ALTI-2 COMMUNICATION PROTOCOL

The communication order

Get Type 0 record

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

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. See below the description of this 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.

I’ve not discover write commands in case to do not damage my N3. So I know only two more commands: to read memory and to end communication command.

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 in table below.

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

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

Other commands and acknowledgements

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	Acknowledgement that communication is ABORTED
49	31	00110001	Acknowledgement that command recognized successfully
50	32	00110010	Acknowledgement that the length of the send packet is incorrect
51	33	00110011	Acknowledgement that the checksum of the send packet is incorrect
52	34	00110100	Acknowledgement to repeat packet (command)

command			description
DEC	HEX	BIN	
53	35	00110101	Acknowledgement that device/host is ready to send/receive packets
54	36	00110110	Acknowledgement that received command is not recognized
55	37	00110111	Acknowledgement that received command has incorrect syntax
56	38	00111000	Acknowledgement that there is error writing to EEProm/Fram
57	39	00111001	Acknowledgement that there is Flash Erase error
58	3A	00111010	Acknowledgement that the requested memory address is out of bounds
59	3B	00111011	Acknowledgement that there is Flash write error
60	3C	00111100	Acknowledgement that there is no Boot loader present to respond to request
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	??? I don’t know but it often seen in NMU protocol log. It seems that NMU checking N3 is alive with this command
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	Acknowledgement that there is communication error

Encryption algorithm

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

Data structures addresses in device 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
DZ Names	84	0x0054	322	0x0142
AC Names	406	0x0196	322	0x0143
Alarm Names	728	0x02D8	322	0x0144

Alarm Tone Directory	1050	0x041A	18	0x0012
Alarm Tone Data	1068	0x042C	160	0x00A0
Alarm Settings	1228	0x04CC	84	0x0054
Jumps	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 record

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 1 = meters	Altitude measure

BYTE	BITS	VALUE	DESCRIPTON
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

Log book summary info record

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)

BYTE	BITS	VALUE	DESCRIPTON
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
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

Drop zone's names array

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

BYTE	BITS	VALUE	DESCRIPTON
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

Aircraft’s names array

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: 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

BYTE	BITS	VALUE	DESCRIPTON
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

Alarm’s names array

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 settings record

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

BYTE	BITS	VALUE	DESCRIPTON
14-23			Alarm 2: 10 - BYTE ARRAY the same as above
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

Speed group record

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
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

BYTE	BITS	VALUE	DESCRIPTON
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

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
		17			
		18			

WORD	BYTE	BIT	SIZE in BITS	VALUE	DESCRIPTION
		19			2007. To calculate the month of jump take the module (%) of 12 on this value
		20			
		21			
		22			
		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			

WORD	BYTE	BIT	SIZE in BITS	VALUE	DESCRIPTION
		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			
		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			

WORD	BYTE	BIT	SIZE in BITS	VALUE	DESCRIPTION
		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	I don’t know the propose of LT parameter so I don’t use it in my program
		143			
9	18	144	10	Drop zone altitude	I don’t show this parameter in my program
		145			

WORD	BYTE	BIT	SIZE in BITS	VALUE	DESCRIPTION
		146			
		147			
		148			
		149			
		150			
		151			
	19	152			
		153			
		154	6	Lo bits of LT index	I don’t know the propose of LT parameter so I don’t use it in my program
		155			
		156			
		157			
		158			
		159			
10	20	160	4	software revision number	
		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 so I don’t use it in my program
		169			
		170			
		171			
		172			
		173			
		174			
		175			