Document status

Published status	Confidentiality status
☒ Draft	☒ Internal
☐ Released	☐ NDA
☐ Canceled	☐ Public

Version control

Ver.	Date	Summary of changes
1.00	2016-04-22	Document created.
1.01	2016-06-24	Added warning when transferring flights from devices with bluetooth module type 1.
1.02	2021-01-08	Updated decl and declzone, proposal for radio control (not implemented yet in Sxxx)
1.03	2021-02-02	Updated PLVXS and PLXVF with flap information (not implemented yet in Sxxx)
1.04	2021-07-19	Updated LXWP0 description

Scope and summary

This document provides information about how to communicate over the serial interface of nano and nano3 devices from any device with serial input or bluetooth module. It only describes version 3 protocol messages.

Table of contents

1. Overview 5
2. Sentence Composition 6
3. PLXVC Sentences 8
3.1 Log book size 8
3.2 Log book 8
3.3 Flight 11
3.4 Read/Write Declaration (ASCII, BASE64) 12
3.5 Read decl zone 15
3.6 Set 17
3.7 Info 19
3.8 Bluetooth info 22
3.9 Free space 22
3.10 Bootloader info 23
3.11 Count files 24
3.12 File list 25
3.13 Deleting file 26
3.14 Download file 27
3.14.1 FILE_INFO & FILE_DATA 27
3.14.2 FILE_OK 30
3.14.3 FILE_DATA_LOST & FILE_CS_INVALID 31
3.14.4 FILE_CANCEL 32
3.14.5 FILE_CRC32 & FILE_CRC_OK 33
3.15 File upload 34
3.15.1 FILE_INFO 34
3.15.2 FILE_DATA 36
3.15.3 FILE_OK 39
3.15.4 FILE_DATA_LOST & FILE_CS_INVALID 39
3.15.5 FILE_CANCEL 41

3.15.6 FILE_CRC32 & FILE_CRC_OK 42
3.16 GPSINFO 43
3.17 KEEP_ALIVE 43
4. LXWP (0, 1, 2, 3, 4, 5) Sentences 47
4.1 LXWP0 47
4.2 LXWP1 47
4.3 LXWP2 48
4.4 LXWP3 49
4.5 LXWP4 50
4.6 LXWP5 51
5. PFLX (0, 1, 2, 3, 4, 5) Sentences 53
5.1 PFLX0 53
5.2 PFLX1 54
5.3 PFLX2 54
5.4 PFLX3 54
5.5 PFLX4 54
5.6 PFLX5 54
6. PLXV0 Sentences 55
6.1 MC Read/Write 55
6.2 BAL Read/Write 55
6.3 BUGS Read/Write 56
6.4 VOL Read/Write 57
6.5 POLAR Read/Write 57
6.6 CONNECTION Read/Write 58
6.7 NMEARATE Read/Write 59
6.8 ELEVATION Read/Write 60
6.9 QNH Read/Write 61
6.10 BRGPS Read/Write 61
6.11 BRPDA Read/Write 62

6.12 ALTOFF Read/Write 63

7. PLXVF Sentence 64

8. PLXVS Sentence 65

9. PLXVTARG Sentence 66

1. Overview

This document provides information about version 3 protocol for communication over the serial or Bluetooth interface of a nano or nano3 device.

The document assumes you are familiar with serial communication and the basics of the NMEA protocol.

2. Sentence Composition

All commands must start with a "$" (0x24) sign and end with a "*" (0x2A) followed by two NMEA 0183 standard checksum characters and <CR><LF> (0x0D0A). The checksum is two – digit hexadecimal representation of XOR of ASCII codes of all characters between, but not including, the $ and * characters.

Fields are delimited with a comma (0x2C), even if a field is empty. The field length is variable. Commands must always include valid ASCII characters. The sentences are not case sensitive. The maximum number of characters in a sentence is 256.

Messages described in this document are version 3 protocol messages.

Syntax:

$<Sentence type>,<Key>,<Query type>,<Parameters>*<CHECKSUM><CR><LF>

Description:

Field	Description
Sentence type	Possible values: - PLXVC - LXWP (0, 1, 2, 3, 4, 5) - PFLX (0, 1, 2, 3, 4, 5) - PLXV0 - PLXVF - PLXVS - PLXVTARG
Key	Possible values: - LOGBOOKSIZE - LOGBOOK - FLIGHT - DECL / DECL_ENCODED - DECL_ZONE

	- SET - INFO - BT_INFO - FREE_SPACE - BOOT_INFO
Query type	Possible values: - R: Request to read content of Key. In this case you can safely disregard Value field. - W: Request to set Value to Key - A: Device answers request of reading content of a Key. - C: Device answers request of setting content of a key. This parameter is left out when dealing with LXWP and PFLX statements. Note that if you send request (R) when sending PLXV0 type of message, it will respond with query type value W (not A).
Parameters	See the following pages.

3. PLXVC Sentences

3.1 Log book size

Syntax:

Request: PLXVC,LOGBOOKSIZE,R

Answer: PLXVC,LOGBOOKSIZE,A,<Size>

Description:

Parameter	Description
Size	This parameter is of type integer. It represents total number of igc files on device. (Note: igc files are counted only in the root folder. Any files in subfolders are ignored)

Example

Request: PLXVC,LOGBOOKSIZE,R

Response: PLXVC,LOGBOOKSIZE,A,10

We send a Request to get the log book size. The device will respond with Response where the last parameter is the total number of igc files. This information is useful when reading log book.

3.2 Log book

Syntax:

Request: PLXVC,LOGBOOK,R,<Start flight>,<End flight>

Answer:

- PLXVC,LOGBOOK,A,<Flight number>,<Number of flights>,<File name>,<Date>,<Start time>,<End time>,<File size>
- PLXVC,LOGBOOK,A,0

Description:

Logbook is a collection of igc files which represents recorded flights. Send this request to device to get a list (not the actual files) of recorded flights between start flight and end flight. In order to receive list of all flights, you will first need to send a request for log book size. This will tell you total number of igc files on the device. If there is no igc files on device, device will return the second answer (PLXVC,LOGBOOK,A,0).

Parameter	Description
Start flight	This parameter is of type integer. It represents the consecutive number of flight file to begin with. Example: Start flight = 10. That means the device will start transferring list of flights beginning with flight number 10 in the overall list of flights on device. The first flight starts with number 1. So in order to get flight number 1 in the list you should set this parameter to 1.
End flight	This parameter is of type integer. It represents the last flight in the list of igc files to be transferred. In order to get list of all flights the start file needs to be set to 1 and the end flight needs to be set to “Log book Size” + 1. Example: PLXVC,LOGBOOK,1,3 Device will return two flights (1 and 2).

Parameter	Description
Flight number	This parameter is of type integer. It represents the consecutive number of current flight transferred from the list of all flights. (starting with 1)

Number of flights	This parameter is of type integer. It represents the number of all flights on device.
File name	Name of the flight currently being transferred. This parameter is of type string. It includes the file extension (.igc)
Date	The date of the flight. It is in the following format: DD.MM.YYYY
Start time	Time of the takeoff. It is written in the following format: HH:MM:SS Nano may also respond with --:--:--
End time	Time of the landing. It is written in the following format: HH:MM:SS Nano may also respond with --:--:--
File size	This is the size (in bytes) of the igc file. This parameter is of type ulong.

Example

Request: PLXVC,LOGBOOK,R,1,3

First response:
PLXVC,LOGBOOK,A,1,5,63NV2Z21.igc,18.05.2016,15:33:40,17:20:11,115230

Second response:
PLXVC,LOGBOOK,A,2,5,63MV36J2.igc,15.04,2016,18:20:05,20:41:46,455677

We send a Request to get the first two flights from the logbook. The device will answer with to messages. The first response will tell you the information about the first flight in the logbook. You can also see the whole log book size from the response (in this case there are 5 flights in the logbook). The second response will tell you the information about the second flight in the logbook.

3.3 Flight

Syntax:

Request: PLXVC,FLIGHT,R,<File name>,<Start row>,<End row>

Answer: PLXVC,FLIGHT,A,<Current line>,<Number of lines>,<Line content>

Description:

Transfer of the igc file can be done by sending a minimum of two requests to device. You will have to send at least two requests because you don't know how many rows the file has. The first response will have that information. The best practice is to start the transfer of only one line, just to get the total amount of rows in file. But of course you don't need to transfer the whole file just in two request. You can transfer the file by any block size (by block size we mean number of rows). But be careful when selecting block size. Devices with bluetooth type 1 may stop responding if the block size is to big. We recommend to set it to 7.

Parameter	Description
File name	This parameter is of type string. It represents the name of the igc file to be transferred. The name must include file extension (.igc).
Start row	This parameter is of type integer. It represents the starting row in file, from where the content should be sent. For example: PLXVC,FLIGHT,R,example.igc,3,6 Here we told the device to send us three lines, starting with the line number 3 (inclusive) and ending at line number 6 (exclusive). To start at the very first line of the file this parameter should be set to 1.
End row	This parameter is of type integer. It represents the last row to be sent in block. So if we want the last row in a block to be row number 5 from the file, this parameter needs to be set to 6.

Parameter	Description
Current line	This parameter is of type integer. It represents the consecutive number of the line in file currently being sent. (The first line starts with number 1).
Number of lines	This parameter is of type integer. It represents the number of all lines in the file.
Line content	This parameter is of type string. It contains the actual data (ASCII chars) from the line "Current line" in file.

Example

Request: PLXVC,FLIGHT,R,63NV2Z21.igc,1,3

Response1: PLXVC,FLIGHT,A,1,30,ALXV23GFLIGHT:1

Response2: PLXVC,FLIGHT,A,2,30, HFDTE191115

We send a Request to get the first two lines of the 63NV2Z21.igc flight. The device will respond with two consecutive messages. The first one will contain the first line of the file, and the second one, the second line of the file. You can notice that the message will also contain the total amount of lines in the file. In this case the file contains 30 flights. If you don't get the response right away, don't stop listening for the response. The device is counting lines before it sends the first line of the file and this can take very long time if the file is large.

3.4 Read/Write Declaration (ASCII, BASE64)

Syntax:

Request1: PLXVC,DECL,R,<Start row>,<End row>

Request2: PLXVC,DECL,W,<Current line>,<Number of lines>,<Line content>

Request3: PLXVC,DECL_ENCODED,R,<Start row>,<End row>

Request4: PLXVC,DECL_ENCODED,W,<Current line>,<Number of lines>,<Base64 line content>

Answer1: PLXVC,DECL,A,<Current line>,<Number of lines>,<Line content>

Answer2: PLXVC,DECL,C,<Current line>

Answer3: PLXVC,DECL_ENCODED,A,<Current line>,<Number of lines>,<BASE64 line content>

Answer4: PLXVC,DECL_ENCODED,C,<Current line>

Description:

Declaration can be written/received in two different encodings. The first encoding is ASCII. To send/receive file in this encoding use Request1, Request2 and Answer1, Answer2.

You can also send/receive file using BASE64 encoding. Everything in the protocol is exactly the same except that the line content is encoded using BASE64 algorithm. To do this use Request3,Request4 and Answer3, Answer4.

Read

Declaration file contains information about pilot and fly task. Reading this file from device can be done by sending a minimum of two requests (request number 1). You will have to send at least two requests because you don't know how many rows the file has. The first response will have that information. The best practice is to start the transfer of only one line, just to get the total amount of rows in file. But of course you don't need to transfer the whole file just in two request. You can transfer the file by any block size (by block size we mean number of rows).

Write

Request2 is the syntax for writing declaration file to device. For each line of the file you sent, you will get Answer2 from device.

Parameter	Description

Start row	This parameter is of type integer. It represents the starting row in file, from where the content should be sent. For example: PLXVC,DECL,R,3,6 Here we told the device to send us three lines, starting with the line number 3 (inclusive) and ending at line number 6 (exclusive). To start at the very first line of the file this parameter should be set to 1.
End row	This parameter is of type integer. It represents the last row to be sent in block. So if we want the last row in a block to be row number 5 from the file, this parameter needs to be set to 6.

Parameter	Description
Current line	This parameter is of type integer. It represents the consecutive number of the line in file currently being sent/received. (The first line starts with number 1).
Number of lines	This parameter is of type integer. It represents the number of all lines in the file.
Line content	This parameter is of type string. It contains the actual data (ASCII chars) from the line “Current line” in file.
Base64 line content	This parameter is of type string. It contains the data from the line “Current line” in file which is encoded with base64 algorithm. This parameter is only acceptable with DECL_ENCODED key.

Example:

Read:

Request1: PLXVC,DECL,R,1,2

Response1: PLXVC,DECL,A,1,30,CONTENT

We send a request to read the first line of the declaration file. From the device's response we can see that there are 30 lines in the declaration file and the content of the first line is "CONTENT".

Write:

Request2: PLXVC,DECL,W,1,30,CONTENT

Response2: PLXVC,DECL,C,1

We send a request to write the first line of the declaration file. Device will know that there is 30 lines in the decl file and the content of the first line is "CONTENT".

Example of decl file:

```
HFPLTPILOT:Uros Krasovic
HFCM2CREW2:
HFGTYGLIDERTYPE:
HFGIDGLIDERID:
HFCIDCOMPETITIONID:
HFCCLCOMPETITIONCLASS:
C050221133458050221020502
C0000000N00000000E
C4846667N01015883EAALEN HEIDENHEI::585.00000
C4711500N00942100ED21CH NENZING::572.00000
C4748667N01346550EEBENSEE EBXTRAUN::420.00000
C4744110N01133501EWACKERSBERG::831.00000
C0000000N00000000E
LLXVOZ=-1,Style=2,R1=3000m,A1=45.0,R2=0m,A2=0.0,A12=193.6,Maxa=0m,Line=1,Autonext=1,Lat=4846.667N,Lon=01015.883E,Near=0
LLXVOZ=0,Style=1,R1=5100m,A1=180.0,R2=0m,A2=0.0,A12=44.7,Maxa=0m,Autonext=1,AAT=1,Lat=4711.500N,Lon=00942.100E,Near=0
LLXVOZ=1,Style=1,R1=10000m,A1=89.5,R2=0m,A2=0.0,A12=261.9,Maxa=0m,Line=1,Autonext=1,Lat=4748.667N,Lon=01346.550E,Near=0
LLXVOZ=2,Style=3,R1=500m,A1=180.0,R2=0m,A2=0.0,A12=87.9,Maxa=0m,Autonext=1,Lat=4744.100N,Lon=01133.100E,Near=1
LLXVTSK,StartOnEntry=false,Short=false,Near=true
```

Elevation of the waypoint C4744110N01133501EWACKERSBERG::**831.00000** Elevation of the wayponit is written as **bold** marked in meters

LLXVOZ=<number>
-1 Starting point
0 1st point
1 2nd point
2 3rd point

Style=<number> `typedef enum { ozFixed=0, ozSymmetric, ozNext, ozPrev, ozStart, ozUnknown } eOzStyle;`

Line=<1 or 0> The zone is specified as Line

R1=<radius in meters – zone1>

A1=<angle in degrees – zone1> for sector 90° the angle is 45°

R2=<radius in meters – zone2>

A2=<angle in degrees – zone2> for sector 90° the angle is 45°

A12=<angle in degrees> is calculated from the Style

Maxa=<maximum altitude in meters>

Autonext=<1 or 0>, 1 – when you enter into the zone, the task will switch to the next waypoint. For AAT zones, if you don't autonext, this value is set to 0

Lat=0000.000N,Lon=00000.000E position of the waypoint

Near=<1 or 0> navigate to the nearest point of the zone

3.5 Read decl zone

Syntax:

Request: PLXVC,DECL_ZONE,R,<Start row>,<End row>

Response: PLXVC,DECL_ZONE,A,<Current line><Number of lines><Row content>

Description:

This request will read the decl zone file. On your device you can edit zones for a task. If you don't edit the zones, device will use default zones. You can edit this default zones on your device, and are then written to decl zone file. For parameter details see the following table.

Parameter	Description
Start row	This parameter is of type integer. It represents the starting row in file, from where the content should be sent. For example: PLXVC,DECL_ZONE,R,1,2 Here we told the device to send us one line (the first line), starting with the line number 1 (inclusive) and ending at line number 2 (exclusive).
End row	This parameter is of type integer. It represents the last row to be sent in block. So if we want the last row in a block to be row number 2 from the file, this parameter needs to be set to 3.

Parameter	Description

Current line	This parameter is of type integer. It represents the consecutive number of the line in file currently being sent/received. (The first line starts with number 1).
Number of lines	This parameter is of type integer. It represents the number of all lines in the file.
Row content	This parameter is of type string. It contains the actual data (ASCII chars) from the line “Current line” in file.

Example:

Request: PLXVC,DECL_ZONE,R,1,2

Response: PLXVC,DECL_ZONE,1,3,CONTENT

We send a request to read the first line of the decl zone file. The device will respond with Response. Be very careful when parsing this line. The actual content (line read from decl zone file) contain commas.

Example of declzone file:

```
LLXVOZSTART=0,Style=2,R1=3000m,A1=45.0,R2=0m,A2=0.0,A12=16.7,Maxa=0m,Autonext=1,Lat=0000.000N,Lon=00000.000E,Near=0
LLXVOZPOINT=0,Style=1,R1=3000m,A1=45.0,R2=0m,A2=0.0,A12=165.8,Maxa=0m,Autonext=1,Lat=0053.961N,Lon=00016.188E,Near=0
LLXVOZFINISH=0,Style=3,R1=3000m,A1=45.0,R2=0m,A2=0.0,A12=315.0,Maxa=0m,Autonext=1,Lat=0016.188N,Lon=00053.961E,Near=0
```

LLXVOZSTART=<number>

Style=<number> `typedef enum { ozFixed=0, ozSymmetric, ozNext, ozPrev, ozStart, ozUnknown } eOzStyle;`

Line=<1 or 0> The zone is specified as Line

R1=<radius in meters – zone1>

A1=<angle in degrees – zone1> for sector 90° the angle is 45°

R2=<radius in meters – zone2>

A2=<angle in degrees – zone2> for sector 90° the angle is 45°

A12=<angle in degrees> is calculated from the Style

Maxa=<maximum altitude in meters>

Autonext=<1 or 0>, 1 – when you enter into the zone, the task will switch to the next waypoint. For AAT zones, if you don’t autonext, this value is set to 0

Lat=0000.000N,Lon=00000.000E position of the waypoint

Near=<1 or 0> navigate to the nearest point of the zone

3.6 Set

Syntax:

Request1: PLXVC,SET,R,<Settings key>

Request2: PLXVC,SET,W,<Settings key><Value>

Response1: PLXVC,SET,A,<Settings key><Value>

Response2: PLXVC,SET,C,<Settings key><Value>

Description:

This command is used for reading/writing various settings from/to device. See the following table for more details about the settings you can change/read. For reading use Request1 along with Response1 and for writing use Request2 along with Response2.

Parameter	Description
Settings key	This parameter can be one of the following string values: - **NMEARATE:** Nmea interval (ms) - **BAUDRATE:** External uart baudrate - **AUTOOFF:** Automatically turn off device - **OFFIN:** Automatically finish flight - **NEARDIS:** Near distance - **ALWRUN:** Logger permanent on - **NMEA:** Enable NMEA output - **ACCELL:** Use accel - **RECINT:** Recording interval
Value	This parameter is of type integer. It is the value set/read of the settings key. Possible values: - **NMEARATE**: integer - **BAUDRATE**: 2400, 4800, 9600, 19200, 38400, 57600, 115200

	- **AUTOOFF**: 0 (disable), 1(enable) - **OFFIN**: 0 (disable), 1(enable) - **NEARDIS**: integer - **ALWRUN**: 0 (disable), 1(enable) - **NMEA**: 0 (disable), 1(enable) - **ACCEL**: 0 (disable), 1(enable) - **RECINT**: 1 - 60

Example:

Read

Request1: PLXVC,SET,R,AUTOFF

Response1: PLXVC,SET,A,1

We send a Request1 to read the setting AUTOFF. The device will respond with Response1 where the last parameter is the value of this setting. In this case AUTOOFF is enabled.

Write

Request2: PLXVC,SET,W,AUTOFF,0

Response2: PLXVC,SET,C,AUTOFF,0

We send Request2 to disable the AUTOFF setting. Device will respond with Response2, where the last parameter is the new value of the AUTOFF setting.

3.7 Info

Syntax:

Request: PLXVC,INFO,R

Response: PLXVC,INFO,A,<Device name>,<Application version>,<Version date and time>,<Hardware serial>,<Battery voltage>,<Backup battery voltage>,<Press Alt>,<Is charging>,<Enl>,<Logger status>,<Power consumption>,<Battery current>,<Battery percent>,<Remaining time>,<Serial number>,<Security key valid>,<Gps status>,<Gps char count>

Description:

Send this request to get the basic information about the device. For more details see the following table.

Parameter	Description
Device name	This parameter is of type string and represents the device name. Possible values: - N3 (stands for nano 3) - NANO
Application version	This parameter is of type string and tells you the current version of software on your device.
Version date and time	This parameter is of type string and gives you information about when the software currently installed on your device was released.
Hardware serial	This parameter is of type string and givers you serial number of your hardware.
Battery voltage	This parameter is of type float rounded by two decimal places. It tells you the voltage of your battery.
Backup battery voltage	This parameter is of type float rounded by two decimal places. It tells you the voltage of your backup battery.
Press Alt	This parameter is of type float rounded by two decimal places. It tells you the altitude pressure.
Is charging	This parameter is of type int. It can take two values (0 or 1), and indicates whether your device is charging or not.

Enl	This parameter is of type long. It tells you the engine noise level. It can take values between 0 and 999.
Logger status	This parameter is of type int. It tells you the logger status. This value can be on off: - 0: Stop - 1: Can stop - 2: Start
Power consumption	
Serial number	This parameter is of type string. It can contain only a serial number, or it can contain serial number along with hardware configuration and hardware version. Example1: 10 (only serial) Example2: 10-0x4d-24 (hardware configuration is hex value)
Security key valid	This parameter is of type int. It tells you if security key is valid or not (value is either 0 or 1).
Gps status	/
Gps char count	/

Example:

Request: PLXVC,INFO,R

Response: PLXVC,INFO,A,N3,2.32,May 13 2016 08:30:24,,4.27,6.60,236.13,1,14,0,,,,,65535-0x03-1,0,0,0

We send a request to get basic information about device. Device returns a Response containing the info. You can see that some of the parameters are not sent.

3.8 Bluetooth info

Syntax:

Request: PLXVC,BT_INFO,R

Response: PLXVC,BT_INFO,A,<Bluetooth type>

Description:

Use this statement to get information about bluetooth device contained inside your device. This information is useful because the BM77 module can freeze when transferring large amount of data. To overcome this problem when transferring for example igc file, the blocksize can be reduced.

Parameter	Description
Bluetooth type	This parameter is of type integer. It represents which bluetooth device is installed in your device. Possible values are: - -1 (No bluetooth) - 0 (BTM 182) - 1 (BM 77)

Example:

Request: PLXVC,BT_INFO,R

Response: PLXVC,BT_INFO,A,1

3.9 Free space

Syntax:

Request: PLXVC,FREE_SPACE,R

Response: PLXVC,FREE_SPACE,A,<Free space>

Description:

This request will tell you total amount of free space on your device. This information is useful when transferring file to your device, because if there is not enough space for some file to be saved, the file transfer will fail.

Parameter	Description
Free space	This parameter is of type integer. The value represent total amount of free space in MB. If for some reason the device cannot calculate the amount of free space, it will return -1.

Example:

Request: PLXVC,FREE_SPACE,R

Response: PLXVC,FREE_SPACE,A,2969

3.10 Bootloader info

Syntax:

Request: PLXVC,BOOT_INFO,R

Response: PLXVC,BOOT_INFO,A,<Bootloader version>

Description:

Send this request if you want to know the current installed bootloader version.

Parameter	Description
Bootloader version	This parameter is of type integer. It represents the bootloader version.

Example:

Request: PLXVC,BOOT_INFO,R

Response: PLXVC,BOOT_INFO,A,180

3.11 Count files

Syntax:

Request: PLXVC,FILE_CNT,R,<File extension>

Response: PLXVC,FILE_CNT,A,<File extension>,<Number of files>

Description:

Send this command to get number of files on your device. This information is useful when transferring file list. It will also scan for files inside folders.

Parameter	Description
File extension	This parameter is of type string. It represents the file extension. You may specify any file extension. In order to get all files on your device set this parameter to ALL. Do not include dot when specifying the extension. For example if you want to count number of igc files you should set this parameter to igc (and not .igc).
Number of files	This parameter is of type int. It represents number of files with the specified file extension. If you set ALL as an file extension then also folders will be counted.

Example:

Request: PLXVC,FILE_CNT,R,ALL

Response: PLXVC,FILE_CNT,A,ALL,42

3.12 File list

Syntax:

Request: PLXVC,FILE_LIST,R,<File extension>

Response:
PLXVC,FILE_LIST,A,<Type>,<Name>,<Depth>,<Day>,<Month>,<Year>,<Size>

Description:

If you send this request you will get a complete list of files that has the same extension that you specified. If you specify "ALL" as a file extension, then you will get a complete list of all files and folders. For each file/folder on device you will get a response line.

Parameter	Description
File extension	This parameter is of type string. It represents the file extension. You may specify any file extension. In order to get all files on your device set this parameter to ALL. Do not include dot when specifying the extension. For example if you want to get a list of igc files you should set this parameter to igc (and not .igc).
Type	This parameter is of type string. It's value is one of: - FOLDER - FILE
Name	This parameter is of type string. It specifies file/folder name. In case of file, file extension is included.
Depth	This parameter is of type int. It represents the depth. If the file or folder is in root it will have a depth of 0. If the file is inside let's say two nested folders than it will have a depth of 2. (Example: Folder airspace contains a folder dummy which contains a file dummy.txt)

Day	This parameter is of type int. It represents the day when this file/folder was created.
Month	This parameter is of type int. It represents the month when this file/folder was created.
Year	This parameter is of type int. It represents the year when this file/folder was created.
Size	This parameter is of type long. It tells you the size of the file/folder in bytes.

Example:

Request: PLXVC,FILE_LIST,R,igc

Responses:

PLXVC,FILE_LIST,FILE,5BJV23G1.igc,0,19,11,2015,1105253
PLXVC,FILE_LIST,FILE,5BEV23G1.igc,0,15,11,2015,4315404

3.13 Deleting file

Syntax:

Request: PLXVC,FILE_DELETE,R,<File name>

Response: PLXVC,FILE_DELETE,A,<File name>,<Success>

Description:

Sent this statement to delete a file on device. Device will respond with confirmation if the file was successfully deleted or not.

Parameter	Description
File name	This parameter is of type string. It represents the name of the file that should be / was deleted.

Success	This parameter is of type int. It's value is either 0 or 1. If the file was successfully deleted this parameter's value will be 1 and 0 otherwise.

Example:

Request: PLXVC, FILE_DELETE,R,5BJV23G1.igc

Response: PLXVC,FILE_DELETE,A, 5BJV23G1.igc,1

3.14 Download file

Overview:

You can download a file from device. This download process is composed of multiple statements such as: FILE_INFO, FILE_DATA, FILE_OK, FILE_DATA_LOST, FILE_CS_INVALID, FILE_CANCEL, FILE_CRC32, FILE_CRC_OK. This statements, including the correct order of sending them, is described below.

3.14.1 FILE_INFO & FILE_DATA

Syntax:

Request: PLXVC,FILE_INFO,R,<File path>,<Block size>,<Payload size>

Response: PLXVC,FILE_DATA,A,<Lost>,<Current line>,<Current crc>,<Payload>

Description:

The file download (device will send file to device that sends this info statement) begins with FILE_INFO statement. Once the device receives this statement, it will begin with sending you actual data of the file wright away.

Parameter	Description
File path	This parameter is of type string. It represents the full path of the file that you want to download from device.
Block size	This parameter is of type int. It represents, how many FILE_DATA messages will you get at once. For example if you set this to 20, you will get the first block which will be 20 lines long. After that you need to send a message FILE_OK, to confirm you have received the first block of data. Do not exaggerate with this number. Depending on the bluetooth type installed in device, set this parameter to either 1 or more. There are known issues with bluetooth type equal to 1 (which is BM77 with no flow control) when sending large amount of data. In this case set this parameter to 1. Anyway, keep in mind that sending large amount of data at once may cause problems. Maximum allowed value is 30.
Payload size	This parameter is of type int. Maximum allowed value is 100. It represents how many bytes of file will be sent in one DATA line. Keep in mind that the actual length of payload inside a DATA statement will not be 100B, in fact it will be a little larger due to the overhead of base64 algorithm.

Parameter	Description
Lost	This parameter is of type int. It's value is either 1 or 0. Let's say the device sends you first block (block_size = 20) and you for some reason didn't get the 10th DATA statement from that block. You send back to device a DATA_LOST statement where you specify the consecutive number (in this case 10) of DATA statement that you didn't get. Device will respond to you with another block of data beginning with that exact DATA statement that you lost. And because this statement was lost the value of this parameter will be 1. But be careful when receiving the lost data. Before you get the lost DATA

	statement you probably will get the leftover statements from the previous block (the one where you lost a statement). So make sure to discard all DATA statements from the point where you send a DATA_LOST messages to the point where you get a DATA message with this parameter set to 1.
Current line	This parameter is of type int. It's the consecutive number of a DATA line (but not inside a block). The very first data statement will have this parameter set to 0, the next one 1 and so on. For example if the block_size = 20, you will receive 20 DATA statements where the last statement will have this parameter set to 19. When you receive another block of DATA statements (same block size), the last DATA statement in second block will have this parameter set to 39. With this parameter you can determine loss of data. For example you receive the very first line with this parameter set to 0. The next statement you receive has this parameter set to 2. That means that you lost the data, because you didn't receive the DATA statement where this parameter is set to 1. In this case send the DATA_LOST message which is described later on.
Current crc	This parameter is of type int. It represents a 32 bit crc code of the base64 encoded data payload. But this crc code is updated at every data statement, so it depends on the previous crc that was sent (except for the first one ofcourse). This parameter is used to ensure integrity of the data. You should be calculating this crc on your side also, and then compare it to the one you got from the DATA message. If they are not the same, something went wrong (either you're not calculating it write or the data was corrupted during the transfer). If the compared numbers are not the same you can send a DATA_LOST message.
Payload	This parameter is of type string. It contains base64 encoded data from the file, so make sure to decode it before appending it to a file.

Example:

Request: PLXVC,FILE_INFO,R,/65AV2Z22.igc,2,10

Responses:

PLXVC,FILE_DATA,R,0,0,-329691785,QUxYVjJaMkZMSQ==

PLXVC,FILE_DATA,R,0,1,450159786,R0hUOjINCkhGRA==

3.14.2 FILE_OK

Syntax:

Request: PLXVC,FILE_OK,R,<Current received line + 1>

Response: PLXVC,FILE_DATA,A,<Lost>,<Current line>,<Current crc>,<Payload>

Description:

Once you receive a block (you know how many lines will be in the block) you send FILE_OK message to receive another block. For more details see the following table.

Parameter	Description
Current received line +1	This parameter is of type int. It represents the next line of the next block. For example if you received the first block where the block size is 20, then set this parameter to 20.

Example:

This example is based on the previous example (block size was 2)

Request: PLXVC,FILE_OK,R,2

Responses:

PLXVC,FILE_DATA,R,0,2,-190256176,VEUxMDA1MTYNCg==

PLXVC,FILE_DATA,R,0,3,-346101300,SEZGWEEwMTUNCg==

3.14.3 FILE_DATA_LOST & FILE_CS_INVALID

Syntax:

Request1: PLXVC,FILE_DATA_LOST,R,<Lost statement number>

Request2: PLXVC,FILE_CS_INVALID,R,<Lost statement number>

Response: PLXVC,FILE_DATA,A,<Lost>,<Current line>,<Current crc>,<Payload>

Description:

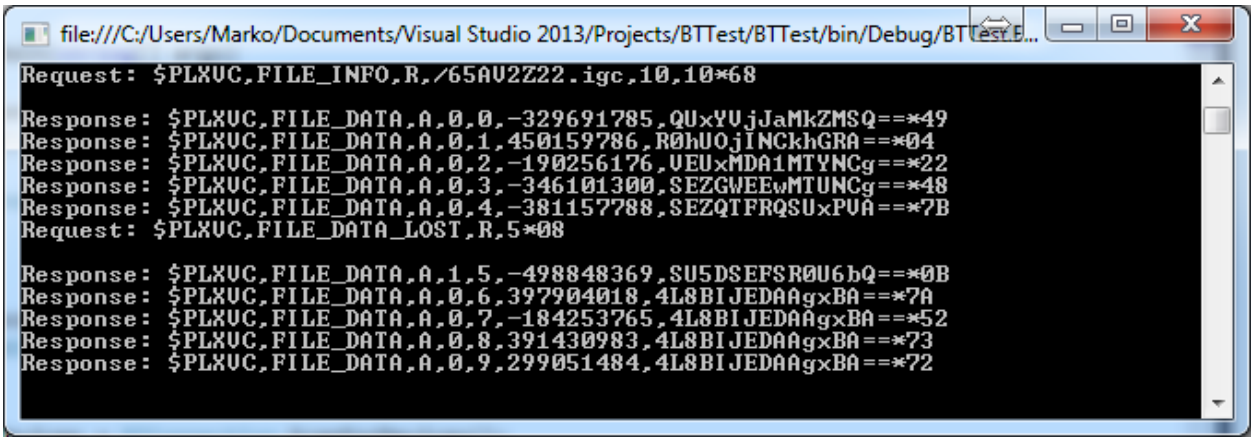
We've already discussed when to send this type of request. Send Request1 if the FILE_DATA sentence that you received, don't have the value of parameter <Current line> bigger for 1 than the previous received statement. For example you receive statement with <Current line> parameter set to 2. The next statement you receive has the parameter <Current line> set to 4. That means that you lost the statement with parameter <Current line> set to 3. To overcome this problem send FILE_DATA_LOST statement with parameter <Lost statement number> set to 3, and this statement will be sent again.

Send Request2 message if you got the message but the checksum of the message was incorrect. That means that part of the message is corrupted.

Parameter	Description
Lost statement number	This parameter is of type int. Set this number to the value of parameter <Current line> of the message you lost. (You know this value because it is equal to previous message <Current line> parameter +1)

Example:

We first send a FILE_INFO statement with block size set to 10. We then receive the first 5 statements (the parameter <Current line> of the last received statement is 4). The next statement we receive has a parameter <Current line> set to 6. That means we didn't get the statement with parameter <Current line> set to 5. We send FILE_DATA_LOST message and skip all other messages until we get the message we lost.



Notice from the picture above that the message we lost has the parameter <Lost> set to 1.

3.14.4 FILE_CANCEL

Syntax:

Request: PLXVC,FILE_CANCEL,R

Response: /

Description:

Send this request if you want to cancel the transfer. You won't get any response. You can safely say that the transfer was canceled and device stop sending

messages. If device do not get this message don't worry because it will eventually time out.

Example:

Request: PLXVC,FILE_CANCEL,R

3.14.5 FILE_CRC32 & FILE_CRC_OK

Syntax:

Request1: PLXVC,FILE_OK,R,<Current received line + 1>

Request2: PLXVC,FILE_CRC_OK,R

Response: PLXVC,FILE_CRC32,A,<File crc>

Description:

When you receive the last FILE_DATA (you will know which one is the last because you know the file size and you know how many lines have you received and you know how many bytes of the file does a single FILE_DATA line contain) message, just respond to it normally with the FILE_OK message. You will get a response FILE_CRC32 containing the 32 bit crc calculated over the whole file. This is just another check that the file is exactly the same on both sides. When this messages is received calculate 32 bit crc code of the file you received and compare it to the one you received in message FILE_CRC32. If these to numbers are the same, the transfer was completed successfully. However, if you don't get this message try sending FILE_OK message again. Once you know that the crc is ok and everything went well, send FILE_CRC_OK message just to notify device to close the file and clean other mess. If you don't do that and you immediately start a new file download task, it will fail immediately when you send FILE_INFO message. But if

you don't start a new transfer right away, nothing will go wrong because the device will eventually timeout and reset everything for the new file transfer.

Example:

Received: PLXVC,FILE_DATA,A,0,11243,-7866704,RTcNCg==

Sent: PLXVC,FILE_OK,R,11244

Received: PLXVC,FILE_CRC32,R,-97241912

Sent: PLXVC,FILE_CRC_OK,R

3.15 File upload

Overview:

You can also upload a file to device. This upload process is composed of multiple statements such as: FILE_INFO, FILE_DATA, FILE_OK, FILE_DATA_LOST, FILE_CS_INVALID, FILE_CANCEL, FILE_CRC32, FILE_CRC_OK. This statements, including the correct order of sending them, is described below. You don't need to send any special commands to cancel / stop the transfer. Device will timeout if it won't receive any messages for some period of time.

3.15.1 FILE_INFO

Syntax:

Request: PLXVC,FILE_INFO,W, <File path>,<Block size>,<File size>

Response: PLXVC,FILE_OK,C,<Current received line + 1>

Description:

To start uploading a file to device, sent FILE_INFO message. Device will immediately respond with a FILE_OK message. You can safely discard the <Current received line +1> parameter at this point. If you don't get the FILE_OK response, something went wrong. Either there is not enough disk space, another transfer is already in progress, you exceeded the maximum allowed block size or for some reason a new file couldn't be created. Also note that the message FILE_OK is sent multiple times (every 100ms) until you send a new message, or the device times out.

Parameter	Description
File path	This parameter is of type string. It's value specify where should device safe the file and what the name of that file should be.
Block size	This parameter is of type int. It's value must be less than or equal to 30. Be very careful when setting this parameter. It tells device how many lines of FILE_DATA will it receive before it sends you back a FILE_OK message. There are known issues with some bluetooth types (type 1 which is BM77, to be precise) if this parameter is set to high. In fact for this bluetooth type you should set this parameter to 1.
File size	This parameter is of type int. It's value must be equal the size of the file you want to upload. The size must be expressed in bytes. It is used to determine if the whole file was transferred successfully.

Example:

Request: PLXVC,FILE_INFO,/file.txt,2,112434

Response: PLXVC,FILE_OK,C,-1

3.15.2 FILE_DATA

Syntax:

Request: PLXVC,FILE_DATA,W,<Lost>,<Line number>,<Payload size>,<Current crc>,<Encoded payload size>,<Encoded payload>

Response1: PLXVC,FILE_OK,C,<Current received line + 1>

Response2: PLXVC,FILE_DATA_LOST,C,<Lost line>

Response3: PLXVC,FILE_CS_INVALID,C,<Lost line>

Response4: PLXVC,FILE_CANCEL,C

Description:

Once you get the response on FILE_INFO message, you can start sending FILE_DATA messages which contains the file content. Note that you must send exactly <Block size> (a parameter from FILE_INFO message) number of FILE_DATA messages. Once you send them you will get another FILE_OK response and you are ready to send another block. Make sure you get the correct FILE_OK message by checking the <Current received line + 1> parameter. If the block size is small the following can happen: The blocksize is 2 you sent the first block (statements 0, 1) and receive a FILE_OK message with parameter value of 2. You send next block (statements 2, 3) and get an FILE_OK message with parameter value of 2. This is not the correct FILE_OK message. Make sure you get the FILE_OK message with parameter of value 4. This can happen because FILE_OK message is sent multiple times (to make sure you get it).

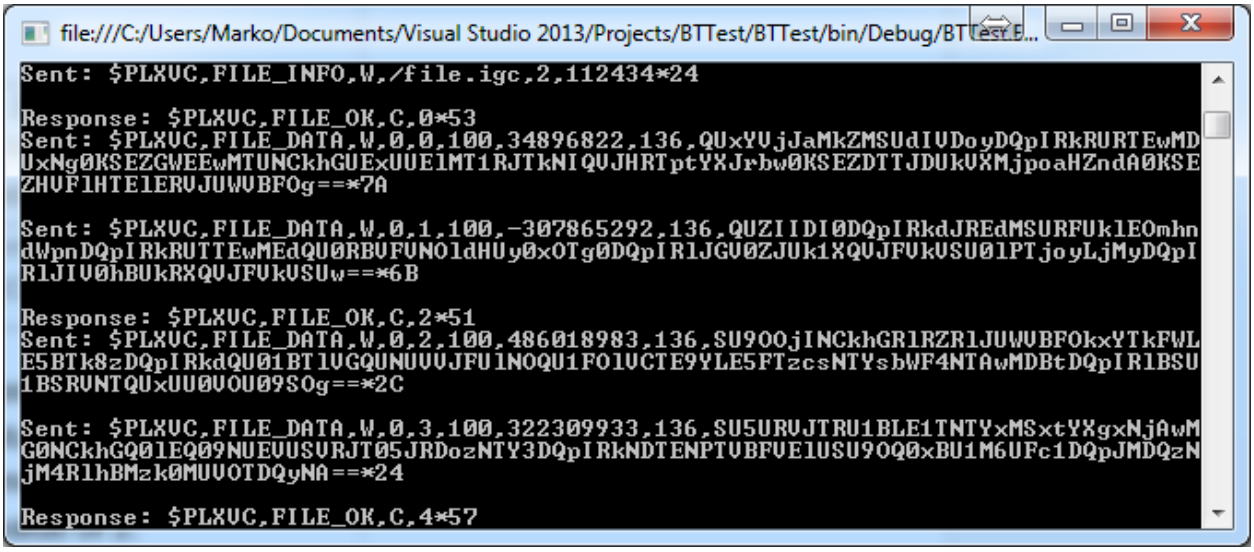
Keep in mind that there could be loss of data and device won't send you FILE_OK message, instead you will get either FILE_DATA_LOST or FILE_CS_INVALID message. This two statements are sent only once! If something terribly went wrong, device will respond with FILE_CANCEL message. In this case stop the transfer, because device will clean everything and delete previously created file.

Parameter	Description
Lost	This parameter is of type int. It's value is either 1 or 0. By default set this value to 0. If device responds you with a message FILE_DATA_LOST or FILE_CS_INVALID set this parameter to 1 when you will retransfer the lost message. For more details of lost messages see FILE_DATA_LOST and FILE_CS_INVALID messages.
Line number	This parameter is of type int. Every FILE_DATA message must have this parameter set to value equal to previous message Line number parameter value + 1, starting with 0. For example the first FILE_DATA line must have this parameter set to 0. Next line must have this parameter 1, next one 2 and so on. If you get the message FILE_DATA_LOST or FILE_CS_INVALID make sure this parameter is correct! For more details see FILE_DATA_LOST and FILE_CS_INVALID messages description.
Payload size	This parameter is of type int. This number represents how many bytes of actual file data does this FILE_DATA line contains. For example if you sent 10B of data from file set this parameter to 10. It must always be expressed in bytes.
Current crc	This parameter is of type int. It represents 32 bit crc code. It must be calculated only on the base64 encoded payload. So first read the data from file, encode it with base64 algorithm and then calculate the 32 bit crc code. When you done this for the very first FILE_DATA statement, store this value somewhere. Once you are sending new FILE_DATA statement do not calculate 32 bit crc on fresh. Use the crc of the previous statement and just update it. Otherwise device will send you FILE_DATA_LOST message. You may also get this message if some of the bytes get corrupted during the transfer and that is the main reason why calculating the crc at all.

Encoded payload size	This parameter is of type int. It represents the size of base64 encoded payload.
Encoded payload	This parameter is of type string. It represents the data encoded with base64 algorithm. Just a quick note: do not exaggerate with the payload size (max 100B).

Example:

In the following example we first send a FILE_INFO statement. We get FILE_OK response. Next we start sending the data. We send two lines since we specified that the block size is equal to 2. The payload size is 100B, the encoded payload size is 136 bytes due to the overhead of base64 algorithm. The last parameter is the base64 encoded data. Once the device receives these two messages it responds back with a FILE_OK message. You can see that the parameter of the FILE_OK message is equal to 2 (which is the next FILE_DATA line consecutive number). Next we send another block. And we wait for FILE_OK answer. But not just any FILE_OK answer. The value of its parameter must be equal to 4. Keep that in mind.



3.15.3 FILE_OK

Syntax:

Response: PLXVC,FILE_OK,C, <Current received line + 1>

Description:

Device will respond with this message when it receives the FILE_INFO statement or when the whole block of FILE_DATA statements is received. You can safely discard the parameter when you receive this statement as a confirmation on FILE_INFO statement. If this statement is received as confirmation of a block received you must check the value of the parameter. We already discussed this in the previous section, but once again, when you get the FILE_OK response you must ensure that the value of the parameter is the same as the consecutive number of the FILE_DATA statement which is sent next. As we already mentioned this message is sent multiple times to ensure you get one. Others must be skipped.

Example:

See the example of the section: 3.15.2 FILE_DATA.

3.15.4 FILE_DATA_LOST & FILE_CS_INVALID

Syntax:

Response1: PLXVC,FILE_DATA_LOST,C,< Consecutive number +1>

Response2: PLXVC,FILE_CS_INVALID,C,< Consecutive number +1>

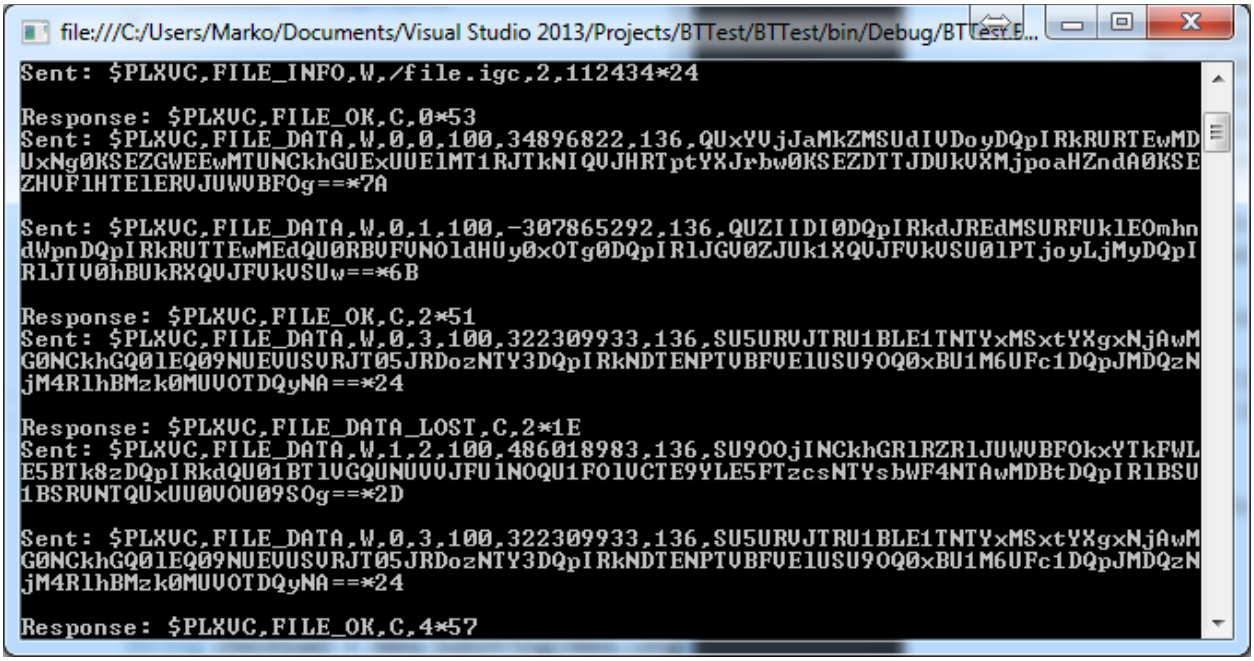
Description:

The parameter of this two responses is of type int. It represents the consecutive number of the FILE_DATA line that was previously received + 1. So that means if we send three FILE_DATA statements with consecutive number 1, 2, 3 and the device received the first one, but the second was lost, the parameter of

FILE_DATA_LOST statement which will be sent by device is going to be 2. This statement must be sent again in order to continue with the file transfer. Be careful when sending this message again because you must move the file pointer correctly and also the 32 bit crc code which you update on every FILE_DATA line sent, must be reversed! The same thing should be done when receiving FILE_CS_INVALID. Also when you are retransferring a FILE_DATA statement don't forget to set parameter "Lost" to 1 (but only for this statement).

Example:

In this example we intentionally generated a FILE_DATA_LOST message. We first sent the FILE_INFO message. Once the FILE_OK is received, we start the transfer by sending the first two FILE_DATA messages. Since the block size is equal to 2 we get the message FILE_OK. We then skipped the next FILE_DATA message (we generated it but just didn't send it to device) and send the 4th FILE_DATA statement. Because the device did not receive the third statement, it replied with FILE_DATA_LOST message. We then "retransferred" the third message (parameter lost is set to 1), along with the 4th message. The device received everything and responded with FILE_OK message.



3.15.5 FILE_CANCEL

Syntax:

Response: PLXVC,FILE_CANCEL,C

Description:

Device will respond with this message if one of the following problems / situations occur:

- If there is not enough disk space
- If there was a problem creating a file
- If there was a problem writing to file

Example:

We won't provide an example for this message because it is trivial. You just need to keep in mind that this message can be received and if it is received, stop transferring the data. (At this point there is nothing more you can do.)

3.15.6 FILE_CRC32 & FILE_CRC_OK

Syntax:

Request: PLXVC,FILE_CRC32,W,<File crc>

Response: PLXVC,FILE_CRC_OK,C

Description:

Once you've send the last FILE_DATA statement, you will receive a FILE_OK statement. In order to complete the transfer you must send message FILE_CRC32. Its parameter value is a 32 bit crc code calculated over the whole file (raw data). If this number will match to the one computed on device, then device will respond back at you with a message FILE_CRC_OK. This message is sent multiple times (just to make sure you get it). If you don't get the message, then the transfer wasn't successful, and device will delete the file from it's file system.

Example:

Request: PLXVC,FILE_CRC32,W, 322309933

Response: PLXVC,FILE_CRC_OK,C

3.16 GPSINFO

Syntax:

Request: PLXVC,GPSINFO,R

Response: PLXVC,GPSINFO,A,<Parameters>

Description:

When V7 receives LXWP1 sentence at GPS port, it will be forwarded as PLXVC,GPSINFO statement: PLXVC,GPSINFO,A,<LXWP1 statement>. Pda can also send a request for INFO of GPS (NANO). See example2 for more details.

Example1:

PLXVC,GPSINFO,A,LXWP1,NANO,1422,2.05,d129 (Forwarding LXWP1 sentence)

Example2:

PLXVC,GPSINFO,R (Pda sends this message to V7)

PLXVC,INFO,R (V7 forwards to GPS – nano)

PLXVX,INFO,R,NANO,2.05,May 12 2012 21:38:28,d129,4.25,2.78,258.18,1,8,0 (Nano responds to V7)

PLXVC,GPSINFO,A,INFO,A,NANO,2.05,May 12 2012 21:38:28,d129,4.25,2.78,256.70,0,4,0 (V7 forwards this message to PDA)

3.17 KEEP_ALIVE

Syntax:

Response: PLXVC,KEEP_ALIVE,W

Description:

This message is sent multiple times. For example, at the beginning of the flight transfer, device needs to count lines in the flight file. This operation can take very long time if the file is very big. When waiting for the first response, you might wait for a couple of seconds, so in case you have a timeout in your function, waiting for an answer, you may parse this line, just to be sure that device is doing something and it will answer you, and you don't need to timeout at this point.

Note: This haven't been implemented so far, but will be in the future.

Example:

PLXVC,KEEP_ALIVE,W

3.16 RADIO

Syntax:

Request: PLXVC, RADIO,R,<Command>

Response: PLXVC, RADIO,A,<Command>,<Parameters>

Request: PLXVC, RADIO,S,<Command>,<Parameters>

Description:

With Command we are setting between active or standby frequency, volume, squelch and vox setting. When we set frequency, additional parameter can be the name of the airfield.

Example1:

```
$PLXVC,RADIO,S,COMM,128800,CELJE
$PLXVC,RADIO,A,COMM,128800,CELJE
$PLXVC,RADIO,R,COMM

$PLXVC,RADIO,S,SBY,128800,CELJE
$PLXVC,RADIO,A,SBY,128800,CELJE
```

$PLXVC,RADIO,R,SBY

$PLXVC,RADIO,S,VOL,60
$PLXVC,RADIO,A,VOL,60
$PLXVC,RADIO,R,VOL

$PLXVC,RADIO,S,SQUELCH,60
$PLXVC,RADIO,A, SQUELCH ,60
$PLXVC,RADIO,R, SQUELCH

$PLXVC,RADIO,S,VOX,60
$PLXVC,RADIO,A,VOX,60
$PLXVC,RADIO,R,VOX

3.16 TRANSPONDER

Syntax:

Request: PLXVC, XPDR,R,<Command>

Response: PLXVC, XPDR,A,<Command>,<Parameter>

Request: PLXVC, XPDR,S,<Command>,<Parameter>

Description:

Example1:

$PLXVC,XPDR,S,SQUAWK,2000
$PLXVC,XPDR,A,SQUAWK,2000
$PLXVC,XPDR,R,SQUAWK

$PLXVC,XPDR,R,ALT
$PLXVC,XPDR,A,ALT,1234
Altitude in feet

$PLXVC,XPDR,S,MODE,<mode>
$PLXVC,XPDR,A,MODE,<mode>

$PLXVC,XPDR,R,MODE
where <mode> is:
NODATA
OFF
SBY
GND
ON
ALT
IDENT

$PLXVC,XPDR,S,STATUS,<status>
$PLXVC,XPDR,A,STATUS,<status>
$PLXVC,XPDR,R,STATUS
where <status> is:
RX
IDENT

4. LXWP (0, 1, 2, 3, 4, 5) Sentences

4.1 LXWP0

Syntax:

Response: LXWP0,<Logger stored>,<Air speed>,<Air altitude>,<vl0>,<vl1>,<vl2>,<vl3>,<vl4>,<vl5>,<Hdg>,<Wind course>,<Wind speed>

Description:

Parameter	Description
Logger stored	This parameter is of type char. It's value can be Y or N. IT represents if the logger was stored or not.
TAS	This parameter is of type uint_8t. It represents the true air speed in km/h.
Standard pressure altitude	This parameter is of type float. It represents air altitude in meters QNE.
vl0, vl1, vl2, vl3, vl4, vl5	These parameters are vario values. These are the last 6 measurements in the last second. Their type is float.
Hdg	This parameter represents heading of plane.
Wind course	/
Wind speed	/

Example: /

4.2 LXWP1

Syntax:

Response: LXWP1,<Instument ID>,<Serial number>,<Sw version>,<Hw version>,<License>

Description:

Parameter	Description
Instument ID	This parameter is of type string. It represents device ID.
Serial number	This parameter is of type int. It represents the serial number.
Sw version	This parameter is of type string. It represents the installed software version
Hw version	This parameter is of type int. It represents the installed hardware version.
License	This parameter is of type string.

Example:

4.3 LXWP2

Syntax:

Response: LXWP2,<Mcready>,<Ballast>,<Bugs>,<Polar a>,<Polar b>,<Polar c>,<Audi vol>

Description:

Parameter	Description
Mcready	This parameter is of type float. It's value is in m/s.
Ballast	This parameter is of type float.
Bugs	This parameter is of type float. It's value is represented in %.
Polar a	This parameter is of type float.
Polar b	This parameter is of type float.
Polar c	This parameter is of type float.
Audio vol	This parameter is of type int.

Example:

4.4 LXWP3

Syntax:

Response: LXWP3,<alti offset>,<sc mode>,<vario fil>,<te filter>,<te level>,<vario avg>,<vario range>,<sc tab>,<sc low>,<sc speed>,<smart diff>,<glider name>,<time offset>

Description:

Parameter	Description
Alti offset	This parameter is of type float. It represents offset necessary to set QNE in ft. Default value is equal to 0.
Sc mode	This parameter is of type byte. It represents methods for automatic SC switch index. Values representation: - 0: External - 1: On circling - 2: Auto IAS Default value is equal to 1.
Vario fil	This parameter is of type float. It represents filtering of vario in seconds. Default value is equal to 1.
Te filter	This parameter is of type float. It represents filtering of TE compensation in seconds. Value range: 0.1 to 2.0. Default value is equal to 1.5
Te level	This parameter is of type float. It represents level of TE compensation from 50 to 150. Default value is equal to 0 which means that TE compensation is off.
Vario avg	This parameter is of type int. It represents averaging time in seconds for integrator. Value range: 5s to 30s. It's default value is 25.

Vario range	This parameter is of type float. It's value can be one of: 2.5, 5 or 10. It is expressed in m/s or kts. Default value is equal to 5.0
Sc tab	This parameter is of type float. It represent area of silence in SC mode. Values can range from 0 to 10. Default value is equal to 1.
Sc low	This parameter is of type int. It represents external switch/taster function. Values representations: - 0: Normal - 1: Inverted - 2: Taster Default value is 1.
Sc speed	This parameter is of type int. It represents speed of automatic switch from vario to sc mode if SCMODE=2. It's values is expressed in km/h. Value range: 50-150. Default value is 110.
Smart diff	This parameter is of type float. It represents smar vario filtering.
Glider name	This parameter is of type string. Maximum 14 characters. It represents glider name.
Time offset	This parameter is of type int. It is expressed in hours.

Example:

4.5 LXWP4

Syntax:

Response: LXWP4,<Sc>,<Netto>,<Relativ>,<Distance>,<Gl diff>,<Leg speed>,<Leg time>,<Integrator>,<Flight time>,<Battery voltage>

Description:

Parameter	Description
Sc	This parameter is of type float. It's value is expressed in m/s.
Netto	This parameter is of type float. It's value is expressed in m/s.
Relativ	This parameter is of type float. It's value is expressed in m/s.
Distance	This parameter is of type float. It's value is expressed in m.
Gl diff	This parameter is of type int. It's value is expressed in ft.
Leg speed	This parameter is of type int. It's value is expressed in km/h.
Leg time	This parameter is of type int. It's value is expressed in km/h.
Integrator	This parameter is of type float. It's value is expressed in m/s.
Flight time	This parameter is of type uint. It's value is expressed in sec.
Battery voltage	This parameter is of type float. It's value is expressed in V.

Example:

4.6 LXWP5

Syntax:

Response: LXWP5,<Audio vol>,<Audio sc>,<Audio mode>,<Audio fn>,<Audio fc>,<Audio fp>,<Alarm f1>,<Alarm f2>,<Alarm pause1>,<Alarm pause2>,<Alarm count>,<Units>,<Lcd ind>

Description:

Parameter	Description
Audio vol	This parameter is of type it. It's value is either 0, 1, 2. Value 0 represents volume low, value 1 represents volume high, value 2 represents "undefined". Default value is 0.
Audio sc	This parameter is of type int. It's value is either 0, 1, 2, 3. Value 0 represents SC POS. Value 1 represents SC NEG. Value 2 represents SC. Value 3 represents undefined. Default value is 2.

Audio mode	This parameter is of type int. It's value ranges from 0 to 6. Value 0 represents LIN/POS. Value 1 represents LIN/NEG. Value 2 represents LINEAR. Value 3 represents DIG/POS. Value 4 represents DIG/NEG. Value 5 represents DIGITAL. Value 6 represents undefined.
Audio fn	This parameter is of type int. It represents frequency for audio (Frequency at -5m/s). Default value is 300 Hz.
Audio fc	This parameter is of type int. It represents frequency for audo (Frequency at 0m/s). Default value is 500 Hz.
Audio fp	This parameter is of type int. It represents frequency for audio (max frequency at +5m/s). Default value is 1000 Hz.
Alarm f1	This parameter is of type int. It represents frequency in Hz. Default value is 3000.
Alarm f2	This parameter is of type int. It represents frequency in Hz. Default value is 2000.
Alarm pause1	This parameter is of type byte. Default value is 2.
Alarm pause2	This parameter is of type byte. Default value is 2.
Alarm count	This parameter is of type byte. It represents number of periods. Default value is 10.
Units	This parameter is of type int16_t. Bit 1, 2: Distance (km = 0, nm=1, ml = 2) Bit 3, 4: Speed (km/h = 0, kts=1, mph=2) Bit 5: Vspeed (m/s = 0, kts= 1) Bit 8: Altitude (m = 0, ft = 1) All other bits are not used.

Lcd ind	It represents LCD setup mask.
	`needle in vario lcd_ind&0x0003; { "VARIO", "SC", "NETTO", "RELATIV", "" }`
	`needle in sc (lcd_ind&0x000c)>>2;`
	`number1 in vario (lcd_ind&0x0070)>>4;`
	`{"ALT.","DIST.","GL.DIF.","SPEED","LEG SP.","TSK SP."}`
	`number1 in sc (lcd_ind&0x0380)>>7;`

	number2 in vario (lcd_ind&0x0c00)>>10; { "INT.", "TIME", "F.TIME ", "LEG TI.", "" } number2 in sc (lcd_ind&0x3000)>>12;

Example:

5. PFLX (0, 1, 2, 3, 4, 5) Sentences

5.1 PFLX0

Syntax:

PFLX0,<Command1>,<Parameter1>,<Command2>,<Parameter2>…<Command5>, <Parameter5>

Description:

Commands can either have parameters or not.

Parameter	Description
Command with parameters	GPGGA, GPRMC, LXWP0, LXWP1, LXWP2, LXWP3, LXWP4, LXWP5
Commands without parameters	**COLIBRI**: Lx1600 switches to colibri port. **LX1600**: Switched back to communication with LX1600 on 4800bps **AUTOZERO**: Request auto zero function **EEPROM**: Store actual settings into eeprom **AUDIODEMO**: Starts with audio demo **ALARM**: Requests alarm. **INITEEPROM**: Initialize all settings to default, writes to eeprom and resets unit **RAWMODE:** Set pda raw mode. **NANO:** Switches to nano port. **VSEVEN:** Switched back to communication with v7.

Example:

5.2 PFLX1

Syntax: Same structure as LXWP1.

Description: Same structure as LXWP1.

Example: /

5.3 PFLX2

Syntax: Same structure as LXWP2.

Description: Same structure as LXWP2.

Example: /

5.4 PFLX3

Syntax: Same structure as LXWP3.

Description: Same structure as LXWP3.

Example: /

5.5 PFLX4

Syntax: Same structure as LXWP4.

Description: Same structure as LXWP4.

Example: /

5.6 PFLX5

Syntax: Same structure as LXWP5.
Description: Same structure as LXWP5.

Example: /

6. PLXV0 Sentences

6.1 MC Read/Write

Syntax:

Request1: PLXV0,MC,R,

Request2: PLXV0,MC,W,<Macc ready>

Response1: PLXV0,MC,W,<Macc ready>

Response2: No response.

Description:

Parameter	Description
Macc ready	Parameter macc ready is type of float. It is expressed in m/s.

Example:

Request1: PLXV0,MC,R

Response1: PLXV0,MC,W,1.2

6.2 BAL Read/Write

Syntax:

Request1: PLXV0,BAL,R

Request2: PLXV0,BAL,W,<Overload factor>

Response1: PLXV0,BAL,W,<Overload factor>

Response2: No response.

Description:

Parameter	Description
Overload factor	Overload factor is defined as ratio between current mas and mass of glider, where polar was calculated. This parameter is of type float.

Example:

Request1: PLXV0,BAL,R

Response1: PLXV0,BAL,W,5.15

6.3 BUGS Read/Write

Syntax:

Request1: PLXV0,BUGS,R

Request2: PLXV0,BUG,W,<Bugs>

Response1: PLXV0,BUG,W,<Bugs>

Response2: No response.

Description:

Parameter	Description
Bugs	Parameter Bugs is of type float. It is degradation of polar in %.

Example:

Request1: PLXV0,BUGS,R

Response1: PLXV0,BUGS,W,10.5

6.4 VOL Read/Write

Syntax:

Request1: PLXV0,VOL,R

Request2: PLXV0,VOL,W,<Volume>

Response1: PLXV0,VOL,W,<Volume>

Response2: No response.

Description:

Parameter	Description
Volume	Parameter volume is of type float. It is expressed in % from 0 to 100.

Example:

Request1: PLXV0,VOL,R

Response1: PLXV0,VOL,W,70.5

6.5 POLAR Read/Write

Syntax:

Request1: PLXV0,POLAR,R

Request2: PLXV0,POLAR,W,<a>,<b>,<c>,<polar load>,<polar wgt>,<maxw>,<empty>,<pilot>,<name>,<stall>

Response1: PLXV0,POLAR,W,<a>,<b>,<c>,<polar load>,<polar weight>,<max weight >,<empty weight >,<pilot weight >,<name>,<stall>

Response2: No response.

Description:

Parameter	Description
A	Used in polar equation. Type = float
B	Used in polar equation. Type = float
C	Used in polar equation. Type = float
Polar load	This parameter is wing load used in polar. It is expressed in kg/m2. Type = float
Polar weight	This parameter is weight used in polar. It is expressed in kg. Type = int
Max weight	This parameter is maximum weight of glider expressed in kg. Type = int
Empty weight	This parameter is empty weight of glider expressed in kg. Type = int
Pilot weight	This parameter is pilot weight expressed in kg. Type = int
Name	This parameter is polar name. It is type of string. Max 14 characters.
Stall	This parameter is stall speed expressed in m/s. Type = float

Example: /

6.6 CONNECTION Read/Write

Syntax:

Request1: PLXV0,CONNECTION,R

Request2: PLXV0,CONNECTION,W,<direction>

Response1: PLXV0,CONNECTION,W,<direction>

Response2: No response.

Description:

Parameter	Description
Direction	Parameter direction is of type string. It's value can be one of: - DIRECT: Enables direct link with GPS port on all baudrates - VSEVEN: Enables communication with v7 on set baudrate - NANO: Enabled direct link with GPS port on all baudrates

Example:

Request1: PLXV0,CONNECTION,R

Response1:PLXV0,CONNECTION,W,DIRECT

Request2: PLXV0,CONNECTION,W,VSEVEN

6.7 NMEARATE Read/Write

Syntax:

Request1: PLXV0,NMEARATE,R

Request2: PLXV0,NMEARATE,W,<plxvf>,<plxvs>,<lxwp0>,<lxwp1>,<lxwp2>,<lxwp3>,<lxwp5>

Response1: PLXV0,NMEARATE,W,<plxvf>,<plxvs>,<lxwp0>,<lxwp1>,<lxwp2>,<lxwp3>,<lxwp5>

Response2: No response.

Description:

Parameter	Description
Plxvf	This parameter is of type int. It represents rate of plxvf record in Hz. Recommended values: 20, 10, 5, 2, 1

Plxvs	This parameter is of type int. It represents period for plxvs statements in seconds.
Lxwp0	This parameter is of type int. It represents period for LXWP0 statements in seconds. If set to zero this statements will be disabled.
Lxwp1	This parameter is of type int. It represents period for LXWP1 statements in seconds.
Lxwp2	This parameter is of type int. It represents period for LXWP2 statements in seconds.
Lxwp3	This parameter is of type int. It represents period for LXWP3 statements in seconds.
Lxwp5	This parameter is of type int. It represents period for LXWP5 statements in seconds.

Example:

Request1: PLXV0,NMEARATE,R

Response1: PLXV0,NMEARATE,W,10,5,1,60,30,30,0,0

6.8 ELEVATION Read/Write

Syntax:

Request1: PLXV0,ELEVATION,R

Request2: PLXV0,ELEVATION,W,<elevation>

Response1: PLXV0,ELEVATION,W,<elevation>

Response2: No response.

Description:

Parameter	Description
Elevation	This parameter is of type int. It is expressed in meters.

Example:

Request1: PLXV0,ELEVATION,R

Response1: PLXV0,ELEVATION,W,244

6.9 QNH Read/Write

Syntax:

Request1: PLXV0,QNH,R

Request2: PLXV0,QNH,W,<pressure>

Response1: PLXV0,QNH,W,<pressure>

Response2: No response.

Description:

Parameter	Description
Pressure	This parameter is of type int. It represents change in elevation and is expressed in pascals.

Example:

Request1: PLXV0,QNH,R

Response1: PLXV0,QNH,W,101025

6.10 BRGPS Read/Write

Syntax:

Request1: PLXV0,BRGPS,R

Request2: PLXV0,BRGPS,W,<baudrate index>

Response1: PLXV0,BRGPS,W,<baudrate index>

Response2: No response.

Description:

Parameter	Description
Baudrate index	This parameter is of type int. It represents GPS port baudrate index. Possible values: - 0: 4800 baudrate - 1: 9600 baudrate - 2: 19200 baudrate - 3: 38400 baudrate - 4: 57600 baudrate - 5: 115200 baudrate - 6: 230400 baudrate - 7: 256000 baudrate - 8: 500k baudrate - 9: 1M baudrate

Example:

Request2: PLXV0,BRGPS,W,5

This will change gps baudrate to 115200.

6.11 BRPDA Read/Write

Syntax:

Request1: PLXV0,BRPDA,R

Request2: PLXV0,BRPDA,W,<baudrate index>

Response1: PLXV0,BRPDA,W,<baudrate index>

Response2: No response.

Description:

Pda baudrate must not be lower than GPS baud rate!

Parameter	Description
Baudrate index	This parameter is of type int. It represents PDA port baudrate index. Possible values: - 0: 4800 baudrate - 1: 9600 baudrate - 2: 19200 baudrate - 3: 38400 baudrate - 4: 57600 baudrate - 5: 115200 baudrate - 6: 230400 baudrate - 7: 256000 baudrate - 8: 500k baudrate - 9: 1M baudrate

Example:

Request2: PLXV0,BRPDA,W,5

This will change pda baudrate to 115200.

6.12 ALTOFF Read/Write

Syntax:

Request1: PLXV0,ALTOFF,R

Request2: PLXV0,ALTOFF,W,<Alt offset error>,<Alt offset qnh>,<Alt take off>

Response1: PLXV0,ALTOFF,W,<Alt offset error>,<Alt offset qnh>,<Alt take off>

Response2: No response.

Description:

Parameter	Description
Alt offset error	This parameter is of type float.
Alt offset qnh	This parameter is of type float.
Alt take off	This parameter is of type float.

Example: /

7. PLXVF Sentence

Syntax:

PLXVF,<time>,<AccX>,<AccY>,<AccZ>,<Vario>,<IAS>,<PressAlt>,<Mode>,<FlapPosition>

Description:

Parameter	Description
Time	This parameter is of type int. It represents millisecond fraction of a second.
AccX	This parameter is of type float. It represents acceleration in X-axis, fixed to device frame, pointing out of device in G
AccY	This parameter is of type float. It represents acceleration in Y-axis, fixed to device frame, pointing to the side of device in G
AccZ	This parameter is of type float. It represents acceleration in Z-axis, fixed to device frame, pointing down of device in G
Vario	This parameter is of type float. It is expressed in m/s.

IAS	This parameter is of type float. It represents indicated airspeed in m/s.
PressAlt	This parameter is of type float. It represents pressure altitude expressed in meters.
Mode	This parameter is of type byte. It indicates in which mode does the device operates. Possible values are 0 which is VARIO mode and 1 which is SC (speed to fly) mode.
Flap Position	This parameter is a string of the flap position

Example:

PLXVF,1.00,0.87,-0.12,-0.25,90.2,244.3,1,L

8. PLXVS Sentence

Syntax:

PLXVS,<OAT>,<Mode>,<Voltage>,<Igc fr press altitude>>,<FlapPosition>

Description:

Parameter	Description
OAT	This parameter is of type float. It represents outside air temperature expressed in degrees Celsius.
Mode	This parameter is of type byte. It indicates in which mode does the device operates. Possible values are 0 which is VARIO mode and 1 which is SC (speed to fly) mode.
Voltage	This parameter is of type float. It represents voltage expressed in volts.
Igc fr press altitude	This parameter is of type float. It represents pressure altitude that comes from IGC FR to GPS port, expressed in meters.
Flap position	This parameter is a string of the flap position

Example:

PLXVS,23.1,0,12.3,L

9. PLXVTARG Sentence

Syntax:

PLXVTARG,<Waypoint name>,<Waypoint latitude>,<Waypoint longitude>,<Waypoint elevation>

Description:

Parameter	Description
Waypoint name	This parameter is of type string. It represents destination waypoint name.
Waypoint latitude	This parameter is of type string. It represents destination waypoint latitude.
Waypoint longitude	This parameter is of type string. It represents destination waypoint longitude.
Waypoint elevation	This parameter is of type float. It represents destination waypoint elevation expressed in meters.

Example:

PLXVTARG,MARIBOR,4628.80,N,01541.167,E,268.0