte and the next bytes are four bytes of an unsigned integer value, the least significant byte was sent first.

7.5 Binary move to absolute position

cmd byte	(hex)	bits	bits	bits	bits	bits	bits	bits	axis
10001100	0x8C	0bbbbbbb	0bcccccc	0ceeeeee	0eeeffff	0fffggg	0ggggghh	0hhhhh0	x
10001101	0x8D	0bbbbbbb	0bcccccc	0ceeeeee	0eeeffff	0fffggg	0ggggghh	0hhhhh0	y
10001110	0x8E	0bbbbbbb	0bcccccc	0ceeeeee	0eeeffff	0fffggg	0ggggghh	0hhhhh0	z

Example:

axis: x

velocity(dec): 25000

velocity(hex): 0x61A8

velocity(bin): 01100001 10101000

delta position(dec): 12000

delta position(hex): 0x00002EE0

delta position(bin): 00000000 00000000 00101110 11100000

	cmd	lsb	msb	lsb			msb	
hex	0x8C	0xA8	0x61	0xE0	0x2E	0x00	0x00	
bin	10000110	01010100	01100001	11100000	00101110	00000000	00000000	
shifted	10000110	00101010	00011000	00111100	00000010	01110000	00000000	00000000

axis: z

velocity(dec): -25000

velocity(hex): 0x9E58

velocity(bin): 10011110 01011000

delta position(dec): -12000

delta position(hex): 0xFFFFD120 (The **two's complement** of a binary number)

delta position(bin): 11111111 11111111 11010001 00100000

	cmd	lsb	msb	lsb			msb	
hex	0x8E	0x58	0x9E	0x20	0xD1	0x00	0x00	
bin	10001011	01011000	10011110	00100000	11010001	11111111	11111111	
shifted	10001011	00101100	00100111	01000100	00001101	00001111	01111111	01111110
hex	0x8E	0x2C	0x27	0x44	0x0D	0x0F	0x7F	0x7E

C/C++:

```
fstatus = FT_Write(m_ftHandle, "\x8E\x2C\x27\x44\x0D\x0F\x7F\x7E", 8, &dwBytesWritten);
fstatus = FT_Read(m_ftHandle, szBufferRead, 5, &dwBytesReturned);
```

response format:

position	0	1	2	3	4
hexadecimal	0x8E	0x01	0x02	0x03	0x00
hexadecimal	0x8E	0x00030201 (I32)			
decimal	cmd	197121 (dec)			
decimal	cmd	counter/position value			

Remarks:

The bytes of the response aren't shifted. The first byte is the command byte and the next bytes are four bytes of an unsigned integer value, the least significant byte was sent first.

7.6 Binary get counter and velocity

command byte	hexadecimal	Explanation
10001111	0x8F	Get counter & velocity x axis
10010000	0x90	Get counter & velocity y axis
10010001	0x91	Get counter & velocity z axis

Example:

axis: x

C/C++:

```
ftstatus = FT_Write(m_ftHandle, "\x8F", 1, &dwBytesWritten);
ftstatus = FT_Read (m_ftHandle, szBufferRead, 7, &dwBytesReturned);
```

response format:

position	0	1	2	3	4	5	6
hexadecimal	0x8F	0x01	0x02	0x03	0x00	0x0A	0x00
hexadecimal	0x8F	0x00030201 (I32)				0x000A (I16)	
decimal	cmd	197121 (dec)				10 (dec)	
decimal	cmd	counter/position value				velocity value	

Remarks:

The bytes of the response aren't shifted. The first byte is the command byte and the next bytes are four bytes of an unsigned integer value, the least significant byte was sent first. The last two bytes are the unsigned short integer value of the velocity the least significant byte was sent first.

response :

counter/position: **197121 (dec)**
velocity: **10 (dec)**

Velocity:

Velocity value	Single steps/s	Full steps/s	Single steps / Full step
0..8191	0..50000	0..781,25	64
8192..16283	50000..100000	781,25 .. 1562,5	64
16284..32767	50000..100000	1562,5..3125	32