HPI-3D Communication protocol

Wrocław 21 Feb 2019

Rev. A.4

1. INTRODUCTION

Communication of a PC host computer with an HPI hardware is done physically over USB port or wireless over Bluetooth. Drivers for USB communication are attached in the installation package in the *HPI Software/FTDI Driver* catalogue or at FTDI site: http://www.ftdichip.com/FTDrivers.htm.

After power on or device reset, after around 10 seconds the device firmware is ready to work. Data are sent on request. There are five main types of data that can be received from the laser:

- Distance
- Velocity
- Dynamic
- Meteo

Protocol details are described in the next paragraph.

USB/COM port parameters	
Speed	3000000 bps
Data length	8 bit
Stop bit	1 stop bit
Parity	No
Handshaking	No

BT port parameters	
Speed	230400 bps
Data length	8 bit
Stop bit	1 stop bit
Parity	No
Handshaking	No

2. PROTOCOL DETAILS

2.1. Frame to the device

| START | COMMAND | DATA | CRC |

START - 1 byte - always 0xAA
COMMAND - 2 bytes
DATA - 4 bytes
CRC - 1 byte

System frame type list:

0xB0, 0x32	**READ DISTANCE ON**
Frame:	|0xAA|0xB0|0x32|0|0|0|0|CRC|
	After this command system changes result type to distance and starts sending distance data to PC. Answer is frame **DISTANCE DATA**.
0xB0, 0x33	**READ DISTANCE OFF**
Frame:	|0xAA|0xB0|0x33|0|0|0|0|CRC|
	This command ends sending measured distance data to PC.
0xB0, 0x34	**READ VELOCITY ON**
Frame:	|0xAA|0xB0|0x34|0|0|0|0|CRC|
	After this command system changes result type to velocity and starts sending distance data to PC. Answer is frame **VELOCITY DATA**.
0xB0, 0x35	**READ VELOCITY OFF**
Frame:	|0xAA|0xB0|0x35|0|0|0|0|CRC|
	This command ends sending measured velocity data to PC.
0xB0, 0x3C	**DON'T SEND ANY STREAM DATA**
Frame:	|0xAA|0xB0|0x3C|0|0|0|0|CRC|
	This command turns off all automatic data transfer from device to PC (i.e. distance, velocity, etc.)

0xB0, 0x3D	**DELETE 'SMALL SIGNAL LEVEL' FLAG**
Frame:	|0xAA|0xB0|0x3D|0|0|0|0|CRC|
	This command resets 'Small Signal Level' flag in system. After sending this command it is necessary to clear measured values.
0xB0, 0x3F	**DELETE 'VELOCITY OVERFLOW' FLAG**
Frame:	|0xAA|0xB0|0x3F|0|0|0|0|CRC|
	This command resets 'Velocity Overflow' flag in system. After sending this command it is necessary to clear measured values.
0xB0, 0x40	**DELETE 'EXTERNAL CAPTURE' FLAG**
Frame:	|0xAA|0xB0|0x40|0|0|0|0|CRC|
	This command resets 'External capture' flag. This command has to be send every time after using external trigger to result capture. Before sending this command system is not sensitive for trigger.
0xB0, 0xAE	**DYNAMIC ON**
Frame:	|0xAA|0xB0|0xAE|0|0|0|0|CRC|
	After this command system begins sending dynamic data to application. **SAMPLE_RATE** (Word) accpets one of following decimal values: 1 – 10 Hz 2 – 20 Hz 5 – 50 Hz 10 – 100 Hz 20 – 200 Hz 50 – 500 Hz 100 – 1 kHz 200 – 2 kHz 500 – 5 kHz 1000 – 10 kHz 2000 – 20 kHz 5000 – 50 kHz 10000 – 100 kHz For sample rate between 10 Hz and 10 kHz answer is frame **DYNAMIC DATA**. For higher sample rate answer is **FAST DYNAMIC DATA**
0xB0, 0xAF	**DYNAMIC OFF**
Frame:	|0xAA|0xB0|0xAF|0|0|0|0|CRC|
	This command stops dynamic data transfer.

0xB0, 0x79	**METEO ON**
Frame:	|0xAA|0xB0|0x79|0|0|0|0|CRC|
	This command turns on TPH unit (meteo station) in system and it begins sending meteo data to application. Answer is frame **METEO DATA WIRELESS** sent periodically every 1 second for each sensor independently.
0xB0, 0x7A	**METEO OFF**
Frame:	|0xAA|0xB0|0x7A|0|0|0|0|CRC|
	This command turns off TPH unit (meteo station) in system and it stops sending meteo data to application.
0xB0, 0x48	**CLEAR MEASURE RESULTS**
Frame:	|0xAA|0xB0|0x48|0|0|0|0|CRC|
	This command forces measured result to zero. It does not clean data waiting in in-system fifo queue.
0xB0, 0x58	**READ XY DATA ON**
Frame:	|0xAA|0xB0|0x58|0|0|0|0|CRC|
	This command changes DPS mode to XY (3D positioning system). If DSP STREAM option is selected, after this command system begins putting measured XY data form DPS into in-system fifo queue and it begins sending data from queue to application. Answer is frame XY DATA (frames form DSP).
0xB0, 0x59	**READ XY DATA OFF**
Frame:	|0xAA|0xB0|0x59|0|0|0|0|CRC|
	This command ends putting measured XY results (3D positioning system) into in-system fifo queue. If there has not been other command putting data into queue, it stops all transmission to application. DSP STREAM is not turned off.
0xB0, 0x5D	**READ XYZ DATA ON**
Frame:	|0xAA|0xB0|0x5D|0|0|0|0|CRC|
	This command changes DPS mode to XYZ (laser head gyroscope). If DSP STREAM option is selected, after this command system begins putting measured XYZ data from DSP into in-system fifo queue and it begins sending data from queue to application. Answer is frame XYZ DATA (frames form DSP).

0xB0, 0x5E	**READ XY DATA OFF**
Frame:	|0xAA|0xB0|0x5E|0|0|0|0|CRC|
	This command ends putting measured XY results (3D positioning system) into in-system fifo queue. If there has not been other command putting data into queue, it stops all transmission to application. DSP STREAM is not turned off.
0xB0, 0x91	**LASER HEAD ON**
Frame:	|0xAA|0xB0|0x91|0|0|0|0|CRC|
	Frame that turns on the laser head.
0xB0, 0x92	**LASER HEAD OFF**
Frame:	|0xAA|0xB0|0x92|0|0|0|0|CRC|
	Frame that turns off the laser head.

2.2. Frame from device

| START | COMMAND | DATA | CRC |

Frame START = **0xAA**:

START - 1 byte - 0xAA (FFT and DYNAMIC DATA – 0xAB)
COMMAND - 2 bytes
DATA - 12 bytes
CRC - 1 byte

Frame START = **0xAB**:

| START | LEVEL | COMMAND | FLAG | DATA (frame size 117 bytes)

START - 1 byte
LEVEL - 1 byte
COMMAND - 1 bytes
FLAG - 2 bytes
DATA - 112 bytes

Frame START = **0xAC**:
| START | COMMAND | LEVEL | FLAG | DATA |CRC2| (frame size 26 bytes)

START - 1 byte
COMMAND - 2 bytes
LEVEL - 1 byte
FLAG - 2 bytes
DATA - 18 bytes
CRC2 - 2 byte

System frame type list:

0xB0, 0x00	**OK**
Frame:	|**0xAA**|0xB0|COM|0|0|0|0|0|0|0|0|0|0|0|CRC|
	COM is equal number of acknowledged command. It should be received after listed command and means that command was received and executed.
0xB0, 0x15	**DISTANCE DATA (sent automatically each 40ms)**
Frame:	|**0xAA**|0xB0|0x15|DISTANCE – 7 bytes|0|0|FLAG2|FLAG|LEVEL|CRC|
	DISTANCE is a distance value in [100pm] units **FLAG** is a first byte of system status: - 1st bit – if set laser head is ready to measure - 3rd bit – that is 'overheat' flag - 4rd bit – that is 'Small Signal Level' flag **FLAG2** is a second byte of system status: - 3rd bit – if set „Velocity Overflow" occurred **LEVEL** is a value of power of laser beam signal on a receiver photodiode.
0xB0, 0x16	**VELOCITY DATA (sent automatically each 40ms)**
Frame:	|**0xAA**|0xB0|0x16|VELOCITY – 4 bytes|0|0|0|0|0|FLAG2|FLAG|LEVEL| CRC|
	VELOCITY is a velocity value in [100nm/s] units. **FLAG** is a first byte of system status: - 1st bit – if set laser head is ready to measure - 3rd bit – that is 'overheat' flag - 4rd bit – that is 'Small Signal Level' flag **FLAG2** is a second byte of system status: - 3rd bit – if set „Velocity Overflow" occurred **LEVEL** is a value of power of laser beam signal on a receiver photodiode.
0xB0, 0x0D	**DYNAMIC DATA (sent automatically each 10µs)**
Frame:	|**0xAC**|0xB0|0x0D|LEVEL|FLAG2|FLAG|DYNAMIC DATA(6B+(3 x 4B))| CRC2 - size 26 bytes
	DYNAMIC DATA includes 4 values – first data, absolute, is 6 bytes long, the next differential data are 4 bytes long. All data are in [100pm] units.

	FLAG is a first byte of system status: - 1st bit – if set laser head is ready to measure - 3rd bit – that is 'overheat' flag - 4rd bit – that is 'Small Signal Level' flag **FLAG2** is a second byte of system status: - 3rd bit – if set „Velocity Overflow" occurred **CRC2** – sum of first 24 bytes – size 2 bytes **LEVEL** is a value of power of laser beam signal on a receiver photodiode.
0x17	**FAST DYNAMIC DATA (sent automatically each 10µs)**
Frame:	|**0xAB**|LEVEL|0x17| FLAG2|FLAG|DYNAMIC DATA (38b+(39 x 22b)) – size 117 bytes
	DYNAMIC DATA includes 40 values of special compressed distance to gain high transmission speed. In one long frame there are transmitted 40 words of data – first data, absolute, is 38 bits long, the next differential data are 22 bits long. All data are in [100pm] units. **FLAG** is a first byte of system status: - 1st bit – if set laser head is ready to measure - 3rd bit – that is 'overheat' flag - 4rd bit – that is 'Small Signal Level' flag **FLAG2** is a second byte of system status: - 3rd bit – if set „Velocity Overflow" occurred **LEVEL** is a value of power of laser beam signal on a receiver photodiode.
0xB0, 0x0A	**WIRELESS METEO DATA (sent automatically each 1s, independently for each sensor)**
Frame:	|**0xAA**|0xB0|0x0A|METEO DATA – 12 bytes|CRC|
	METEO DATA includes six environment parameters: 1st byte - number of wireless sensor: 0 – air temp, 1-3 base temp 2nd - 3rd byte - temp. measured by wireless sensor [0.01 C] 4th byte - air humidity [1 %] 5th byte - battery state 6th byte - wireless link state 7th – 8th byte - air pressure measured by sensor in TPH unit [0.1 hPa] This frame is received when wireless meteo sensors are connected to system.

3. ALGORITHMS

3.1. CRC algorithm

Below there is presented function to calculate control sum of frame (written in C++)

```
unsigned char Count_CRC(unsigned char crc, unsigned char c)
{
const unsigned char CRC_MASK = 0x31;
unsigned char cnt;

for(cnt = 0; cnt < 8; cnt++)
{
if((crc^c) & 0x80)
       crc = (crc << 1)^CRC_MASK;
else
       crc <<= 1;

c <<= 1;
}

return crc;
}
```

This function has two arguments and returns one result. First argument is CRC from calculation before and second argument is the next byte from transmission frame. Initial value of CRC before start counting should be equal 0xFF.

An example for calculation CRC byte frame to device is presented below.

```
Data_Frame[7] = 0xFF;

for(int i = 0; i < 7; i++)
Data_Frame[7] = Count(Data_Frame[7], Data_Frame[i]);

To test value of CRC in received frame may use code as below:

CRC = 0xFF;

for(int i = 0; i < 16; i++)
CRC = Count_CRC(CRC, Data_Frame[i]);

if(CRC == 0)
...data are correct...
else
...transition error...
```

3.2. FTDI algorithm

Set of procedures USB for access through the FTDI USB driver. For more information please look in: http://www.ftdichip.com/Drivers/D2XX.htm.

```
#include "FTD2XX.h"

   FT_HANDLE Con_Handle;
   unsigned long device_number;
```

```
   FT_DEVICE_LIST_INFO_NODE device_table[16];
   FT_STATUS is_ok;

       FT_CreateDeviceInfoList(&device_number);
       FT_GetDeviceInfoList(device_table,&device_number);
       FT_Open(i,&Con_Handle);
       FT_SetBaudRate(Con_Handle,3000000);
       FT_SetDataCharacteristics(Con_Handle,        FT_BITS_8,        FT_STOP_BITS_1,
FT_PARITY_NONE);
       FT_SetTimeouts(Con_Handle, 5000, 1000);
       FT_Purge(Con_Handle, FT_PURGE_RX | FT_PURGE_TX);

       FT_Write(Con_Handle, buffer, 2, &Number_Bytes_Written);
       FT_Read(Con_Handle, buffer, 1, &Number_Bytes_Read);
```

4. Initialization and measurement procedures

4.1. Simple measure initialization

After power on HPI needs 4-5 min for heating (depends on external conditions). During this time measuring cannot be proceed. When LSB bit in FLAG byte is set that laser frequency become stable and measure is possible

To start sending measure results it is required to execute commands 0xB0, 0x32 or 0xB0, 0x36. After that device sends measure results continuously with period time 40ms.

If 1st bit in FLAG byte is set that laser head is ready to measure.

Application should control FLAG and FLAG2 in each frames to control if measure result is right.

After any error occurs measure data may be incorrect. Thus reset measure and delete flag by send adequate commands is required.