Alti-2 Reader Help

Introduction

Altimaster N-series devices data reader version 1.2.0

Welcome

Hi!
My name is Alexey Lobanov.
Some words about me.
Now I'm technical consultant but worked as programmer many years ago.

Programming is my hobby, but the most important hobby is skydive.

In august of 2011 I came back home from Russian State Record and recognize that had to record a lot of jumps in my skydive book.
I use Altimaster N3 to keep logbook and decided to download jumps to computer from it.
After downloading and installing Altimaster NMU utility and drivers I tried to do this but it was big surprise: NMU can't download logbook.
I read on Altimaster web site that I can do it only with Paralog which is not for free and Altimaster do not want to open the communication protocol to another company or person.
I remember that many years ago I've successfully disassembling many MS-DOS programs and decided to correct this "error" with logbook.
The result is this program and description of the communication protocol.

This program is intended to be used worldwide so I decided to use English language for interface and help information.
You can use this program absolutely for free and download new versions from here.

You can contact me by e-mail or on my Facebook page.

What's new

This is the first version of the program.

With **Alti-2 Reader** You can:

- Connect and read all parts of information from Altimaster N-series products.
- Save this information to binary archive.
- Save jump details information (logbook) to ASCII file with delimiters format (CSV-files).
- Print jump details information (logbook).
- Save jumps profiles to ASCII file with delimiters format (CSV-files).
- Make graphs on jumps profiles, which are stored for 256 jumps.
- Make statistics graphs.
- Print and save all graphs in Enhanced Meta File (vector) graphics format.

Reading Alarms tones is not implemented yet.

In settings dialog box You can set many option of program behave such as:

- connection, for example auto connect if device is detected
- communication, for example show data exchange between device and host on the screen
- logging the communication process

- appearance of jumps details information

Small bonus:

- With Tools option You can read any part of Neptune memory and execute some other commands. The results can be printed and saved to ASCII file with delimiters format (CSV-files).

Getting Started

System requirements

This program is developed with MS Visual C# using .NET Framework 4.0 Client package and ported to Ubuntu with MonoDevelop 2.6.
You must have Mono Runtime 2.10.5 installed on your system.
All other necessary DLLs components are installed with the program.

This program is tested on:

- on computer with Intel Core i7 2.7GHz CPU and 2GB RAM under Ubuntu 11.10 32-bit running in VMware virtual machine.

You should install Altimaster USB drivers to communicate with N3 or N3Audio devices.
Drivers can be download from here.

Getting help

I can provide only limited support for this program on this site.
Also You can contact me by e-mail or on my Facebook page.

How to connect

There are two ways to establish connection with device.

- The first way is to invoke Connect command from Device menu or press button on tool bar
- The second way is to check Auto connect on the Connection page in the settings dialog box. If You do this connection will be automatically established when device is detected. If more than one devices are detected You should choose on of them from the settings dialog box Connection page.

How to read data

- You can read device data after connection is established.
  There are two ways to read device data.

  - The first way is to invoke Read Selected command from Device menu or press button on tool bar. You can Read All data or partial by check the parts of data on the tree view pane. But I recommend You read all data because jumps details are depended on the most of these data. To select all data items in tree view pane select root device information check box or invoke command Select All in View menu or press button ✓ on tool bar

  - The second way is to check Read data on connection on the Connection page in the settings dialog box. If You do this data will be automatically read when connection is established.

- Also You can read data previously saved in binary archive. You do not need to establish connection with device to do this. To read data from binary archive invoke Read Archive command from Device menu or press button on tool bar. If You have previously saved all data to binary archive all the functionality of the program to analyze, print and save to ASCII file with delimiters format (CSV-files) are available except working with Tools.

How to save data

- You can save all or partial selected data to binary archive.
  In case Alti-2 Reader doesn't include functionality of writing to device the propose of the binary archive is only to save data for future analyzing, but not for backup.
  By default binary archives get Alti2 extension which is registered as file type in Windows, so You can open archive by double click on it.
  I recommend You to save all parts of data to archive.
  To save selected parts of data invoke Save Selected command from Device menu or press button on tool bar. To save all data to archive select all data items in tree view pane by selecting root device information check box or by invoking command Select All in View menu or by pressing ✓ button on tool bar and then do save selected action. Another way is to invoke Save All command from Device menu.

- You can save jumps details (logbook) to ASCII file with delimiters (CSV-file).
  First line of this file contains names (headers) of jumps details columns, other lines contains information for each jump. You can specify which kind of delimiter to use in Communication page on settings dialog box. It is easy to import such CSV-file to your favorite spreadsheet of database. To save jumps details invoke Save command from

Device menu or press button on tool bar.

- You can save jump profile information. How to open jump profile see here. Alti-2 Reader will store detailed jump information and jump profile to two separated ASCII files with delimiters (CSV-file) and profile graph to Enhanced Meta (vector) graphics file. To save this information invoke Save command from File menu in jump profile window or press button on tool bar in this window.
- When You are working with statistics, jumps profiles graphs or tools the Save command and tool bar buttons are automatically changed.
  - You can save graph to Enhanced Meta (vector) graphics file. To do so invoke Save command from the Device menu or press button on the tool bar.
  - You can save results of invoking commands from Tools window to ASCII file with delimiters (CSV-file). To do so invoke Save command from the Device menu or press button on the tool bar.

How to print data

- With Alti-2 Reader You can print jumps details, jumps profiles graphs, jumps statistics, and jump profile information. Also You can print results of invoking commands from Tools window.
- You can can preview results before printing and setup page size, orientation and margins. I advise You to preview results before printing. If results are not fits well on page change page settings. To change page settings invoke Page setup command from Device menu or Page setup command from File menu if You are in jump profile window.
- Print and Print preview commands and buttons on tool bar automatically changed depending on the information You are working with. If You are in jumps details or in jump profile window the Print preview button changes to . If You are in jumps profiles graphs or in statistics windows the Print preview button changes to . If You are in Tools window the Print preview button changes to .

How to disconnect

- To disconnect from device invoke Disconnect command from Device menu or press button on tool bar

How to view jump profile

- Device store jump profile information for some jumps. For N3 and N2 with the latest firmware profiles are stored for the latest 256 jumps. If You have more jumps in the device the earlier jump profiles are overwrite by latest jump profiles. Each profile store the jump number to which it belongs. Alti-2 Reader detects on this number if jump has valid profile and allows You to open it.
- You can open jump profile window by double click jump in jumps details view. In this window You can see detailed jump information, profile information and graph of the jump. You can save and print all this information.

How to view profiles graphs

How to view and print statistics

Screens

Data items tree view

Device information

Display settings

Alarms

Alarms names

Alarms tones

Dropzone names

Aircraft names

Speed groups

Logbook summary

Jumps details

Jump profile

Jumps profiles graphs

Data exchange

This screen is for information propose only. It shows data as it send and received from device. Remember that data is encrypted.
Yellow digits is output from host to device. Green digits is input from device to host.
To close this screen click on communication avatar on the upper left corner in the screen.

Tools

Statistics

Commands

Device menu

Connect

Disconnect

Read Selected

Read All

Read Archive

Save Selected

Save All

Save

Print

Print Preview

Page Setup

Exit

View menu

Collaps All

Expand All

Select All

Unselect All

Show data exchange

Options menu

Settings

Tools

Statistics

Help menu

Content

Index

Protocol

About

File menu

Save

Print

Print preview

Page setup

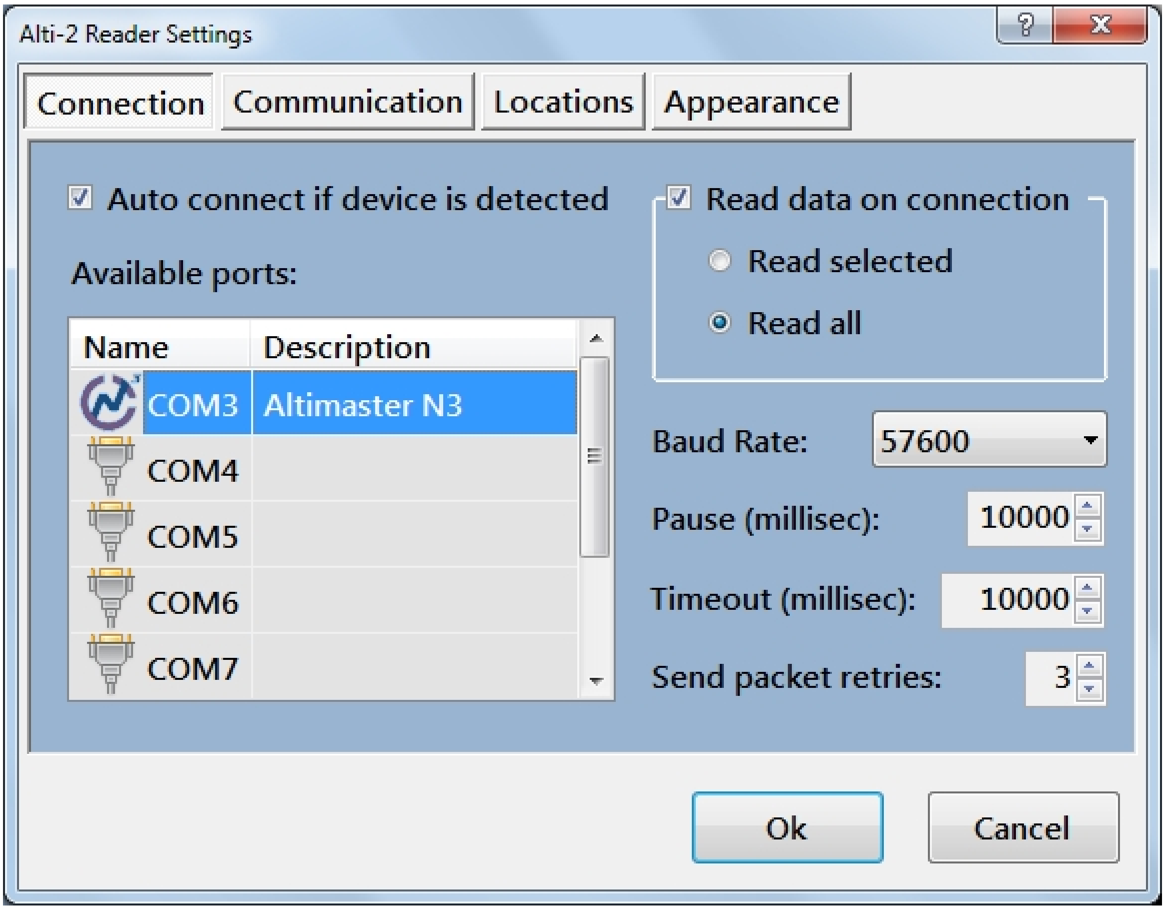
Close

Settings dialog box

To show this dialog box invoke Settings command from Options menu or press Settings button on tool bar.

You can customize program behavior in this dialog box.

Connection page



☑ Auto connect if device is detected

Available ports:

Name	Description
COM3	Altimaster N3
COM4	
COM5	
COM6	
COM7	

☑ Read data on connection

○ Read selected

● Read all

Baud Rate: 57600

Pause (millisec): 10000

Timeout (millisec): 10000

Send packet retries: 3

If You use N3 or N3Audio device it will be automatically detected and shown in **available ports** list.

If **Auto connect** is set connection will be established automatically.

If **Read data on connection** is set device data will read automatically after connection is established.

If You device is not detected or You have problems to read data from it try to change connection parameters shown here.
I have no problems with this settings when test program with my N3.

Auto connect

Read data on connection

Available ports

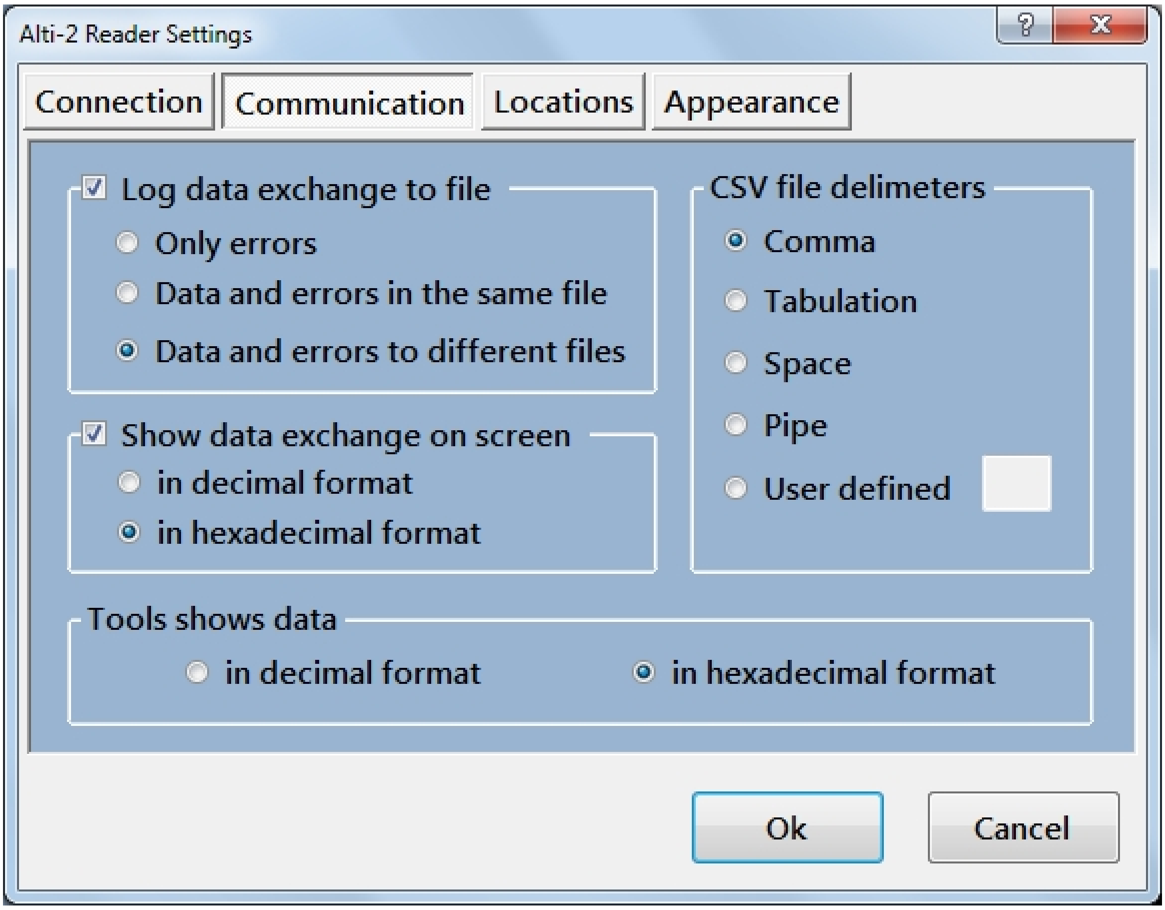
Baud rate

Pause

Timeout

Send packet retries

Communication page



Alti-2 Reader Settings

☑ Log data exchange to file
- ○ Only errors
- ○ Data and errors in the same file
- ● Data and errors to different files

☑ Show data exchange on screen
- ○ in decimal format
- ● in hexadecimal format

CSV file delimeters
- ● Comma
- ○ Tabulation
- ○ Space
- ○ Pipe
- ○ User defined

Tools shows data
- ○ in decimal format
- ● in hexadecimal format

On this page You can set parameters of communication behavior.

You can store detailed logs of communication in text files. To do this check **Log data exchange to file** box. Locations of these files are set on Locations page.

Also You can see data exchange between device and host on the screen. To do it always when communicate with device check **Show data exchange box** and choose digits format.

You can save logbook jumps detailed information to ASCII text file with delimeters. You can choose here what kind of delimeter to use.

I provide tool for discover device memory. Here You can set digits format showing data in this tool.

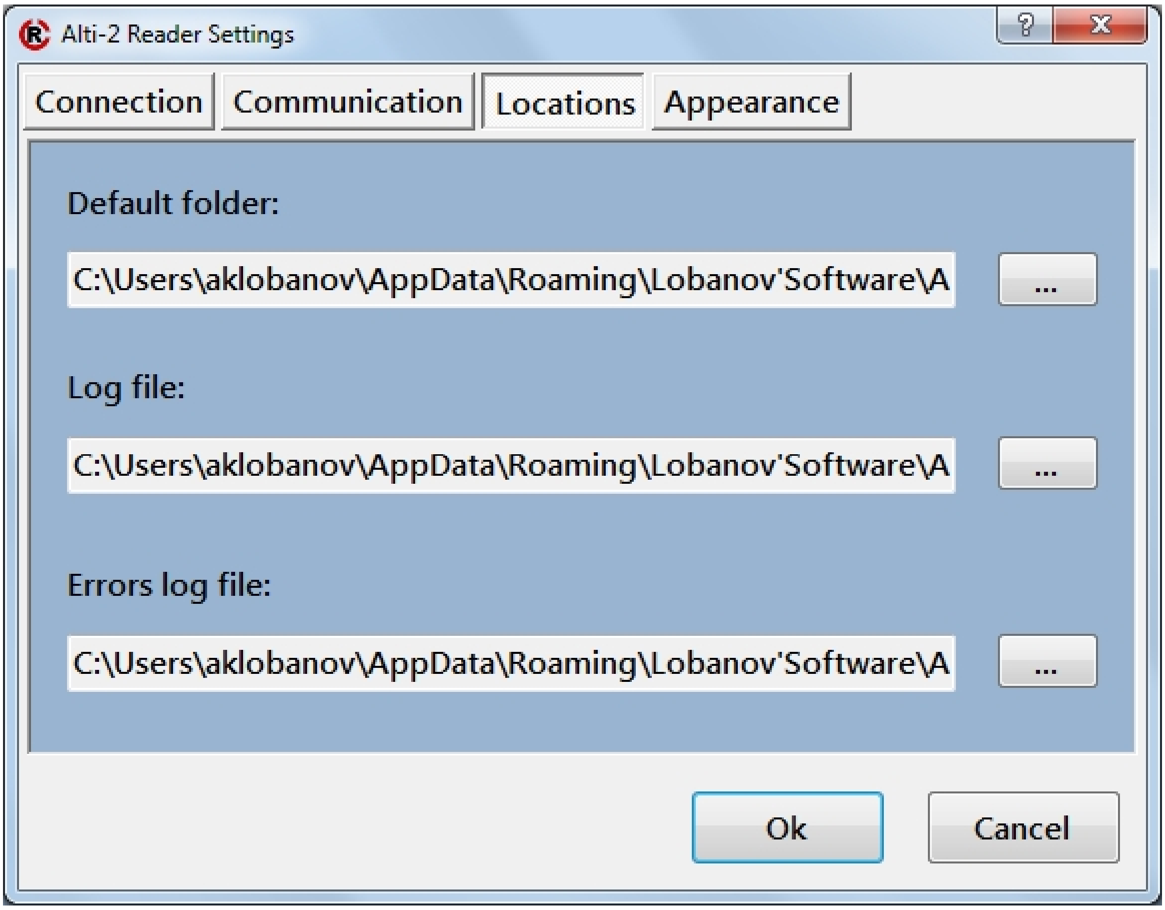
Log data exchange to file

Shows data exchange on screen

CSV file delimeters

Tools shows data

Locations page



Default folder:

C:\Users\aklobanov\AppData\Roaming\Lobanov'Software\A

Log file:

C:\Users\aklobanov\AppData\Roaming\Lobanov'Software\A

Errors log file:

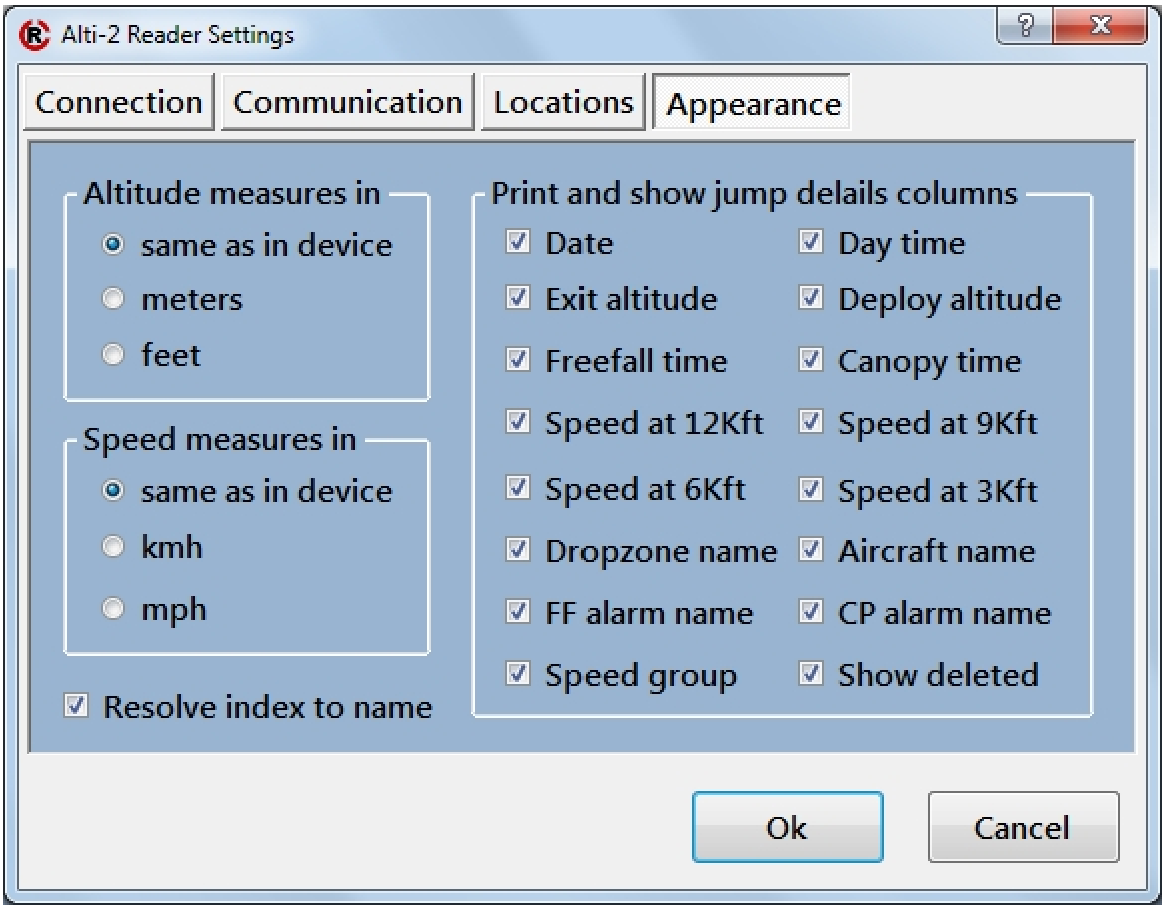
C:\Users\aklobanov\AppData\Roaming\Lobanov'Software\A

Default folder

Log file

Errors log file

Appearance page



Altitude measures in

● same as in device
○ meters
○ feet

Speed measures in

● same as in device
○ kmh
○ mph

☑ Resolve index to name

Print and show jump delails columns

☑ Date	☑ Day time
☑ Exit altitude	☑ Deploy altitude
☑ Freefall time	☑ Canopy time
☑ Speed at 12Kft	☑ Speed at 9Kft
☑ Speed at 6Kft	☑ Speed at 3Kft
☑ Dropzone name	☑ Aircraft name
☑ FF alarm name	☑ CP alarm name
☑ Speed group	☑ Show deleted

Altitude measures

Speed measures

Resolve index to name

Print and show jumps details columns

Altimaster communication protocol

I'm not using any inside information from Altimaster to discover this protocol.

The only way I do was:

- decompiling NMU utility
- analyze with port sniffer data exchange between NMU/Paralog and my N3.

Also I found useful information here.

I divide protocol description in tree parts:

The communication order - describe commands

Encryption algorithm - describe encryption used in N-series devices

Data structures - describe N-series device memory data structures

The communication order

First of all you should open the COM port the device is connected to. I use for this propose **USB.DLL** that can be found in the **NMU** installation directory. This DLL is .NET assembly and can be easy added to your own program. To communicate by IRDA port use **IrDAComms.DLL** which you can found in the same directory. These DLLs contains **_open** function to open port.

After COM port is opened make pause for about 10 seconds. Than send the first command.

All commands have identical formats. The first byte contains the length of the command packet. The length does not include this first byte and the last byte which contains the checksum of the packet. Checksum is calculated as sum of packet bytes values by module of 256. This sum does not include the first length byte and the last checksum byte.

BYTE 0	BYTES 1..N	BYTE N+1
Packet length	Command packet	Checksum of BYTES 1..N by module 256

I am not discover parameters of the write commands in case to do not damage my N3. So I know parameters only of tree commands: get type 0 record, read memory and end communication commands.

Get type 0 record command

The first command packet is very simple and contains only one byte. This command is send as ASCII text string “01 80 80” and is recognized by most of all Altimaster devices. I recommend you to send it without spaces, but it is recognized with spaces too.

BYTES	0	1	2	3	4	5
ASCII	0	1	8	0	8	0
HEX	30	31	38	30	38	30

In response for this command Altimaster devices send Type 0 record. It seems that this type of communication using ASCII strings representing HEX digits was used in devices with firmware prior to 2.6.3. But now all other command are represented in bytes and encrypted.

On the base of Type 0 record bytes is generated encryption key which is used to encrypt and decrypt all packages sent to and received from N2, N3 and N3A devices. All packages are even to 32 bytes length and if necessary are added by zeros. For example read command package contains 7 bytes plus length and checksum bytes - total 9 bytes. These 9 bytes are added with zero bytes to 32 than encrypted and send to device.

Read memory command

Read memory command consist of one byte code decimal 160 (A0 hexadecimal), 4 bytes of memory address and 2 bytes of the requested memory block length.

BYTE	0	1	2	3	4	5	6	7	8
	packet length	command code	Memory address low byte	Memory address middle byte	Memory address high byte	Memory address highest byte	Memory block length low byte	Memory block length low byte	Check sum
DEC	7	160							
HEX	7	A0							

Memory address is DWORD and memory block length is WORD stored in Little-Endian format.

In response to this command device send two acknowledgements 49 decimal (31 hexadecimal) and 53 decimal (35 hexadecimal). Than device send requested memory block divided in packages of 32 bytes length. When you successfully receive each 32-byte package you should send acknowledgement 49 decimal (31 hexadecimal) to device. All packages are encrypted and you need to decrypt them before use the data. The first received packet in first 4 bytes contains the memory address you requested, so you receive requested memory block plus 4 more bytes.

Addresses and lengths of data structures I've discovered you can see here.

But you can read any address and length you need. Program contains tool for it. For example Paralog reads only bytes with "Total Physical Jumps" data before reading logbook instead reading all logbook summary information.

End communication command

On ending communication send command 175 without parameters, flash device and that send command 03 without parameters in form of ASCII string.

Keep alive command

NMU uses command 164 without parameters to keep device alive. I also include this command in my program.

Other commands

command			description
DEC	HEX	BIN	
03	03	00000011	The last command for ending communication. It is sent in the form of ASCII string representing HEX digits in the same manner as the first (get “Type 0” record) command. This command is sent after command 175
48	30	00110000	Acknowledgment that communication is ABORTED
49	31	00110001	Acknowledgment that command recognized successfully
50	32	00110010	Acknowledgment that the length of the send packet is incorrect
51	33	00110011	Acknowledgment that the checksum of the send packet is incorrect
52	34	00110100	Acknowledgment to repeat packet (command)
53	35	00110101	Acknowledgment that device/host is ready to send/receive packets
54	36	00110110	Acknowledgment that received command is not recognized
55	37	00110111	Acknowledgment that received command has incorrect syntax
56	38	00111000	Acknowledgment that there is error writing to EEProm/Fram
57	39	00111001	Acknowledgment that there is Flash Erase error
58	3A	00111010	Acknowledgment that the requested memory address is out of bounds
59	3B	00111011	Acknowledgment that there is Flash write error
60	3C	00111100	Acknowledgment that there is no Boot loader present to respond to request

command			description
DEC	HEX	BIN	
128	80	10000000	Command to get “Type 0” record. It is sent in the form of ASCII string representing HEX digits.
160	A0	10100000	Command to read memory block
164	A4	10100100	NMU uses this command to keep device alive
175	AF	10101111	Command to end communication (may be for it. I use it without any parameters as in NMU)
176	B0	10110000	Command to write memory block
-1	FF	11111111	Acknowledgment that there is communication error

Encryption algorithm

All send packages include commands must be encrypted.

All received packages must be decrypted.

Exceptions are:

- acknowledgments
- get type 0 command
- end communication 03 command

You should generated encryption key using type 0 record bytes.

How to generate encryption key

Encryption key consists of 4 DWORDs. These DWORDs are formed from “Type 0” record bytes and explicit values.

DWORDs	0				1				2				3			
DWORD BYTES	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
BYTE # or VALUE	BYTE	BYTE	BYTE	VALUE	BYTE	BYTE	BYTE	BYTE	BYTE	VALUE	BYTE	BYTE	BYTE	BYTE	VALUE	BYTE

Type 0 records byte or explicit values (decimal)	24	26	8	78	6	25	23	13	10	117	7	22	9	11	126	21

How to encrypt packet

If packet length is less than 32 bytes expand it to this size by adding zeros. If packet length is more than 32 bytes divide it to 32-bytes packets and if the length is not even to 32 expand last packet to 32 bytes by adding zeros. Convert each 32-packet to DWORD array. Remember that bytes are stored in Little-Endian format. Take pair of DWORD and encrypt it with code placed bellow. Than next pair, etc. Convert DWORD array to 32-byte packet where bytes are stored in Little-Endian format. Packet is encrypted.

```
UInt32 U;  // first DWORD from the pair
UInt32 U1; // second DWORD from the pair
UInt32 U2 = 0;
for (int i = 16; i > 0; i--)
{
    U += (((U1 << 4) ^ (U1 >> 5)) + U1) ^ (U2 + KEY[U2 & 3]);
    U2 += 0x9E3779B9;
    U1 += (((U << 4) ^ (U >> 5)) + U) ^ (U2 + KEY[(U2 >> 11) & 3]);
}
```

Encrypted pair is in U and U1 DWORDs, KEY is array of four DWORDs with encryption key generated in order I've explained above.

How to decrypt packet

Decryption is made in the same way as encryption, but the code is different.

```
UInt32 U; // first DWORD of the pair
UInt32 U1; // second DWORD of the pair
UInt32 U2 = 0xE3779B90;
for (int i = 16; i > 0; i--)
{
     U1 -= (((U << 4) ^ (U >> 5)) + U) ^ (U2 + KEY[(U2 >> 11) & 3]);
     U2 -= 0x9E3779B9;
     U -= (((U1 << 4) ^ (U1 >> 5)) + U1) ^ (U2 + KEY[U2 & 3]);
}
```

Decrypted pair is in U and U1 DWORDs, KEY is array of four DWORDs with encryption key generated in order I've explained above.

Data structures

Analyzing NMU code I've discovered this data structures:

- Type 0 record
- Device settings
- Drop zones names
- Aircrafts names
- Speed groups
- Alarms settings
- Alarms names
- Alarm tone directory
- Alarm tone data
- Jumps summary (logbook summary)
- Jumps details (logbook)

Addresses in memory

In memory	offset		length	
Structure name	DEC	HEX	DEC	HEX
Jumps summary	10	0x000A	30	0x001E
Device Settings	44	0x002C	13	0x000D
Speed Groups	58	0x003A	26	0x001A
Drop zones Names	84	0x0054	322	0x0142
Aircraft Names	406	0x0196	322	0x0143
Alarms Names	728	0x02D8	322	0x0144
Alarm Tone Directory	1050	0x041A	18	0x0012
Alarm Tone Data	1068	0x042C	160	0x00A0
Alarms Settings	1228	0x04CC	84	0x0054
Jumps details	1312	0x0520	7766	0x1E56

Type 0 record

The length of record may vary depending on Neptune device product and firm ware (software) version.

BYTE	BITS	VALUE	DESCRIPTON
0	0-7		Packet length
1	0-7	0	Packet type
2	0-7	3 = N3	Communication type
3	4-7		Software major version number
	0-3		Software minor version number
4	0-7		Software revision number
5	0-7	ASCII code	Serial number index (first letter)
6-13	0-7	ASCII code	Serial number digits, some last may be spaces (0x20)
14	0-7	1 = N3/N3A 3,4,5,6,7 = N2	Hardware revision number
15	07	0 = Unknown 1 = Neptune 2 = Wave 3 = Tracker 4 = Data Logger 5 = N3 6 = N3A	Product type
16	0-7		NVRAM configuration
17-20	0-7		?
21-26	0-7		Used in KEY generation with bytes of Serial number.
27-30	0-7		?
31	0-7	Checksum	Sum bytes from 1 to 30 mod 256. In my case this is the last byte

Device settings

In my program it is named Display settings in the same way as in N3 device.

BYTE	BITS	VALUE	DESCRIPTON
0	0-7	0 = feet	Altitude measure

BYTE	BITS	VALUE	DESCRIPTON
		1 = meters	
1	0-7	0 = mph 1 = kmh	Speed measure
2	0-7	0 = Fahrenheit 1 = Celsius	Temperature measure
3	0-7	0 = not flipped 1 = flipped	Display view mode
4	0-7	0 = disabled 1 = enabled	Log book usage
5	0-7	0 = 12 hour 1 = 24 hour	Time format
6	0-7	0 = US 1 = International	Date format
7	0-7	0 = disabled 1 = enabled	Canopy display mode
8	0-7	0 = show time 1 = show altitude	Climb display mode
9	0-7	0 in my N3	?
10	0-7		Display contrast value
11	0-7	0x5B in my N3	?
12	0-7	0 = normal 1 = loud	Canopy alarms mode

Drop zones names

For name used 10 bytes, so each block contains two names. Names are stored as ASCII values using bits 0-6 of each byte. High bit (number 7) of the name's first byte (number 0) is a flag which is indicating that the name is hidden. High bit (number 7) of the name's second byte (number 1) is a flag which is indicating that the name is used.

BYTE	BITS	VALUE	DESCRIPTON
0	0-7		Checksum
1	0-7		Count
2-21	0-159	ASCII or 0x00	Block 0: Drop zone name in ASCII
22-41	0-159	ASCII or 0x00	Block 1: Drop zone name in ASCII
42-61	0-159	ASCII or 0x00	Block 2: Drop zone name in ASCII
62-81	0-159	ASCII or 0x00	Block 3: Drop zone name in ASCII
82-101	0-159	ASCII or 0x00	Block 4: Drop zone name in ASCII
102-121	0-159	ASCII or 0x00	Block 5: Drop zone name in ASCII
122-141	0-159	ASCII or 0x00	Block 6: Drop zone name in ASCII
142-161	0-159	ASCII or 0x00	Block 7: Drop zone name in ASCII
162-181	0-159	ASCII or 0x00	Block 8: Drop zone name in ASCII
182-201	0-159	ASCII or 0x00	Block 9: Drop zone name in ASCII
202-221	0-159	ASCII or 0x00	Block 10: Drop zone name in ASCII
222-241	0-159	ASCII or 0x00	Block 11: Drop zone name in ASCII
242-261	0-159	ASCII or 0x00	Block 12: Drop zone name in ASCII
262-281	0-159	ASCII or 0x00	Block 13: Drop zone name in ASCII
282-301	0-159	ASCII or 0x00	Block 14: Drop zone name in ASCII
302-321	0-159	ASCII or 0x00	Block 15: Drop zone name in ASCII

Aircrafts names

For name used 10 bytes, so each block contains two names. Names are stored as ASCII values using bits 0-6 of each byte. High bit (number 7) of the name's first byte (number 0) is a flag which is indicating that the name is hidden. High bit (number 7) of the name's second byte (number 1) is a flag which is indicating that the name is used.

BYTE	BITS	VALUE	DESCRIPTON
0	0-7		Checksum
1	0-7		Count

BYTE	BITS	VALUE	DESCRIPTON
2-21	0-159	ASCII or 0x00	Block 0: Aircraft name in ASCII
22-41	0-159	ASCII or 0x00	Block 1: Aircraft name in ASCII
42-61	0-159	ASCII or 0x00	Block 2: Aircraft name in ASCII
62-81	0-159	ASCII or 0x00	Block 3: Aircraft name in ASCII
82-101	0-159	ASCII or 0x00	Block 4: Aircraft name in ASCII
102-121	0-159	ASCII or 0x00	Block 5: Aircraft name in ASCII
122-141	0-159	ASCII or 0x00	Block 6: Aircraft name in ASCII
142-161	0-159	ASCII or 0x00	Block 7: Aircraft name in ASCII
162-181	0-159	ASCII or 0x00	Block 8: Aircraft name in ASCII
182-201	0-159	ASCII or 0x00	Block 9: Aircraft name in ASCII
202-221	0-159	ASCII or 0x00	Block 10: Aircraft name in ASCII
222-241	0-159	ASCII or 0x00	Block 11: Aircraft name in ASCII
242-261	0-159	ASCII or 0x00	Block 12: Aircraft name in ASCII
262-281	0-159	ASCII or 0x00	Block 13: Aircraft name in ASCII
282-301	0-159	ASCII or 0x00	Block 14: Aircraft name in ASCII
302-321	0-159	ASCII or 0x00	Block 15: Aircraft name in ASCII

Speed groups

Three speed groups each of four bands. Each group occupies 8-byte record. Each record consists of four pair of bytes, one for each band. Fist byte of pair contains start value, second contains stop value.

BYTE	BITS	VALUE	DESCRIPTON
0	0-7		?
1	0-7	0 – default 1 – group 1 2 – group 2 3 – group 3	Selected group

BYTE	BITS	VALUE	DESCRIPTON
2	0-7		Start value band 1 group 1
3	0-7		Stop value band 1 group 1
4	0-7		Start value band 2 group 1
5	0-7		Stop value band 2 group 1
6	0-7		Start value band 3 group 1
7	0-7		Stop value band 3 group 1
8	0-7		Start value band 4 group 1
9	0-7		Stop value band 4 group 1
10	0-7		Start value band 1 group 2
11	0-7		Stop value band 1 group 2
12	0-7		Start value band 2 group 2
13	0-7		Stop value band 2 group 2
14	0-7		Start value band 3 group 2
15	0-7		Stop value band 3 group 2
16	0-7		Start value band 4 group 2
17	0-7		Stop value band 4 group 2
18	0-7		Start value band 1 group 3
19	0-7		Stop value band 1 group 3
20	0-7		Start value band 2 group 3
21	0-7		Stop value band 2 group 3
22	0-7		Start value band 3 group 3
23	0-7		Stop value band 3 group 3
24	0-7		Start value band 4 group 3
25	0-7		Stop value band 4 group 3

Alarms settings

Consist of eight 10-byte arrays presiding by four bytes. First two bytes are unknown for me. Second two bytes contain array index of active free fall and canopy alarms respectively. If the high bit (7) of these bytes is set means than free fall (or canopy) alarms are disabled.

BYTE		BITS	VALUE	DESCRIPTON
0		0-7		?
1		0-7		?
2		0-7	If BIT 7 is set all free fall alarms are disabled	Active alarm array number for free fall.
3		0-7	If BIT 7 is set all canopy alarms are disabled	Active alarm array number for canopy.
4-13 10 BYTE ARRAY	0	2-7		Alarm name index
		0-1	0 = free fall 1 = canopy	Free fall/canopy indicator
	1	0-7		Alarm tone index for Alarm 1
	2	0-7		Alarm tone index for Alarm 2
	3	0-7		Alarm tone index for Alarm 3

	4-5	0-15		Alarm altitude 1.
				Resulted altitude calculated
				in meters
				for free fall: (round (100 * (value/2)))/100
				for canopy: (round (10 * (value/2)))/10
				in feet
				for free fall: round(((((value / 2) * 1000) / 25.4) / 12) / 100) *100
				for canopy: round(((((value / 2) * 1000) / 25.4) / 12) / 10)*10

	6-7	0-15		Alarm altitude 2
	8-9	0-15		Alarm altitude 3
14-23				Alarm 2: 10 – BYTE ARRAY the same as above

BYTE	BITS	VALUE	DESCRIPTON
24-33			Alarm 3: 10 – BYTE ARRAY the same as above
34-43			Alarm 4: 10 – BYTE ARRAY the same as above
44-53			Alarm 5: 10 – BYTE ARRAY the same as above
54-63			Alarm 6: 10 – BYTE ARRAY the same as above
64-73			Alarm 7: 10 – BYTE ARRAY the same as above
74-83			Alarm 8: 10 – BYTE ARRAY the same as above

Alarms names

For name used 10 bytes, so each block contains two names. Names are stored as ASCII values using bits 0-6 of each byte. High bit (number 7) of the name's first byte (number 0) is a flag which is indicating that the name is hidden. High bit (number 7) of the name's second byte (number 1) is a flag which is indicating that the name is used.

BYTE	BITS	VALUE	DESCRIPTON
0	0-7		Checksum
1	0-7		Count
2-21	0-159	ASCII or 0x00	Block 0: Alarm name in ASCII
22-41	0-159	ASCII or 0x00	Block 1: Alarm name in ASCII
42-61	0-159	ASCII or 0x00	Block 2: Alarm name in ASCII
62-81	0-159	ASCII or 0x00	Block 3: Alarm name in ASCII
82-101	0-159	ASCII or 0x00	Block 4: Alarm name in ASCII
102-121	0-159	ASCII or 0x00	Block 5: Alarm name in ASCII
122-141	0-159	ASCII or 0x00	Block 6: Alarm name in ASCII
142-161	0-159	ASCII or 0x00	Block 7: Alarm name in ASCII
162-181	0-159	ASCII or 0x00	Block 8: Alarm name in ASCII
182-201	0-159	ASCII or 0x00	Block 9: Alarm name in ASCII
202-221	0-159	ASCII or 0x00	Block 10: Alarm name in ASCII
222-241	0-159	ASCII or 0x00	Block 11: Alarm name in ASCII

BYTE	BITS	VALUE	DESCRIPTON
242-261	0-159	ASCII or 0x00	Block 12: Alarm name in ASCII
262-281	0-159	ASCII or 0x00	Block 13: Alarm name in ASCII
282-301	0-159	ASCII or 0x00	Block 14: Alarm name in ASCII
302-321	0-159	ASCII or 0x00	Block 15: Alarm name in ASCII

Alarm tone directory

I am not discover it yet.

Alarm tone data

I am not discover it yet.

Jumps summary

All WORDs and DWORDs are stored in Little Endian format: low byte first, high byte second, etc.

BYTE	BITS	VALUE	DESCRIPTON
0-1	0-15	0x04DC in my N3	?
2-3	0-15		Number of jumps since last odometer reset
4-5	0-15		Total physical jumps stored (include deleted jumps)
6-7	0-15		Total jumps (total physical jumps exclude deleted)
8-11	0-31		Total free fall time in seconds
12-15	0-31		Total time under canopy in seconds
16-17	0-15		Next jump number
18-19	0-15		Top jump number (the most resent jump number)
20-23	0-31	0x00610161 in my N3, 0x0161 is total physical jumps in my N3	?
24-25	0-15		Current drop zone name index
26-27	0-15		Current aircraft name index

BYTE	BITS	VALUE	DESCRIPTON
28-29	0-15	0 = off 1 = on	Student mode

Maximum of Total Physical Jumps depending on HW revision number

HW revision	Max Total Physical Jumps
1	2900
6	1600
7	2900
Other	149

Jumps details

Jumps are stored in logbook as sequence of 22-bytes records. Deleted jumps are not physically deleted but are marked as deleted. Each record is representing one stored jump. Information in the record is sequences of bits which are described in table below. It is surprise but I can't found in this record “Average speed” which my N3 shows.

WORD	BYTE	BIT	SIZE in BITS	VALUE	DESCRIPTION
0	0	0	16	jump number	
		1			
		2			
		3			
		4			
		5			
		6			
		7			
	1	8			
		9			
		10			
		11			
		12			
		13			
		14			
		15			
1	2	16	7	month quantity	Quantity of months from 2007 year. To calculate the year of jump divide this value minus 1 on 12 and add 2007. To calculate the month of jump take the
		17			
		18			
		19			
		20			
		21			
		22			

WORD	BYTE	BIT	SIZE in BITS	VALUE	DESCRIPTION
					module (%) of 12 on this value
		23	1	deleted	0 – not deleted 1 - deleted
	3	24	8	Free fall alarm name index	If high bit is set to 1 this means that free fall alarms are deactivated
		25			
		26			
		27			
		28			
		29			
		30			
		31			
2	4	32	10	Free fall time in seconds	
		33			
		34			
		35			
		36			
		37			
		38			
		39			
	5	40			
		41			
		42	6	software minor version number	
		43			
		44			
		45			
		46			
		47			
3	6	48	6	Minutes of the day of the jump	
		49			
		50			
		51			
		52			
		53			
		54	5	Hour of the day of the jump	
		55			
	7	56			
		57			
		58			
		59	4	software major version number	
		60			
		61			
		62			
		63	1	Hi bit of aircraft name index	
4	8	64	7	speed on 3Kft altitude	Stored in meters per second. To calculate in KMH multiply on 3.6. To calculate in MPH multiply on 2.236936
		65			
		66			
		67			
		68			

WORD	BYTE	BIT	SIZE in BITS	VALUE	DESCRIPTION
		69			
		70			
		71	7	speed on 6Kft altitude	Stored in meters per second. To calculate in KMH multiply on 3.6. To calculate in MPH multiply on 2.236936
	9	72			
		73			
		74			
		75			
		76			
		77			
		78	7	speed on 9K feet altitude	Stored in meters per second. To calculate in KMH multiply on 3.6. To calculate in MPH multiply on 2.236936
		79			
5	10	80			
		81			
		82			
		83			
		84			
		85	7	speed on 12K feet altitude	Stored in meters per second. To calculate in KMH multiply on 3.6. To calculate in MPH multiply on 2.236936
		86			
		87			
	11	88			
		89			
		90			
		91			
		92	4	Lo bits of aircraft name index	
		93			
		94			
		95			
6	12	96	10	Exit altitude	Stored as number of 2hPa. To calculate in meters multiply on 16. To calculate in feet multiply on 52.4934
		97			
		98			
		99			
		100			
		101			
		102			
		103			
	13	104			
		105			
		106	5	Day of the jump	
		107			
		108			
		109			
		110			
		111	1	Lo bit of Speed Group number	Zero speed group number means Default speed group
7	14	112	10	Deploy altitude	
		113			
		114			

WORD	BYTE	BIT	SIZE in BITS	VALUE	DESCRIPTION
		115			
		116			
		117			
		118			
		119			
	15	120			
		121			
		122	4	Drop zone name index	
		123			
		124			
		125			
		126	1	not used	
		127	1	Hi bit of Speed Group number	Zero speed group number means Default speed group
8	16	128	12	canopy time in seconds	
		129			
		130			
		131			
		132			
		133			
		134			
		135			
	17	136			
		137			
		138			
		139			
		140	2	Hi bits of canopy alarm name index	If set to 1 it means that canopy alarms are deactivated
		141			
		142	2	Hi bits of LT index	It is index to jump profile table
		143			
9	18	144	10	Drop zone altitude	
		145			
		146			
		147			
		148			
		149			
		150			
		151			
	19	152			
		153			
		154	6	Lo bits of LT index	It is index to jump profile table
		155			
		156			
		157			
		158			
		159			
10	20	160	4	software revision number	

WORD	BYTE	BIT	SIZE in BITS	VALUE	DESCRIPTION
		161			
		162			
		163			
		164	4	Lo bits of canopy alarm name index	
		165			
		166			
		167			
	21	168	8	Max speed	Always 0 in my N3 and in N2 which I've tested
		169			
		170			
		171			
		172			
		173			
		174			
		175			

Jump profile