REQUESTING FLIGHTS FROM FLYMASTER F1

Contents

Introduction
This is a technical document for use by anyone who wants to develop an application for reading flight data from the F1 module. Other users should use a ready made program, like ReadF1.exe available on the Flymaster site or GPSDump developed by Stein Sorensen, to read this data.

1. **Communicating with the F1**
   The communication with the F1 module is achieved through an RS232 port or Bluetooth serial port. The following parameters should be used:

Baud rate	57600
Bits	8
Parity	None
Stop bits	1
Flow control	None

   When running normally the F1 will be sending standard NMEA navigation sentences to the serial port.

2. **Requesting device identification:**
   *$PFMSNP,*3A*

   F1 will reply with a:
   *$PFMSNP,Flymaster F1,HW:n,FW:k.kk,jjjjj*cs*

   Where:
   n= hardware version
   k.kk = firmware version (note: functionality described in document only applies to version 1.16 or higher)
   jjjjj = device serial number
   cs= standard nmea checksum.

3. **Requesting flight list**
   To request list of flights stored in memory the host application should send:
   *$PFMDNL,LST,*56*

   On receiving this command the F1 stops sending NMEA navigation sentences, and will reply with the list of flights stored in its memory using the following format.

   *$PFMLST,ttt,nnn,dd.mm.yy,hh:mm:ss,hh:mm:ss*cs*

   where:

Ttt	Number of flights in memory
Nnn	0 based flight number, most recent flight will be first
dd.mm.yy	date of first record of the flight (UTC)
hh:mm:ss	time of first record (UTC)
hh:mm:ss	Duration of the flight

   example:
   ....
   $PFMLST,062,052,01.06.07,14:15:32,00:32:58*33
   $PFMLST,062,053,01.06.07,12:29:45,00:02:46*37
   $PFMLST,062,054,01.06.07,11:53:15,00:33:00*3b
   $PFMLST,062,055,01.06.07,11:46:35,00:06:17*3c
   $PFMLST,062,056,01.06.07,11:41:51,00:04:29*35
   ....

4. Requesting a flight

To request a flight from the F1, send:
$PFMDNL,yymmddhhmmss,*cs

where:
yymmddhhmmss is the date and time obtained from the previously mentioned flight list and cs is the standard nmea checksum.

example:
*$PFMDNL,070601141532,*1D*

will downlaod flight 052 from the example list above.

From here on the F1 goes into binary mode. and will send data blocks, which will be at most 200 bytes long.

Each data block starts with packet identification (2 bytes), followed by the data length (1 byte), x data bytes, 1 crc character. See table bellow.

	Position	**Length**	**Value**
Packet ID	0	2	0xa0a0
Packet Length	2	1	Number of data bytes
Data	3	variable	
crc	3+variable		Xor of packet length and data bytes

For every data block block received the host application must reply 1 byte:

0xb1 ok, F1 will send the next block
0xb2 not ok, F1 will resend data block
0xb3 abort transfer, F1 will go back into command mode.

When the F1 has no more data to send, it will send 0xa3a3 indicating no more data. It will return to command mode.

There are 3 types of data blocks:

Packet ID	
0xa0a0	Flight information record
0xa1a1	Key track position record
0xa2a2	Track position record deltas

a. Flight information record:

	Position	Length	Value
Packet ID	0	2	0xa0a0
Packet Length	2	1	0x3f
Firmware version	3	2	16 bit int
Hardware version	5	2	16 bit int
Device serial number	7	4	32 bit int
Pilot competition number	11	8	
Pilot name	19	15	
Glider brand	34	15	
Glider model	49	15	
Crc	64	1	Xor of packet length and data bytes

b. Key track position record

This data block identifies a position of a track record.

	Position	Length	Value
Packet ID	0	2	0xa1a1
Packet Length	2	1	0x11
Fix flag	3	1	8 bit unsigned int
Latitude	4	4	32 bit signed int
Longitude	8	4	32 bit signed int
Altitude	12	2	16 bit signed int
Barometric pressure *10	14	2	16 bit signed int
Time	16	4	32bit unsigned int
Crc	20	1	Xor of packet length and data bytes

Notes:

A negative Latitude is South, negative Longitude is East.

To convert integer coordinates (latitude,longitude) to degrees divide the integer value by 60000, in other words: Degrees = Lat/60000.

The 8th bit, of the Fix flag is set for an ‘A’ type fix and unset for a ‘V’ type fix.

To convert barometric pressure to altitude, use the following formula:

*(1 - pow(fabs((Presure/10.0)/1013.25), 0.190284)) * 44307.69;*

Time is the number of seconds elapsed since the first second of the year 2000. To convert this value to time please contact us we will provide a C a sample source code function to obtain date and time from this value.

c. Track position record deltas

This data block contains several (up to 30) variations to the previous position.

	Position	Length	Value
Packet ID	0	2	0xa2a2
Packet Length	2	1	Variable
Fix flag	3	1	8 bit unsigned int
Latitude offset	4	1	8 bit signed int
Longitude offset	5	1	8 bit signed int
Altitude offset	6	1	8 bit signed int
Pressure offset	7	1	8 bit signed int
Time offset	8	1	8bit unsigned int
…… fix flag… etc…			
Crc	??	1	Xor of packet length and data bytes

The offsets should be added to the previous record calculated.

For example: Altitude on the “key track position record” was initially 150, on the “record deltas” the we find an altitude offset of -2, now the Altitude is 148, then a +1 , now the Altitude is 149. Etc…

5. Sending pilot information

To set pilots competition number name glider information in the F1 module, the send the following command to the F1.

*$PFMPLT,name,compnumber,gliderbrand,glidermodel,*cs*

name =pilot name
compnumber = competition number
gliderbrand= glider brand
glidermodel= glider model
cs= standard nmea checksum

6. Calculating and Validating NMEA Checksums

Error correction and detection in NMEA data is handled through the use of checksums. A checksum is a two-character hexadecimal number, located at the end of each NMEA sentence, representing the "two's complement" of the sentence:

$GPGSA,A,2,29,19,28,,,,,,,,,23.4,12.1,20.0*0F

In other words, each byte value between the dollar sign ($) and asterisk (*) is XOR'ed.