Neptune Protocol

Because Alti-2 won't release the spec, I have started a project to reverse-engineer the Neptune communications protocol. The goal is to be able to download the logbook entries without having to use Paralog.

Disclaimer

Using the information on this web site, or any software derived from the information on this web site, could cause any or all of the following:

- Damage to your Neptune
- Damage to your computer
- Damage to your reputation

Please do not use this information and/or software on this site unless you are willing to ruin your Neptune in the process.

It is also possible that using the information on these pages could introduce subtle errors into the operation of a Neptune. While this is extremely unlikely, anyone using a Neptune that has interacted with unofficial software must consider themselves a test jumper.

Reverse Engineering

All of the information on this site was obtained by sniffing the serial communications between the Neptune and a PC running the demo version of Paralog. I did not and will not disassemble or decompile the source code to Paralog or the Neptune firmware. Because of this, I believe I am in the clear legally. There is a long history such "black box" practices in the computer industry (the Linux operating system, for example, wouldn't exist without it).

The Neptune Protocol

Data comes across the serial connection as ASCII strings. The strings are hexadecimal digits, plus spaces and carriage return-linefeed pairs. For example, a string might look like "0C 01 92 00 0A 92 00 E2 1D 00 00 91 00 BF \r\n\". This happens to be a Log Summary.

Communications Summary

Data is sent and received in packets. The first hex byte in a packet is the packet length. This is followed by a hex byte that indicates the type of packet. Then there are zero or more data fields. The packet is ended by a checksum. Packets sent by the Neptune are usually followed by a CR/

LF. Packets sent by Paralog are not. At this time, I do not know if the CR/LFs (or lack thereof) are mandatory or optional.

Here are a few example packets:

“01 80 80” - This is a packet of length 1, type 80, no data, and a checksum of 80 (any packet with no data will always have a checksum equal to it’s type).

“17 02 91 00 20 10 0B 09 05 03 42 36 34 31 14 40 10 D0 03 11 05 00 3B 00 44” This packet has length of 23 (hex 17), type 02, 22 data bytes, and a checksum of 44.

Known Packet Types

- 00 - Device Info
- 01 - Log Summary
- 02 - Jump Summary
- 03 - Completion
- 04 - Detailed Log Event
- 05 - Detailed Log Header
- 06 - Detailed Log Data
- 80 - Go Ahead
- A4 - Unknown Command
- A6 - Time Set
- Non-Standard Reply

Typical Session

This is a summary of a typical session between the Neptune and a PC.

- Paralog sends an Go Ahead packet, letting the Neptune know that it is ready.
- Neptune responds with a Device Info packet.
- Paralog sends a Unknown Command packet.
- Neptune responds with a Non-Standard Reply
- Paralog sends an optional Time Set packet.
- Neptune replies with a Go Ahead if Time Set has been sent.
- Paralog sends a Go Ahead
- Neptune sends a Log Summary
- Neptune sends a series of Jump Summary packets.
- Neptune sends a Detailed Log Header, followed by a bunch of Detailed Log Data packets for each jump with detailed logging. Some type Detailed Log Event packets are mixed in.
- Neptune sends a Completion
- Connection closes.

Connection

The connection to the Neptune is serial over IRDA at 9600 baud, 8-N-1, no flow control. It is possible to use a simple terminal program to perform basic communication with the Neptune.

The upgrade guide at Alti-2.com has details on how to configure a Windows computer to communicate with the Neptune, including setting up the irCOMM2k driver and creating a virtual serial port.

Timing

It appears that the connection with the device is not fully completed until the warbling tone is completed. Any data sent between opening the port in software and the end of the tone seems to disappear into the bit bucket. I have been using a 10 seconds delay between opening the port and sending data.

What paralog might be doing: Sending spaces and waiting until the send buffer is cleared? Try this.

Also, I have been using a 100 msec delay after sending a command before attempting to read from the port. This seems to reduce or eliminate hangs.

Log Summary Packet

This packet type contains a summary of the log information stored in the Neptune.

Example log summary packet: “0C 01 92 00 0A 92 00 E2 1D 00 00 91 00 BF”

- Byte 0: Packet length
- Byte 1: Packet type
- Byte 2: Jumps on the unit (as displayed on the log summary screen)
- Byte 3: Jumps on the unit high byte.
- Byte 4: Number of jumps with detailed logging
- Byte 5: Number of the highest jump on the unit (low byte)
- Byte 6: Number of the highest jump on the unit (high byte)
- Byte 7: Total freefall time in seconds (low byte)
- Byte 8: Total freefall time in seconds (high byte)
- Byte 9: Unknown
- Byte 10: Unknown
- Byte 11: Index of highest jump on the unit (low byte)
- Byte 12: Index of highest jump on the unit (high byte)
- Byte 13: Checksum

Checksum

Every record received and command send has a checksum. To calculate the check sum:

Sum the type and data bytes. Take the sum modulo 256.

Device Info Packet

This packet type contains the Neptune's firmware version, serial number, and possibly other information.

Example device info packet: "14 00 00 26 00 45 33 38 39 30 38 20 20 20 05 01 01 00 00 00 00 DE"

- Byte 0: Packet length
- Byte 1: Packet type
- Byte 2: Always observed null
- Byte 3: High nibble: Firmware major version number. Low nibble: Firmware minor version numbers.
- Byte 4: Firmware revision number.
- Bytes 5-10: Neptune serial number (ASCII values)
- Byte 11-13: Always observed as spaces (hex 20)
- Bytes 14-20: Absent in firmware 2.3.1B. Observed as "05 01 01 00 00 00 00" in 2.4.2B and 2.6.0B
- Byte 14/21: Checksum

Tested against: Firmware 2.3.1B, 2.4.2B, 2.6.0B

Log Summary Packet

This packet type contains a summary of the log information stored in the Neptune.

Example log summary packet: "0C 01 92 00 0A 92 00 E2 1D 00 00 91 00 BF"

- Byte 0: Packet length
- Byte 1: Packet type
- Byte 2: Jumps on the unit (as displayed on the log summary screen)
- Byte 3: Jumps on the unit high byte.
- Byte 4: Number of jumps with detailed logging
- Byte 5: Number of the highest jump on the unit (low byte)
- Byte 6: Number of the highest jump on the unit (high byte)
- Byte 7: Total freefall time in seconds (low byte)
- Byte 8: Total freefall time in seconds (high byte)
- Byte 9: Unknown
- Byte 10: Unknown
- Byte 11: Index of highest jump on the unit (low byte)
- Byte 12: Index of highest jump on the unit (high byte)
- Byte 13: Checksum

Jump Summary Packet

This packet contains the summary information from a particular jump.

Example summary packet: “17 02 91 00 20 10 0B 09 05 03 42 36 34 31 14 40 10 D0 03 11 05 00 3B 00 44”

- Byte 0: Packet length
- Byte 1: Packet type
- Byte 2: Zero based index of the jump, low byte.
- Byte 3: Zero based index of the jump, high byte.
- Byte 4: Time of jump, minutes.
- Byte 5: Time of jump, hours (24 hour format)
- Byte 6: Day of jump.
- Byte 7: Month of jump.
- Byte 8: Year of jump.
- Byte 9: Unknown, possibly the alarm setting used (swoop, 4-way, etc.)
- Byte 10: Max speen in m/s (unconfirmed)
- Byte 11: 12k speed in m/s
- Byte 12: 9k speed in m/s
- Byte 13: 6k speed in m/s
- Byte 14: 3k speed in m/s
- Byte 15: Exit altitude in meters (low byte)
- Byte 16: Exit altitude in meters (high byte)
- Byte 17: Deployment altitude in meters (low byte)
- Byte 18: Deployment altitude in meters (high byte)
- Byte 19: Unknown
- Byte 20: Unknown
- Byte 21: Unknown
- Byte 22: Freefall time in seconds (low byte)
- Byte 23: Freefall time in seconds (high byte) - unconfirmed, anybody want to make a 256+ second skydive to test it?
- Byte 24: Checksum

Completion Packet

This packet seems to indicate that the Neptune has completed sending log data. Paralog always closes the connection immediately after receiving this packet.

Example completion packet: “01 03 03”

- Byte 0: Packet length
- Byte 1: Packet type
- Byte 2: Checksum

Detailed Log Event Packet

This packet is used to flag events during detailed logging.

Example summary packet: “04 04 05 7A 00 83”

- Byte 0: Packet length
- Byte 1: Packet type
- Byte 2: Event type
- Byte 3: Time, low byte
- Byte 4: Time, high byte
- Byte 5: Checksum

This packet indicates that event type 5 occured at time 0x007A.

Event Types

- 0×05: Exit
- 0×06: Deployment / Opening

These types may not be exact. For example, they oocasionaly disagree slightly with the apparently related times in the Detailed Log Header.

Detailed Log Header Packet

This packet proceeds detailed logging information for each jump.

Example summary packet: “0B 05 F4 01 10 01 EE 09 DE 00 77 01 58”

- Byte 0: Packet length
- Byte 1: Packet type
- Byte 2: Zero-based jump number, low byte
- Byte 3: Zero-based jump number, high byte
- Byte 4: Pressure altitude (QNE) in meters, low byte (unconfirmed but high confidence)
- Byte 5: Pressure altitude (QNE) in meters, high byte
- Byte 6: Exit altitude, meters, low byte
- Byte 7: Exit altitude, meters, high byte
- Byte 8: Exit time, meters, low byte
- Byte 9: Exit time, meters, high byte
- Byte 10: Opening time, low byte
- Byte 11: Opening time, meters, high byte
- Byte 12: Checksum

Detailed Log Data Packet

This packet contains datapoints from the detailed logging of a jump.

Example summary packet: “05 06 DA 0A 20 00 0A”

- Byte 0: Packet length
- Byte 1: Packet type

- Byte 2: Altitude, low byte
- Byte 3: Altitude, high byte
- Byte 4: Time, low byte
- Byte 5: Time, high byte
- Byte 6: Checksum

Notes:

- The time appears to be approximately 1/4 second.
- The altitude value needs to be treated as a signed integer. Due to calibration drift, changing atmospheric conditions, or simply landing below the takeoff altitude, it is possible for the altitude to register as negative

Go Ahead Packet

This packet seems to be sent my the computer to indicate that it is ready to receive data. It is also sent by the computer to acknowledge a Time Set.

Example go ahead packet: “01 80 80”

- Byte 0: Packet length
- Byte 1: Packet type
- Byte 2: Checksum

Unknown Command Packet

This packet is sent by the computer, apparently to request that the Neptune sends it’s stored logbook information.

Apparently not. It is actually sent before the Time Set packet, if the Time Set is sent at all. Who knows that it does.

Example log request packet: “05 A4 24 00 04 00 CC”

- Byte 0: Packet length
- Byte 1: Packet type
- Bytes 2-5: Unknown
- Byte 6: Checksum

This packet is always followed by the Non-Standard Reply packet.

The same packet was sent by firmware versions 2.3.1B, 2.4.2B, and 2.6.0B.

Time Set Packet

This packet is sent by the computer to set the time on the Neptune.

Example log request packet: “08 A6 01 14 03 D5 07 0C 1E C4”

- Byte 0: Packet length
- Byte 1: Packet type
- Byte 2: Hour (24 hour format)
- Byte 3: Minutes

- Byte 4: Seconds
- Byte 5: Year low byte
- Byte 6: Year high byte
- Byte 7: Month
- Byte 8: Day of month
- Byte 9: Checksum

Non-Standard Reply Packet

This four byte packet is sent by the Neptune after receiving a Log Request packet. It does not conform to the format seen in every other packet. The length is wrong, the type varies among neptune units, and the checksum doesn't make sense.

Example non-standard replies:

- "01 03 3A 00" Firmware 2.3.1B
- "51 02 3A 00" Firmware 2.4.2B
- "41 02 3B 00" Firmware 2.6.0B

Log Reguest

This has been renamed the Unknown Command Packet