Author: Computational Ecology and Environmental Science Group, Microsoft Research
Last Saved: 7/15/2014 1:47:00 PM

Mataki Documentation v1

Mataki is an open, reconfigurable, flexible, wirelessly-enabled, low-cost GPS tracking technology with wireless communication. Our aim is to keep Mataki low-cost and readily reprogrammable to allow researchers to explore novel tracking approaches by developing their own firmware and applications. We already provide firmware for tracking and base-station applications with a flexible set of parameters that should allow researchers to apply the devices in a number of tracking scenarios without any modifications or new development.

This manual sets out the necessary steps to connect and communicate with the devices, and to download logged data. It also described how to modify tracking parameters (e.g. update rate, sleep duration) and radio parameters (how often to attempt base station contact) to be set and updated, and outlines the process of base station/tracker communication.

Table of Contents

1. Physically connecting devices 4

    1.1 Connecting to Support/Communication boards. 5
    1.2 - How do I determine the correct COM port? 6

2. Setting up Putty.exe for communication 7

3. Communicating with the devices 11

4. Downloading Logged Data 13

5. Converting the downloaded data 15

6. Command Reference 16

    General Commands 16
    Debug commands 16
    GPS Tracker Firmware specific commands 16
    Additional Base Station Firmware Commands 17

7. Reprogramming the devices 18

  7.1. Firmware source code 18
  7.2. Building firmware 18
    7.2.1. Installing Python 2.7.3 18
    7.2.2. Installing Scons 2.2.0 18
  7.3. Reprogramming firmware 19
    7.3.1. Reprogramming hardware settings 20
    7.3.2. Reprogramming software settings 20
    7.3.3. Loading and programming 20

8. Default Commands - Explanation and Examples 23

    ID 23
    APP 23
    BATT 23
    FORMAT 24
    READ 24
    LOGSIZE 24
    CHANNEL 24
    TIME 24
    UPTIME 25
    TIMERS 25
    RR 25
    RESET 26
    ABORT 26
    HELP 26

9. Tracking Device Commands and Configuration 27

    SETGPS with examples 27
    SETRADIO with examples 27
    GPS 29
    SUPPRESS 29
    BASE 29

F_RESET 30
SCPREQ 30
SHOWGAPS 30
FILLGAPS 31
REQUEST 31

10. Base Station Commands and Configuration 32

10.1. An overview of the automatic download process 32
10.2. Commands 34
SHOWGAPS 34
FILLGAPS 35
REQUEST 35
F_RESET 35
FORCEDL 36

11. Remote Commands (TBC) .... **Error! Bookmark not defined.**

1. Physically connecting devices

There are two main ways to communicate and configure devices. The first, uses the larger Support board (fig 1a), the second the smaller Communication/charger board (fig 1b). The support board supports communication, charging and, with a programming cable, re-flashing devices (to update their firmware). The smaller board just allows for charging and communication.

Both the Support and Communication boards should be connected to a PC via a USB to Serial FTDI cable (Black cable supplied by Farnell part number 1329311[*]).
Necessary Software:

- Putty.exe[†] (terminal emulation software for communicating with devices over the serial port)
- USB-Serial drivers from FTDI[‡] (drivers for the USB-Serial cables, although these may now be included in Windows 7/8)

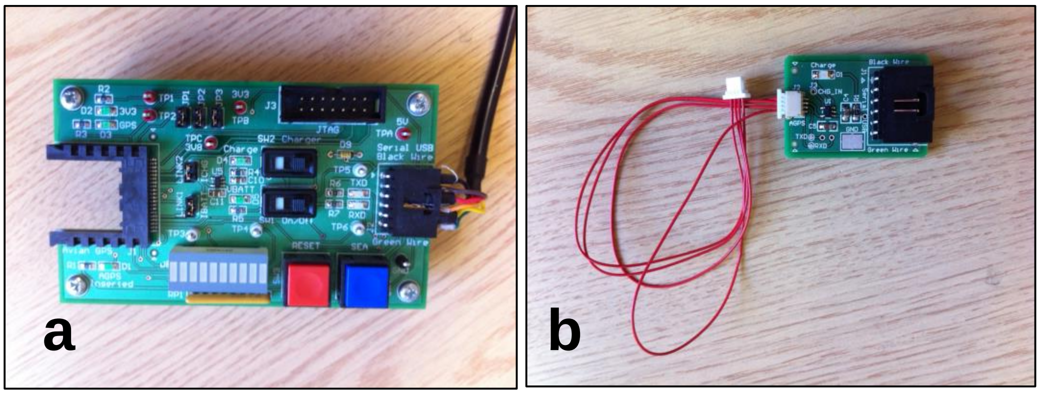


Figure 1 – Support (a) and Communications/charger board (b)

[*] http://uk.farnell.com/ftdi/ttl-232r-3v3/cable-usb-to-ttl-level-seri-converter/dp/1329311
[†] http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe
[‡] http://www.ftdichip.com/Drivers/CDM/CDM%202.08.24%20WHQL%20Certified.zip

1.1 Connecting to Support/Communication boards.

Devices can be connected to the Support board by sliding the PCB edge connector into the socket on the board (Figure 2, left). Take care to ensure that the battery connector wire on the rear of the device does not get pinched during insertion.

Connecting to the Communication/charger board uses the small 4-pin connector (Figure 2, right), the flatter contactless side of the connector orients to the top of the device (the side with the GPS antenna).

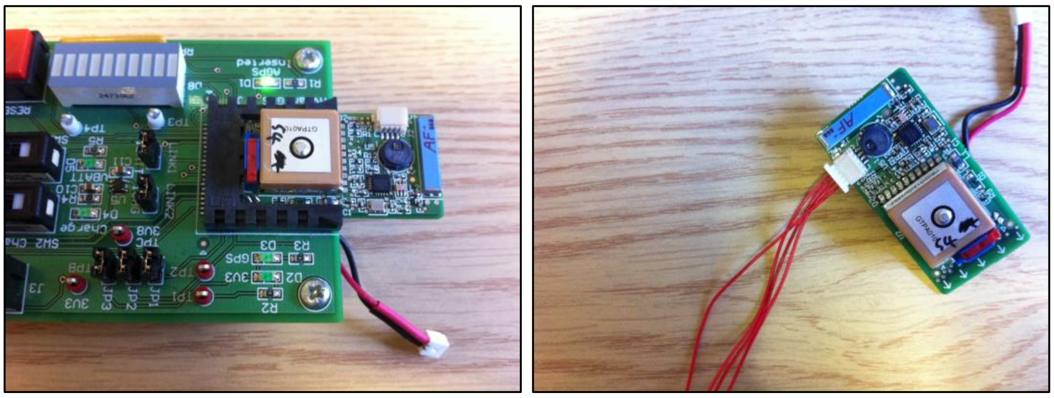


Figure 2 – Connecting to the Support board (left) and the Communication/charger board (right). Both boards can charge a battery connected to the device. The Support board also enables firmware reprogramming and other debugging measurements

1.2 - How do I determine the correct COM port?

In Windows, open Device Manager (see grey box below). In Device Manager, expand (click the + icon) the section titled Ports (COM & LPT). Here, for each connected USB-Serial cable/device, you should see a line named ‘USB Serial Port (COMX), where X is the corresponding COM port number for that device (e.g. COM8 is port 8 – see **Error! Reference source not found.**).

Windows XP:

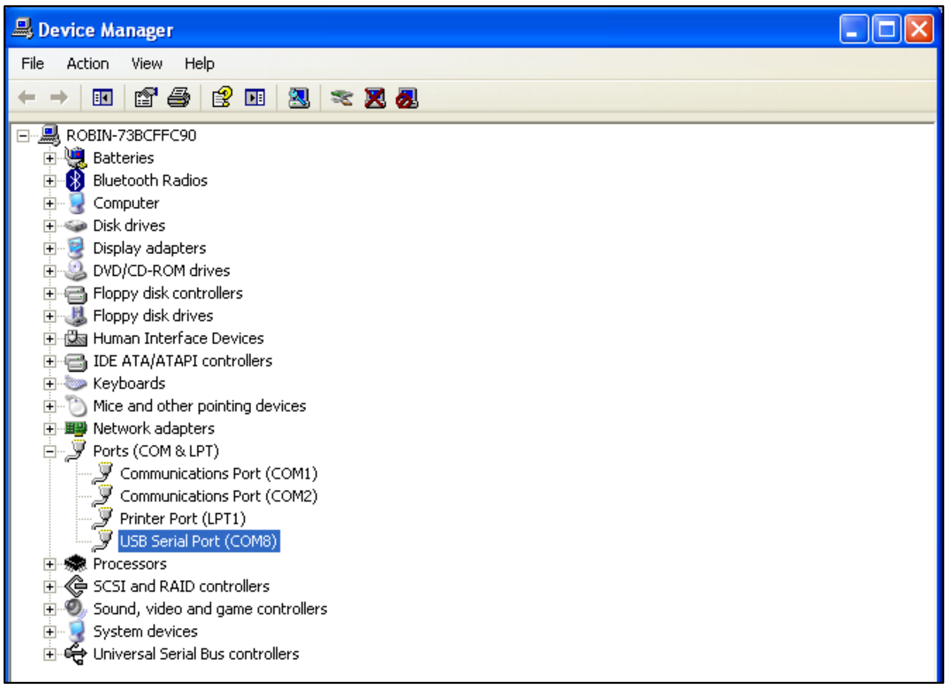
a) From the start menu, select ‘Run...’ and type **devmgmt.msc** in the Open box and click OK.
b) Alternatively, right click on ‘My Computer’ select ‘Properties’, then the ‘Hardware’ tab, and then click ‘Device Manager’

Windows Vista:

a) Click on the start button and type **devmgmt.msc** in the Search box and press the enter key.
b) Alternatively, go to the Start Menu, then the Control Panel, click on the ‘Sytem and Maintainance’ link, in the Systems and Maintainance’ window click on the device manager link near the bottom.
    a. If you’re using the *Classic* view of the control panel, you may not see a this link, in this case you should simply be able to double-click on the Device Manager icon in the control panel.

Windows 7:

a) Click on the Start button, and enter **devmgmt.msc** in the Search box and press the enter key.
b) Alternatively, go to the Start Menu, then the Control Panel, click on the ‘System and Security’ link, in the Systems and Maintenance’ window click on the device manager link near the bottom.
    a. If you’re using the *Large icons or Small icons* view of the control panel, you may not see this link, in this case you should simply be able to double-click on the Device Manager icon in the control panel.



```
ROBIN-73BCFFC90
  Batteries
  Bluetooth Radios
  Computer
  Disk drives
  Display adapters
  DVD/CD-ROM drives
  Floppy disk controllers
  Floppy disk drives
  Human Interface Devices
  IDE ATA/ATAPI controllers
  Keyboards
  Mice and other pointing devices
  Network adapters
  Ports (COM & LPT)
    Communications Port (COM1)
    Communications Port (COM2)
    Printer Port (LPT1)
    USB Serial Port (COM8)
  Processors
  SCSI and RAID controllers
  Sound, video and game controllers
  System devices
  Universal Serial Bus controllers
```

Figure 3 – Device Manager, showing a connected USB-Serial cable/device on COM8

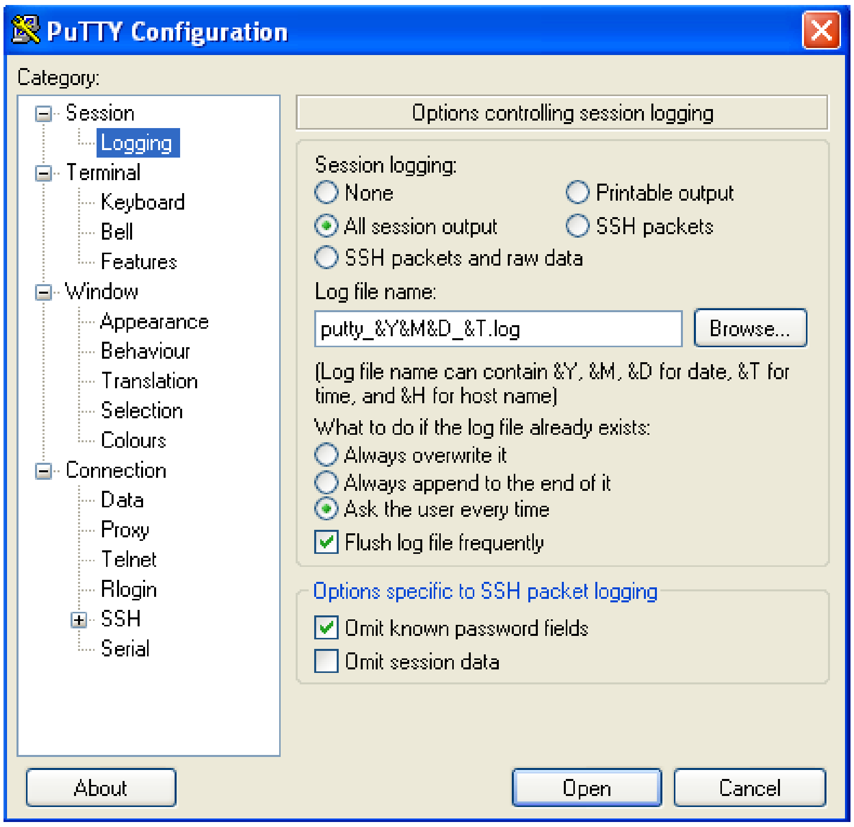
2. Setting up Putty.exe for communication

To communicate with devices using the boards, terminal emulation software with logging support is needed. Putty[§] is a freely available terminal emulation software application for Windows that supports serial connections and logging. Putty can be downloaded at (http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe) and runs simply by double clicking on the downloaded file. In order to enable logging, and to make the communication process more straightforward, a number of settings should be enabled within Putty. These settings can be selected once, and can then be saved for future use.

1) Double click on putty.exe
2) Under ‘Connection Type’ select ‘Serial’
3) In the ‘Serial line’ box above, enter the COM port that your Support/Communication board is connected to (see ‘how do I determine the COM port’ below). For example if your device is connected to port 10, enter COM10
4) Set the ‘Speed’ (baud rate) to 115200
5) Click on ‘Logging’ in the left menu, under ‘Session’ (see Figure 4)
    a. Select ‘All session output’ under ‘Session logging’
    b. Enter a filename where the session will be logged. A good option here is to use the support control characters for time and date, e.g. entering putty_&Y&M&D_&T.log would create a file with the date and time that logging started whenever a new session began (resulting, for example, in a file called putty_20120821_134245.log). By default, the destination of the created log files is the same directory where the file putty.exe is located. Other destination directory can be selected by pressing the Browse button in front of the Log file name input box.
6) Click on ‘Terminal’ in the left menu (see Figure 5)
    a. Select ‘Force on’ in both cases under ‘Line discipline options’, these will enable you to see the commands you type in the terminal, making it easier to identify any potential errors.
7) Click on ‘Session’ in the left menu
    a. Enter a name for your new settings in the box labeled ‘Saved Sessions’
        i. Chose a meaningful name, such as COM10_115200 to enable you to remember which port this saves session references.
    b. Click ‘Save’ to save these settings

Once these settings have been saved, you can double click on the name of the saved sessions in the larger box on the ‘Session’ page (Figure 6). **If either Support or Communication board is connected to the appropriate port**, you should see a black screen appear where communication to/from the device can occur. If neither Support nor Communication board is connected, an error message will appear in small window titles ‘Putty Error’ stating “Unable to open connection to COM? Unable to open serial port.

[§] http://www.chiark.greenend.org.uk/~sgtatham/putty/



Options controlling session logging

Session logging:
None
Printable output
[✓ All session output]
SSH packets
SSH packets and raw data

Log file name: putty_&Y&M&D_&T.log

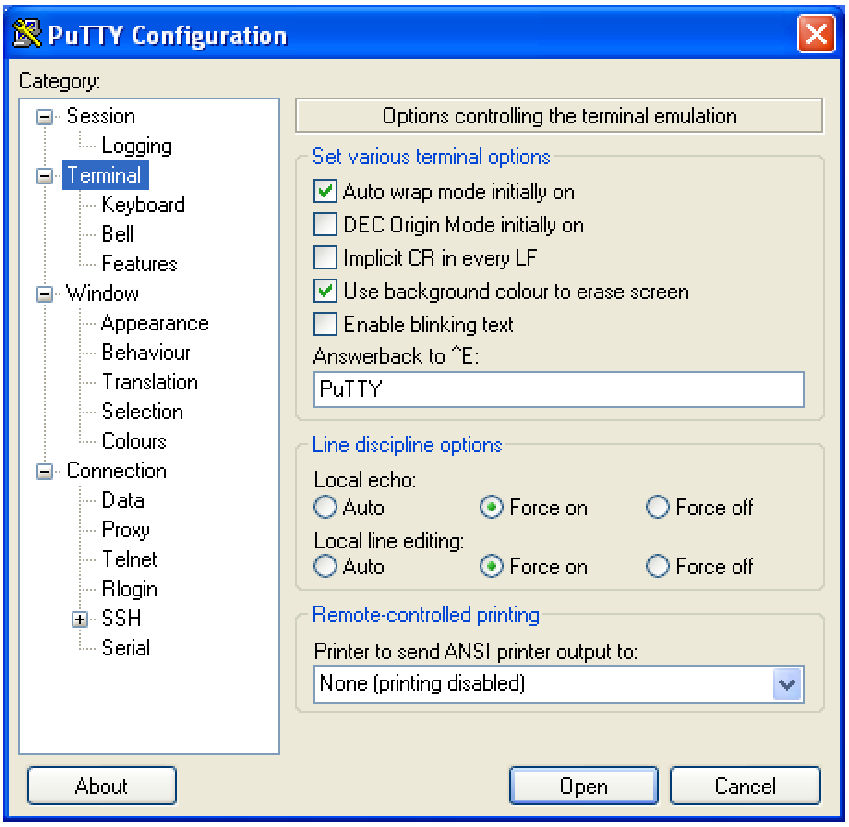
(Log file name can contain &Y, &M, &D for date, &T for time, and &H for host name)

What to do if the log file already exists:
Always overwrite it
Always append to the end of it
[✓ Ask the user every time]

[✓] Flush log file frequently

Options specific to SSH packet logging
[✓] Omit known password fields
[ ] Omit session data

Figure 4 - Logging panel in putty.exe. The selected setting ('All session output') causes Putty to create a log file whenever a session is started (using the filename pattern shown). Files are created in the same directory as Putty


PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Options controlling the terminal emulation

Set various terminal options

- [✓] Auto wrap mode initially on
- [ ] DEC Origin Mode initially on
- [ ] Implicit CR in every LF
- [✓] Use background colour to erase screen
- [ ] Enable blinking text

Answerback to ^E: PuTTY

Line discipline options

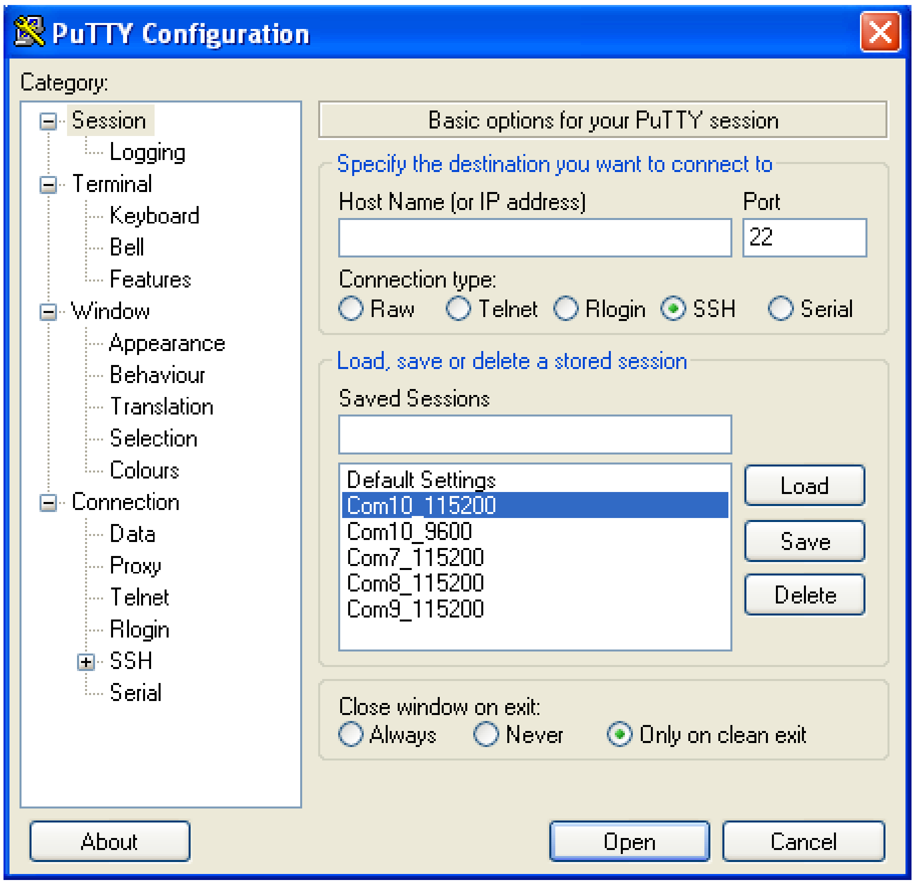
Local echo: Auto [✓ Force on] Force off

Local line editing: Auto [✓ Force on] Force off

Remote-controlled printing

Printer to send ANSI printer output to: None (printing disabled)

Figure 5 – Terminal panel in putty.exe, here the selected settings (Local echo: force on, Local line editing: force on) enable the user to see and edit the commands they type



Host Name (or IP address):
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Saved Sessions:
Default Settings
Com10_115200
Com10_9600
Com7_115200
Com8_115200
Com9_115200
Close window on exit: Always Never [✓ Only on clean exit]

Figure 6 - Saved sessions panel in putty.exe. Each saved session appears in the larger lower area and can be double clicked on to start a session with those settings. Here sessions are given informative names including the used COM port and baud rate

3. Communicating with the devices

Once you have a connection to the device established with putty.exe, you should see a blank putty window (Figure 7). If needed, you can test the logging capability by typing some text and pressing enter, the log file should appear in the same directory as putty.exe and should contain the entered text in addition to a header line with the start time and data of the logging (Figure 8). Files should not be moved or renamed while logging is occurring, but the operating system should prevent this.

Figure 7 – Blank putty window, ready to communicate with devices

Once communication is established and logging has begun, you can connect a device to the Support or Communication board. The only miniature switch on the device (SW1) is used to power up the device through either the power connection supplied by the Support board (when SW1 is on ON mode), or otherwise, the power supplied by the device's battery connaction (SW1 on OFF mode).
When powered up, the devices will display a number of initialization and status messages on the screen (**Error! Reference source not found.**). Once the initialization is complete, you can enter commands. As described in the command sections below, entering the 'abort' command here allows you to prevent the device from entering the tracking regime (or the automatic base station mode), thus allowing you to interact with the device until you decide to reset it.

In general, commands modify settings on the local device and are recorded in the device log. Some more advanced commands (such as **request**, **forcedl** effect remote devices, when communication is available). Once a command is entered, the device should respond to acknowledge that the command has been executed. For example, entering:

```
app
```

should return the result of the command:

```
[69:app] 'Tracker' 1.2 compiled 00:11:05.0000 28-08-2012
```

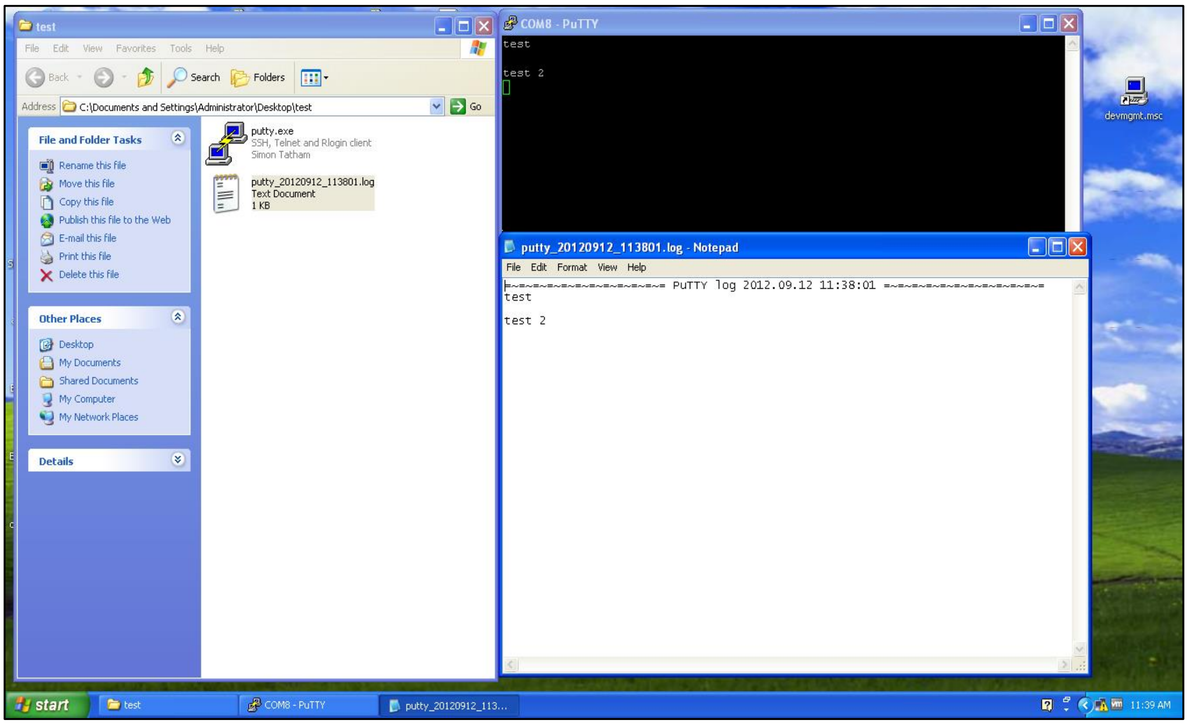
Here, the text in square bracket [69:app], contains the ID of the responding device, and the command the responds relates to. Alternatively, entering

```
abort
```

displays the message OK to indicate that the command was successful.

```
[54:abort] OK
```

See the sections below for a command reference, configuration of trackers, base stations communication and downloading logs.



```
test

test 2
```

```
=~=~=~=~=~=~=~=~=~=~=~= PuTTY log 2012.09.12 11:38:01 =~=~=~=~=~=~=~=~=~=~=~=
test

test 2
```

Figure 8 – Putty window with test text entered and logged to a file in the same directory as putty.exe. The file contains a header line with the date and time that logging started. The file will be updated as communication continues, but to see the changes in notepad (for example), you would need to reopen the file. Files should not be moved or renamed while logging is occurring.

```
[54:L] App: 'Tracker' V: 1.2 for MSP430 with AVIAN5 [00:00:00.0045 01-01-2010]
[54:L] Compiled: 00:11:05.0000 28-08-2012 UTC [00:00:00.0159 01-01-2010]
[54:L] NodeID: 54 [00:00:00.0229 01-01-2010]
 [54:L] SMCLK: 3964928 Hz (from 1114112 Hz in 854 steps) [Aimed: 4000000 (122),
Min: 3960000, Max: 4040000 | dco: 5 mod: 22 rsel: 10] [00:00:00.0541 01-01-2010]
[54:D]  CMA3000: I2C is disabled [00:00:00.0662 01-01-2010]
[54:D]  CC1101: PartNumber 0, Version 4 [00:00:00.0759 01-01-2010]
[54:D]  Computing RF Configuration [00:00:00.0847 01-01-2010]
[54:D]  CC.Carrier: 867999936 Hz [00:00:01.0331 01-01-2010]
[54:D]  CC.BaudRate: 249939 Baud [00:00:01.0431 01-01-2010]
[54:D]  CC.Bandwith: 325000 Hz [00:00:01.0531 01-01-2010]
[54:D]  CC.ChannelSpace: 325317 Hz [00:00:01.0633 01-01-2010]
[54:D]  CC.Channel: 0 [00:00:01.0713 01-01-2010]
[54:D]  CCxxx0_FREQ2 0x21 [00:00:01.0797 01-01-2010]
[54:D]  CCxxx0_FREQ1 0x62 [00:00:01.0881 01-01-2010]
[54:D]  CCxxx0_FREQ0 0x76 [00:00:01.0964 01-01-2010]
[54:D]  CCxxx0_MDMCFG4 0x5D [00:00:01.1050 01-01-2010]
[54:D]  CCxxx0_MDMCFG3 0x3B [00:00:01.1135 01-01-2010]
[54:D]  CCxxx0_MDMCFG1 0x23 [00:00:01.1221 01-01-2010]
[54:D]  CCxxx0_MDMCFG0 0x9A [00:00:01.1307 01-01-2010]
[54:L] CC1101 is in RX [00:00:01.1400 01-01-2010]
[54:D]  Network: Registered handler for protocol 1 [00:00:01.1502 01-01-2010]
[54:D]  Network: Registered handler for protocol 2 [00:00:01.1603 01-01-2010]
[54:D]  Network: Registered handler for protocol 3 [00:00:01.1704 01-01-2010]
[54:L] System init completed. [00:00:01.1782 01-01-2010]
[54:L] Init sensors... [00:00:01.1854 01-01-2010]
[54:D]  S25FL128P Init... [00:00:01.1937 01-01-2010]
[54:D]  Capacity: 16777216 bytes [00:00:01.2044 01-01-2010]
[54:L] S25FL128P Init: OK [00:00:01.2119 01-01-2010]
[54:D]  LogEngine: total slots 466032 (log entry size 36) [00:00:01.2240 01-01-
2010]
[54:D]  LogEngine: slots used 473 (log entry size 36) [00:00:01.2916 01-01-2010]
[54:L] LogEngine Init: OK [00:00:01.2991 01-01-2010]
[54:D]  Network: Registered handler for protocol 208 [00:00:01.3104 01-01-2010]
[54:D]  Network: Registered handler for protocol 10 [00:00:01.3206 01-01-2010]
[54:D]  Network: Registered handler for protocol 11 [00:00:01.3309 01-01-2010]
[54:D]  Network: Registered handler for protocol 12 [00:00:01.3412 01-01-2010]
[54:D]  Network: Registered handler for protocol 13 [00:00:01.3515 01-01-2010]
[54:L] LogEngine.Sync Init: OK [00:00:01.3593 01-01-2010]
[54:D]  SCP Enable... [00:00:01.4152 01-01-2010]
[54:D]  SCP ASIC revision number: 3 [00:00:01.4405 01-01-2010]
[54:D]  SCP operation register: 0x00 [00:00:01.4497 01-01-2010]
[54:D]  SCP CFG: 0x05 (0x05=17bits, 0x0D=15bits) [00:00:01.4597 01-01-2010]
[54:L] SCP Enable: OK [00:00:01.4669 01-01-2010]
[54:L] Battery: 3.77V [00:00:02.4762 01-01-2010]
[54:D]  GPS Init... (up to 20 seconds) [00:00:02.4854 01-01-2010]
[54:D]  GPS: Got ACK from GPS, length: 10 [00:00:08.0126 01-01-2010]
[54:D]  GPS: Detected at 4800 baud. Reconfiguring to 38400... [00:00:08.0233 01-01-
2010]
[54:D]  GPS: Attemping comm at 38400...attempt 0 [00:00:08.1551 01-01-2010]
[54:D]  GPS: Got ACK from GPS, length: 10 [00:00:14.4720 01-01-2010]
[54:D]  GPS: Detected at 38400 baud [00:00:14.4813 01-01-2010]
[54:D]  GPS: NMEA Configuration..attempt 0 [00:00:14.4912 01-01-2010]
[54:D]  GPS: Got ACK from GPS, length: 12 [00:00:15.0939 01-01-2010]
[54:D]  GPS: NMEA configured: GGL: 0 RMC: 5 VTG 0 GGA: 1 GSA: 0 GSV: 0 GRS: 0 GST:
0 [00:00:15.1063 01-01-2010]
[54:D]  GPS: Got ACK from GPS, length: 23 [00:00:15.3906 01-01-2010]
[54:D]  GPS: Got ACK from GPS, length: 22 [00:00:17.5830 01-01-2010]
[54:D]  GPS: Position Update Rate: 10 Hz [00:00:17.5928 01-01-2010]
[54:L] GPS Init: OK [00:00:17.6008 01-01-2010]
[54:L] Init sensors: OK [00:00:17.6082 01-01-2010]
[54:D]  GPS Powered Waiting 5 seconds [00:00:17.6877 01-01-2010]
[54:D]  Powering Serial line [00:00:22.6962 01-01-2010]
[54:D]  Registering GPS NMEA Handler... [00:00:22.7063 01-01-2010]
[54:D]  ...Registered GPS NMEA Handler. [00:00:22.7157 01-01-2010]
[54:D]  Network: Registered handler for protocol 100 [00:00:22.7262 01-01-2010]
[54:D]  Network: Registered handler for protocol 101 [00:00:22.7367 01-01-2010]
[54:L] Use the 'abort' command to abort the experiment! [00:00:22.7474 01-01-2010]
[54:L] App init completed. [00:00:22.7550 01-01-2010]
[54:L] UART1 RX BUFF LEN: 38 [00:00:22.7748 01-01-2010]
[54:L] UART1 RX BUFF LEN: 48 [00:00:23.1303 01-01-2010]
```

Figure 9 - Initialization output from device with Tracker firmware

4. Downloading Logged Data

When a device is connected and communication is established (and logged) the **read** command can be used to download all stored logs on the device. For example, on a device with 47293 logs:

```
read
```

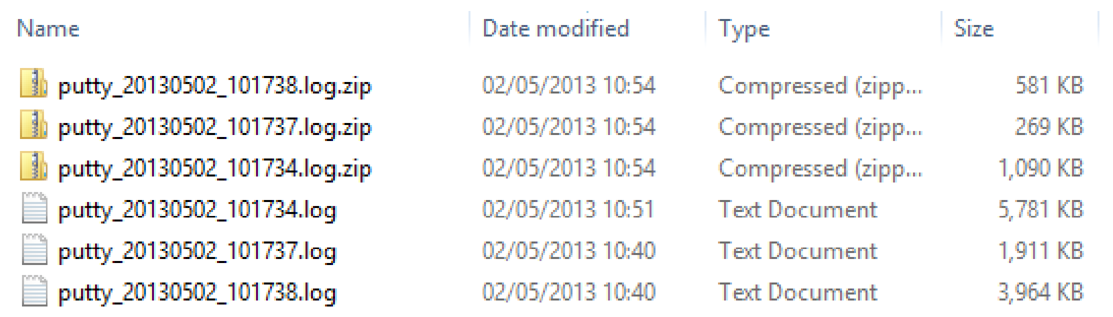
would display all the logged information on the device:

```
Selected range [1 - 47293]
[23:L] 1       0       0       23      6       00:01:53.1072 01-01-2010      23      \
[23:L] 2       0       0       23      6       00:01:54.7486 01-01-2010      23
       logsize
[23:L] 3       0       0       23      6       00:01:57.1984 01-01-2010      23
       setgps
[23:L] 4       0       0       23      6       00:01:58.6968 01-01-2010      23
       setradio
[23:L] 5       0       0       23      6       00:02:01.3886 01-01-2010      23      gps
6 10
[23:L] 6       0       0       23      5       00:00:01.2466 01-01-2010      11
       pw:0    rr:0    fp:0    sp:0
[23:L] 7       0       0       23      8       00:00:30.3965 01-01-2010      GPS_Off
[23:L] 8       0       0       23      2       00:00:30.4124 01-01-2010
[23:L] 9       0       0       23      8       00:00:35.5359 01-01-2010      GPS_On
[23:L] 10      0       0       23      8       00:01:40.6048 01-01-2010      GPS_On
[23:L] 11      0       0       23      8       00:01:40.6655 01-01-2010      NMEAchecksu
[23:L] 12      0       0       23      8       00:01:42.7187 01-01-2010      NMEAchecksu
[23:L] 13      0       0       23      3       00:02:11.0979 01-01-2010      GPGGA
       201426.500      5110.0154       -440.0082       1       4       3.0     141.7   OK
[23:L] 14      0       0       23      3       00:02:11.1799 01-01-2010      GPGGA
       201426.600      5110.0154       -440.0082       1       4       3.0     140.9   OK
[23:L] 15      0       0       23      3       00:02:11.2619 01-01-2010      GPGGA
       201426.700      5110.0153       -440.0079       1       4       3.0     140.2   OK
[23:L] 16      0       0       23      3       00:02:11.3437 01-01-2010      GPGGA
       201426.800      5110.0153       -440.0074       1       4       3.0     139.2   OK
[23:L] 17      0       0       23      3       00:02:11.4263 01-01-2010      GPGGA
       201426.900      5110.0152       -440.0067       1       4       3.0     138.1   OK
[23:L] 18      0       0       23      5       20:14:26.4424 16-08-2012      3
       Updated RTC from 00:02:11.4423 01-01-2010
[23:L] 19      0       0       23      3       20:14:26.5076 16-08-2012      GPGGA
       201427.000      5110.0151       -440.0069       1       4       3.0     138.0   OK
[23:L] 20      0       0       23      3       20:14:26.5895 16-08-2012      GPGGA
       201427.100      5110.0151       -440.0075       1       4       3.0     138.4   OK
[23:L] 21      0       0       23      3       20:14:26.6715 16-08-2012      GPGGA
       201427.200      5110.0151       -440.0079       1       4       3.0     138.5   OK
[23:L] 22      0       0       23      3       20:14:26.7534 16-08-2012      GPGGA
       201427.300      5110.0151       -440.0084       1       4       3.0     138.7   OK
[23:L] 23      0       0       23      3       20:14:27.0166 16-08-2012      GPGGA
       201427.400      5110.0151       -440.0090       1       4       3.0     139.2   OK
...
...
...
[23:L] 47284   0       0       23      8       00:01:41.6900 01-01-2010      GPS_Off
[23:L] 47285   0       0       23      8       00:01:41.7069 01-01-2010      GPS_Off
[23:L] 47286   0       0       23      5       00:01:44.1061 01-01-2010      11
       pw:0    rr:0    fp:0    sp:0
[23:L] 47287   0       0       23      8       00:02:13.2820 01-01-2010      GPS_Off
[23:L] 47288   0       0       23      2       00:02:13.2974 01-01-2010
[23:L] 47289   0       0       23      8       00:02:18.4219 01-01-2010      GPS_On
[23:L] 47290   0       0       23      6       00:02:20.5660 01-01-2010      23
       abort
[23:L] 47291   0       0       23      8       00:02:20.5677 01-01-2010      GPS_Off
[23:L] 47292   0       0       23      4       00:02:20.5685 01-01-2010      2327
       3.75V
[23:L] 47293   0       0       23      6       00:02:26.7828 01-01-2010      23
       read
[23:read] OK
```

As the session is being logged, this output will be logged to the corresponding log file. At this point it is important to check the logfile to ensure no corruption has occurred before deleting data from the device. Once you are happy that the data has been downloaded successfully, and if you wish to delete the data, the device can be formatted. *****Note that data cannot be recovered once the device is formatted.*****

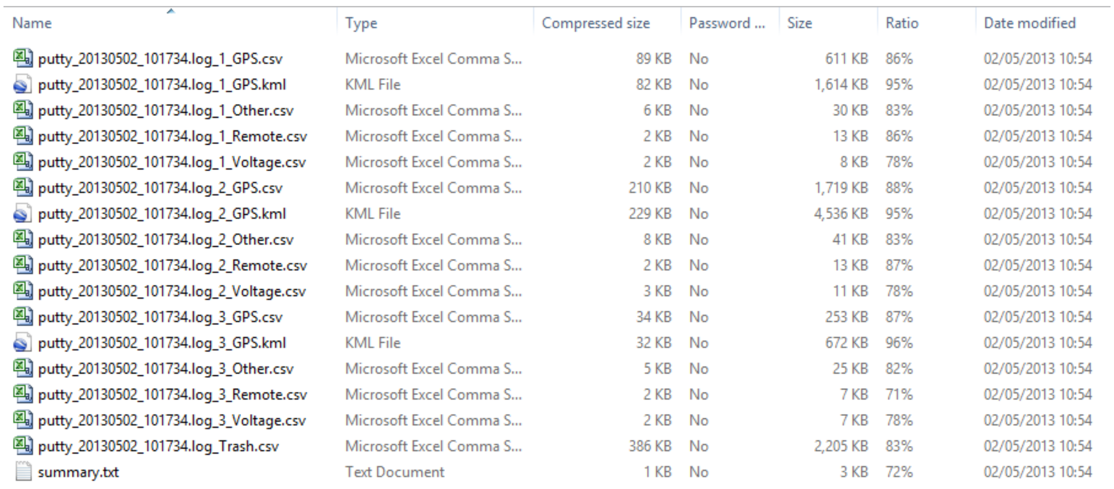
5. Converting the downloaded data

The created logfile may be converted to the separate files for easier post-processing of different sensor data. One tool that can help with this is available on Mataki website using the link http://mataki.org/logfile-convertor/ . The installable file MatakiLogConvertor_Setup.exe, or ConvertMatakiLog.dmg depending on the Operating System, once installed, creates the folder MatakiConvertLog on the Program Files (this is based on the Windows Operating System). This folder contains the executable file ConvertMatakiLog.exe that automatically parses and converts the logfile to a set of spreadsheets and other file formats. All needs to be done is selecting the logfile. The resulting converted files are compressed in zip format as one zip file stored in the same location where the logfile is located. Figure 10 and 11 is an example snapshots of the files created.



Name	Date modified	Type	Size
putty_20130502_101738.log.zip	02/05/2013 10:54	Compressed (zipp...	581 KB
putty_20130502_101737.log.zip	02/05/2013 10:54	Compressed (zipp...	269 KB
putty_20130502_101734.log.zip	02/05/2013 10:54	Compressed (zipp...	1,090 KB
putty_20130502_101734.log	02/05/2013 10:51	Text Document	5,781 KB
putty_20130502_101737.log	02/05/2013 10:40	Text Document	1,911 KB
putty_20130502_101738.log	02/05/2013 10:40	Text Document	3,964 KB

Figure 10 - Conversion tool generates one zip folder per each logfile



Name	Type	Compressed size	Password ...	Size	Ratio	Date modified
putty_20130502_101734.log_1_GPS.csv	Microsoft Excel Comma S...	89 KB	No	611 KB	86%	02/05/2013 10:54
putty_20130502_101734.log_1_GPS.kml	KML File	82 KB	No	1,614 KB	95%	02/05/2013 10:54
putty_20130502_101734.log_1_Other.csv	Microsoft Excel Comma S...	6 KB	No	30 KB	83%	02/05/2013 10:54
putty_20130502_101734.log_1_Remote.csv	Microsoft Excel Comma S...	2 KB	No	13 KB	86%	02/05/2013 10:54
putty_20130502_101734.log_1_Voltage.csv	Microsoft Excel Comma S...	2 KB	No	8 KB	78%	02/05/2013 10:54
putty_20130502_101734.log_2_GPS.csv	Microsoft Excel Comma S...	210 KB	No	1,719 KB	88%	02/05/2013 10:54
putty_20130502_101734.log_2_GPS.kml	KML File	229 KB	No	4,536 KB	95%	02/05/2013 10:54
putty_20130502_101734.log_2_Other.csv	Microsoft Excel Comma S...	8 KB	No	41 KB	83%	02/05/2013 10:54
putty_20130502_101734.log_2_Remote.csv	Microsoft Excel Comma S...	2 KB	No	13 KB	87%	02/05/2013 10:54
putty_20130502_101734.log_2_Voltage.csv	Microsoft Excel Comma S...	3 KB	No	11 KB	78%	02/05/2013 10:54
putty_20130502_101734.log_3_GPS.csv	Microsoft Excel Comma S...	34 KB	No	253 KB	87%	02/05/2013 10:54
putty_20130502_101734.log_3_GPS.kml	KML File	32 KB	No	672 KB	96%	02/05/2013 10:54
putty_20130502_101734.log_3_Other.csv	Microsoft Excel Comma S...	5 KB	No	25 KB	82%	02/05/2013 10:54
putty_20130502_101734.log_3_Remote.csv	Microsoft Excel Comma S...	2 KB	No	7 KB	71%	02/05/2013 10:54
putty_20130502_101734.log_3_Voltage.csv	Microsoft Excel Comma S...	2 KB	No	7 KB	78%	02/05/2013 10:54
putty_20130502_101734.log_Trash.csv	Microsoft Excel Comma S...	386 KB	No	2,205 KB	83%	02/05/2013 10:54
summary.txt	Text Document	1 KB	No	3 KB	72%	02/05/2013 10:54

Figure 11 - Each zip folder contains spreadsheets and kml files, as well as a summary file

6. Command Reference

Each of our firmware applications/device configurations uses a ‘command’ system to allow easy configuration and control of the programmed devices. A short description of these commands are given below. More complete description and examples are provided in Chapters 7, 8, 9, and 10. It’s important to note that the available commands vary depending on the specific firmware (e.g. some commands are only available in the ‘basestation’ firmware).

General Commands

- **help**: List the commands available on the current device/firmware.
- **id**: Configuration: node id
- **app**: Configuration: application’s metadata
- **batt**: Measures the battery voltage
- **flashtst**: Performs self test of the device.
- **format**: Formats the flash memory
- **read**: Displays log entries in specified range
- **logsize**: Provides the current number of stored log entries
- **channel**: Reconfigures the current radio channel
- **time**: Manage local time
- **uptime**: Provides the uptime of the device since the last reset
- **timers**: Debug command: Provides information about registered software timers.
- **rr**: Provides information about the reset reason of the device.
- **reset**: reset Forces the device to reset.
- **abort**: Prevent the device entering into the autonomous tracker/basestation mode.

Debug commands

- **timers**: Debug: Provides information about registered software timers.
- **rr**: Debug: Provides the recent reset reason
- **reset**: Resets the device
- **channel**: Radio: communication channel
- **ccrx**: Debug: reserved for firmware developers
- **ccfrx**: Debug: reserved for firmware developers
- **ccget**: Debug: reserved for firmware developers
- **ccoff**: Debug: reserved for firmware developers
- **ccidle**: Debug: reserved for firmware developers
- **cc**: Debug: reserved for firmware developers
- **batt**: Battery voltage (correct results only when GPS is enabled)
- **light**: Debug: reserved for firmware developers
- **flashtst**: Debug: reserved for firmware developers

GPS Tracker Firmware specific commands

- **gps**: GPS debug interface
- **showgaps**: Shows gaps in downloaded logs
- **fillgaps**: Finds gaps in downloaded logs and requests missing entries from remote devices
- **request**: Requests a sequence of remote log entries

- **format**: Formats the flash memory
- **read**: Displays log entries from a specified range
- **logsize**: Number of local log entries
- **scpreq**: Triggers temperature and pressure measurment
- **abort**: Aborts the experiment
- **f_reset**: Formats and resets the device
- **setradio**: Configures the radio experiment
- **base**: Brings the device to the ‘base station’ mode in the selected channel
- **setgps**: Configures the GPS experiment
- **setacc**: Configures the Accelerometer experiment
- **suppress**: Configures down time for the GPS (overrides setgps)

Additional Base Station Firmware Commands

- **showgaps**: Shows gaps in downloaded logs
- **fillgaps**: Finds gaps in downloaded logs and requests missing entries from remote devices
- **request**: Requests a sequence of remote log entries
- **scpreq**: Triggers temperature and pressure measurment
- **abort**: Aborts the automatic base station mode
- **f_reset**: Formats and resets the device
- **forcedl**: Forces a download from a selected tracker. Restarts the base station state machine.

7. Reprogramming the devices

The open source nature of the project allows users to modify the firmware and reprogram the processor on the device as according to their particular requirements. This chapter introduces the tools and steps required for compiling the source code and reprogramming the device. However, this chapter does not intend to provide description of the firmware code. Relevant API/Firmware documentation is provided separately and can be found on Mataki website (http://mataki.org/firmware/).

7.1. Firmware source code

A series of files that are written in C and resided in the Firmware directory comprise the firmware for Mataki device. Relevant documentation is provided in the Documentation subdirectory to introduce the Application configuration, commands, and data structure. It is also explained how to create your own application based on the existing files in the Apps\Empty subdirectory.

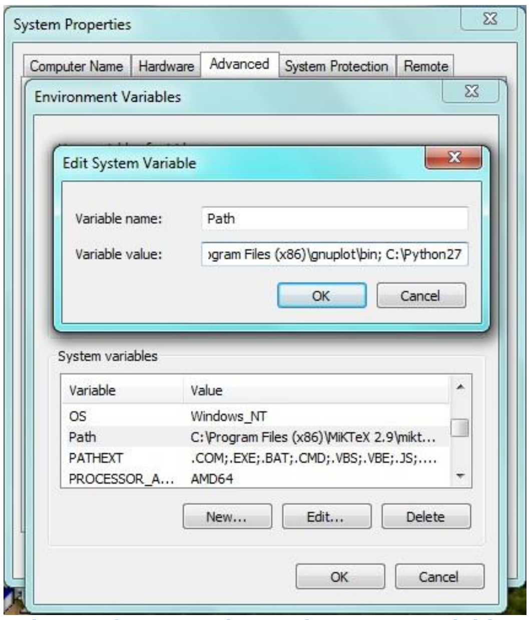
7.2. Building firmware

Although the classic C compiling tool can be used to compile the source code and generate hex file, other C compiling tools may shorten this task. Amongst them is Scons software construction tool and its requirement tool Python. The installation of these tools will be explained in the following subsections.

7.2.1. Installing Python 2.7.3

The link http://www.python.org/getit/ points to the source of Python software where it can be downloaded from. After Python is installed, make sure it works. You can do so by typing 'python --version' at the command line, that should show 2.7.3 or higher.

In addition, add path to the installed Python to the PATH Environment Variables (see the snapshot in Figure 12).



Variable name: Path
Variable value: ›gram Files (x86)\gnuplot\bin; C:\Python27

Variable	Value
OS	Windows_NT
Path	C:\Program Files (x86)\MiKTeX 2.9\mikt...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	AMD64

Figure 12 - Add path to Python software to the Environment Variable Path. Similar task may be repeated for other purpose, e.g. add path to Scons software in the following section

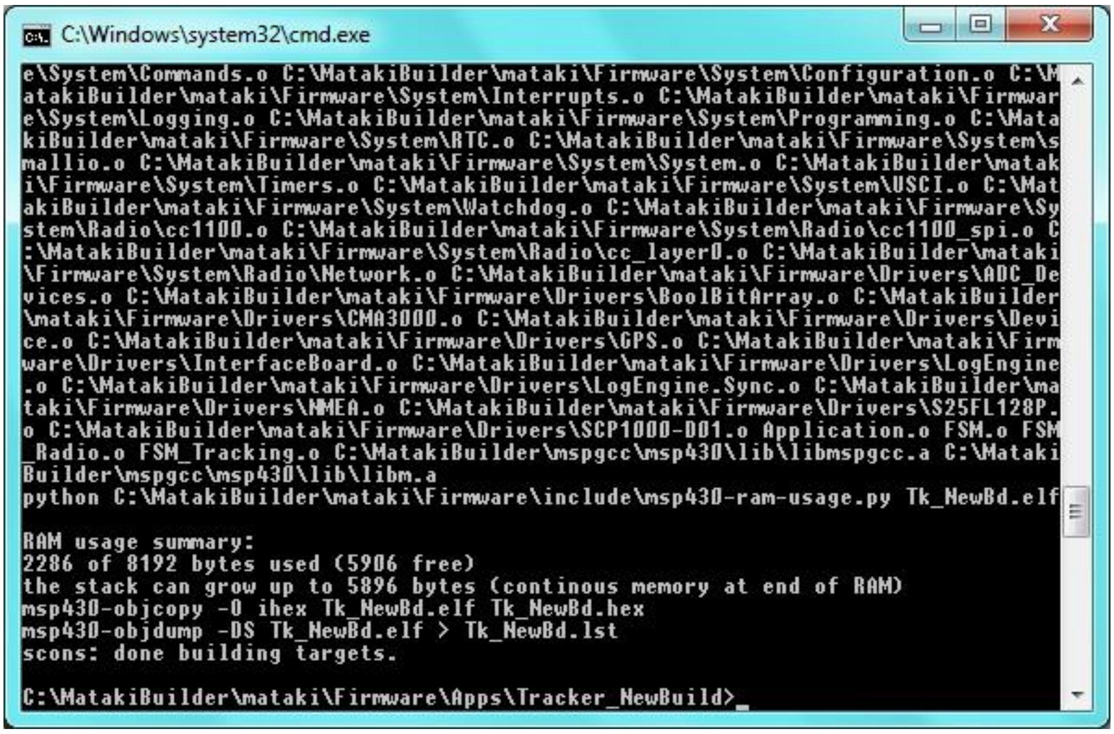
7.2.2. Installing Scons 2.2.0

The link http://www.scons.org/download.php points to the source of Scons software when it can be downloaded from. Install SCons, for example on Windows systems type ‘python setup.py install' command on the command prompt when it is in the scons-x.x.x directory. After Scons is installed, make sure it works. You can do so by typing scons at the command line, that should result in Scons running, but not finding any files. To do this, you'll need to make sure that the python scripts directory is on your PATH system environment variable as well (e. g. C:\Python27\Scripts).

Then download the installer file from the link **Mataki Device Build Pack (701.9 kB)** on the following web page:

http://mataki.org/make-devices/

Run the installer and let it install to the default location (should be C:\MatakiBuilder). Then, it will create some directories in that location (mataki and mspgcc). Now, open a command prompt and navigate to C:\MatakiBuilder\mataki\Firmware\Apps\Tracker_NewBuild and type 'scons' to build. If the build task completes with no error it will end up with the message "scons: done building targets." On your command prompt (see Figure 13).



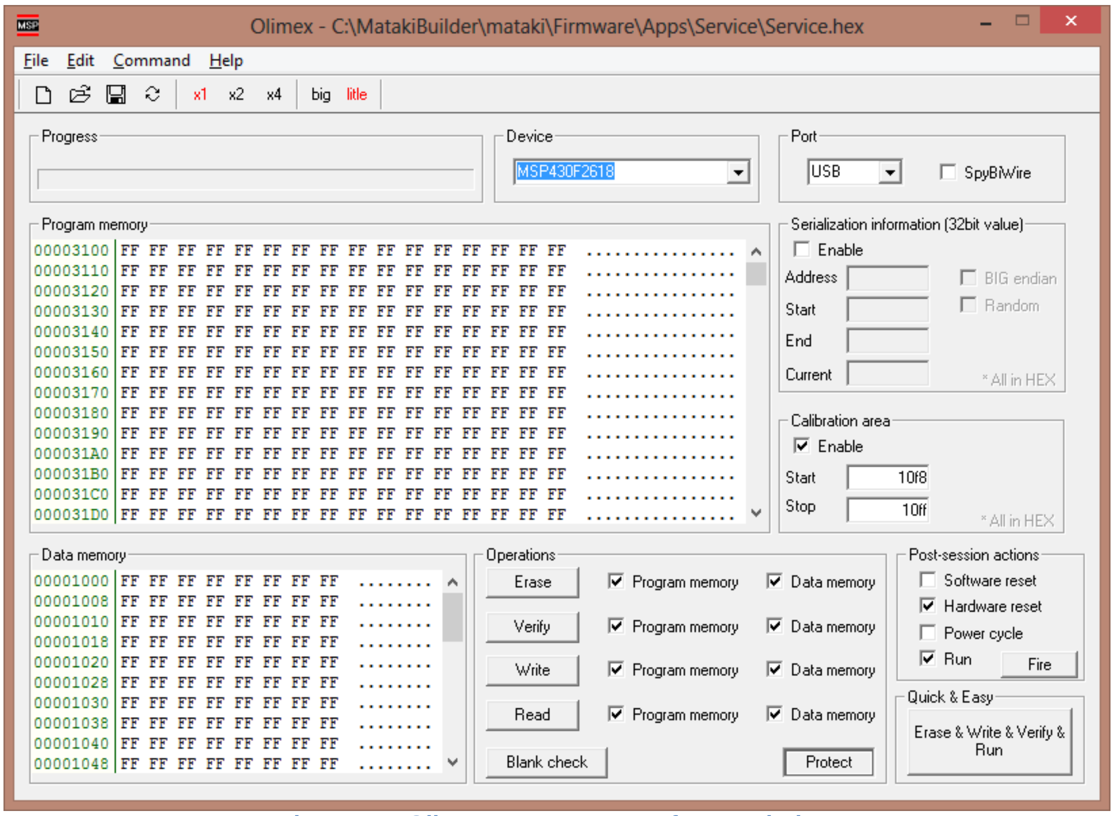
```
e\System\Commands.o C:\MatakiBuilder\mataki\Firmware\System\Configuration.o C:\M
atakiBuilder\mataki\Firmware\System\Interrupts.o C:\MatakiBuilder\mataki\Firmwar
e\System\Logging.o C:\MatakiBuilder\mataki\Firmware\System\Programming.o C:\Mata
kiBuilder\mataki\Firmware\System\RTC.o C:\MatakiBuilder\mataki\Firmware\System\s
mallio.o C:\MatakiBuilder\mataki\Firmware\System\System.o C:\MatakiBuilder\matak
i\Firmware\System\Timers.o C:\MatakiBuilder\mataki\Firmware\System\USCI.o C:\Mat
akiBuilder\mataki\Firmware\System\Watchdog.o C:\MatakiBuilder\mataki\Firmware\Sy
stem\Radio\cc1100.o C:\MatakiBuilder\mataki\Firmware\System\Radio\cc1100_spi.o C
:\MatakiBuilder\mataki\Firmware\System\Radio\cc_layer0.o C:\MatakiBuilder\mataki
\Firmware\System\Radio\Network.o C:\MatakiBuilder\mataki\Firmware\Drivers\ADC_De
vices.o C:\MatakiBuilder\mataki\Firmware\Drivers\BoolBitArray.o C:\MatakiBuilder
\mataki\Firmware\Drivers\CMA3000.o C:\MatakiBuilder\mataki\Firmware\Drivers\Devi
ce.o C:\MatakiBuilder\mataki\Firmware\Drivers\GPS.o C:\MatakiBuilder\mataki\Firm
ware\Drivers\InterfaceBoard.o C:\MatakiBuilder\mataki\Firmware\Drivers\LogEngine
.o C:\MatakiBuilder\mataki\Firmware\Drivers\LogEngine.Sync.o C:\MatakiBuilder\ma
taki\Firmware\Drivers\NMEA.o C:\MatakiBuilder\mataki\Firmware\Drivers\S25FL128P.
o C:\MatakiBuilder\mataki\Firmware\Drivers\SCP1000-D01.o Application.o FSM.o FSM
_Radio.o FSM_Tracking.o C:\MatakiBuilder\mspgcc\msp430\lib\libmspgcc.a C:\Mataki
Builder\mspgcc\msp430\lib\libm.a
python C:\MatakiBuilder\mataki\Firmware\include\msp430-ram-usage.py Tk_NewBd.elf

RAM usage summary:
2286 of 8192 bytes used (5906 free)
the stack can grow up to 5896 bytes (continous memory at end of RAM)
msp430-objcopy -O ihex Tk_NewBd.elf Tk_NewBd.hex
msp430-objdump -DS Tk_NewBd.elf > Tk_NewBd.lst
scons: done building targets.

C:\MatakiBuilder\mataki\Firmware\Apps\Tracker_NewBuild>_
```

Figure 13 - Scons software has built the target

7.3. Reprogramming firmware

The programming software that supports the microprocessor on the device is Olimex MSP Programmer. The software can be found at www.olimex.com and its installation process is fairly straight forward.

Running the software will open up the following application window shown in Figure 14.


Olimex - C:\MatakiBuilder\mataki\Firmware\Apps\Service\Service.hex

Device: MSP430F2618
Port: USB
☐ SpyBiWire

Program memory

```
00003100 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003110 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003120 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003130 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003140 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003150 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003160 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003170 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003180 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003190 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
000031A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
000031B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
000031C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
000031D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
```

Serialization information (32bit value)
☐ Enable
Address
Start
End
Current
☐ BIG endian
☐ Random
* All in HEX

Calibration area
☑ Enable
Start 10f8
Stop 10ff
* All in HEX

Data memory

```
00001000 FF FF FF FF FF FF FF FF  ........
00001008 FF FF FF FF FF FF FF FF  ........
00001010 FF FF FF FF FF FF FF FF  ........
00001018 FF FF FF FF FF FF FF FF  ........
00001020 FF FF FF FF FF FF FF FF  ........
00001028 FF FF FF FF FF FF FF FF  ........
00001030 FF FF FF FF FF FF FF FF  ........
00001038 FF FF FF FF FF FF FF FF  ........
00001040 FF FF FF FF FF FF FF FF  ........
00001048 FF FF FF FF FF FF FF FF  ........
```

Operations
Erase ☑ Program memory ☑ Data memory
Verify ☑ Program memory ☑ Data memory
Write ☑ Program memory ☑ Data memory
Read ☑ Program memory ☑ Data memory

Post-session actions
☐ Software reset
☑ Hardware reset
☐ Power cycle
☑ Run

Figure 14 - Olimex programmer software window

There are a few settings that you need to do before you could read or write firmware into the Mataki's processor through this software.

7.3.1. Reprogramming hardware settings

To start with, you need to connect the Support board to the USB port on your computer, then to connect one end of the JTAG cable to another USB port of your computer and the other end of the JTAG cable to the Support board. The snapshot in Figure 11 shows these connections.

As it could be seen on the same snapshot in Figure 12, the Mataki board should be also connected to the Support board. The power switch SW1 on the Support board should be set to 'on' position to power up the Mataki board. Mataki board is now at program execution status that can be checked by observing the execution messages using Putty software. It is optional to set the Charger switch SW2 on the Support board to 'on' position (the same side as the switch SW1). This will charge the battery if it is connected to the Mataki board.

7.3.2. Reprogramming software settings

On the Olimex window select the correct microcontroller target, i.e. MSP430F2618 on the 'Device' drop-down-menu above the screen. Also, make sure the little-endian mode is checked by clicking on the button 'little' that turns it to red colour, like shown in figure 14. The 'Port' setting must be set on 'USB'. It is optional, and somehow better, if you also check 'Hardware reset' and 'Run' checkboxes on the right side of the screen. Depending on the type of programming, i.e. if it is required to program the program memory in the microcontroller, or its data memory, the associated boxes on the 'Operations' area below the screen can be checked or unchecked. In the case of reprogramming the microcontroller with the original firmware that is left available, all 'Operation' boxes can be checked.

7.3.3. Loading and programming

To deploy a program into the microcontroller memory, the program should first be loaded into the Olimex software. This is achieved in two steps;

1) Depending on the type of programming, you may need to clear the memory of the Olimex software prior to populating it with the new code that you want to take by Olimex software and deploy. You can do so by clearing the Data Memory and Program Memory as shown in figure 16. In the case of reprogramming the microcontroller with the original firmware that is left available, it is recommended that you clear the two memory areas of Olimex every time before loading the code.
2) Load the program into the Olimex software by opening the executable code, i.e. the file with extension name '.hex'. like in the snapshot in figure 17.

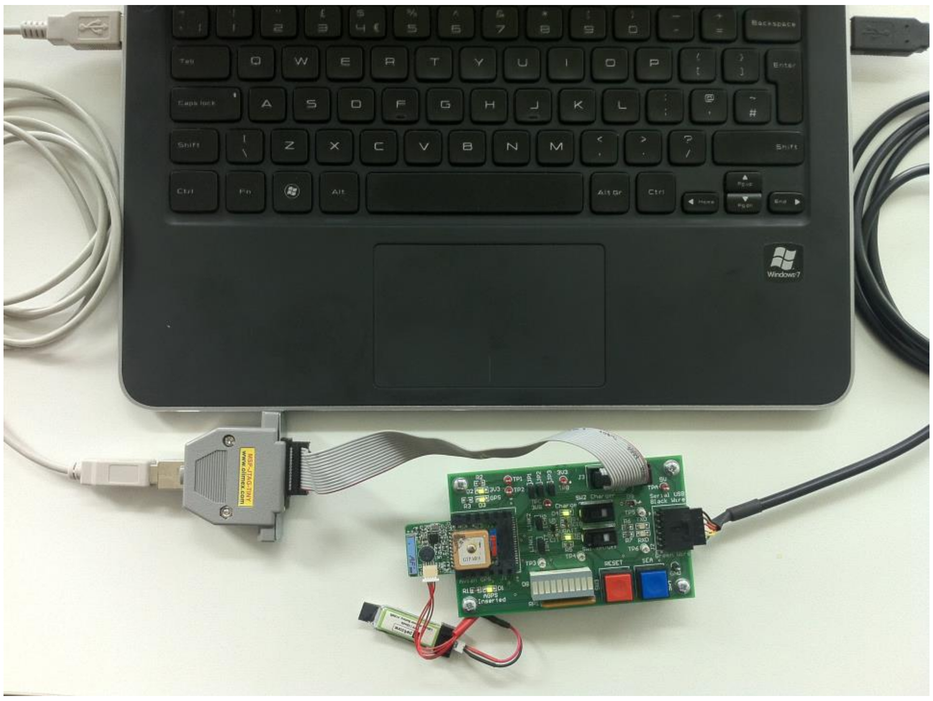
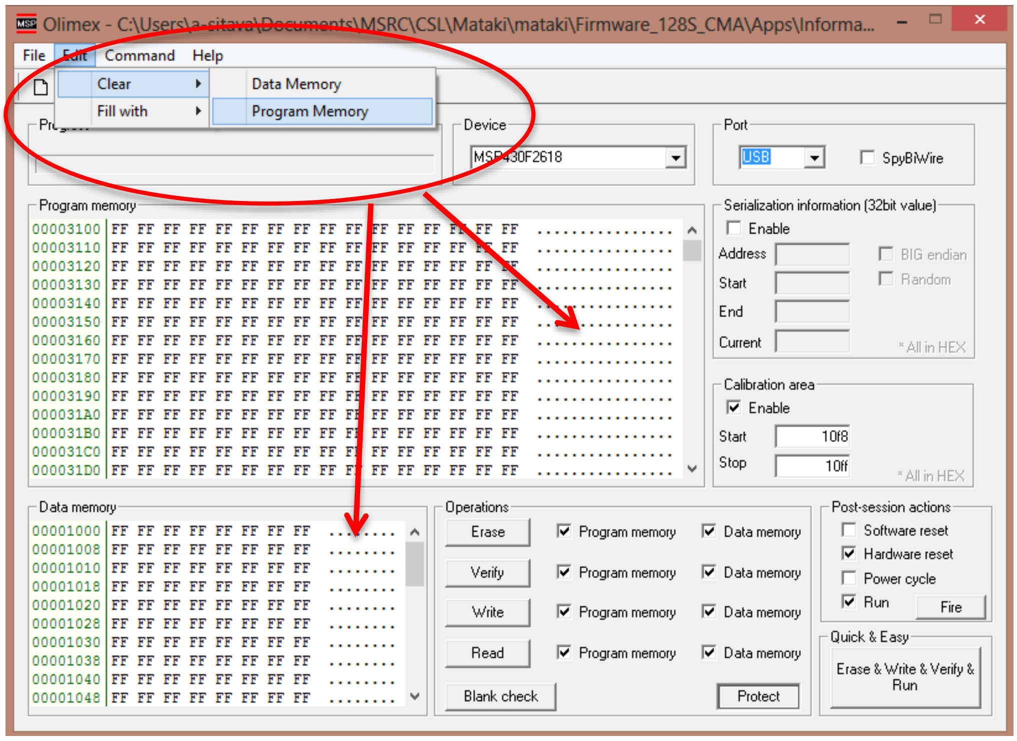


Figure 15 - Hardware connections before reprogramming the Mataki board

After Mataki board is powered up via the Support board, and the program code is loaded into the Olimex software, pressing the button 'Erase & Write & Verify & Run' on the 'Quick & Easy' area below the screen will do the neat job of replacing the whole memory of the microcontroller with the loaded program. If Putty software is running on your computer at the same time, the immediate consequence of the deployment after it is finished, is resetting and the new execution of the firmware.

Further checks that the new firmware is running in the device is using the App command on the Putty software when connected to the device. The App command is introduced in the next chapter together with the other useful commands. If you have changed the Version number of the application on the SConstruct file in the App folder of your code, then you will see the new version number as a result of the App command.

Of course, any of the 'Erase', 'Verify', 'Write', and 'Read' can be used individually depending on the requirements.


Olimex - C:\Users\a-sitava\Documents\MSRC\CSL\Mataki\mataki\Firmware_128S_CMA\Apps\Informa...
File
Edit
Command
Help
Clear
Fill with
Data Memory
Program Memory
Erase
Verify
Write
Read
Blank check
Protect
Fire
Erase & Write & Verify & Run

Device: MSP430F2618
Port: USB ☐ SpyBiWire

Serialization information (32bit value): ☐ Enable; Address; Start; End; Current; ☐ BIG endian; ☐ Random; * All in HEX

Calibration area: ☑ Enable; Start 10f8; Stop 10ff; * All in HEX

Program memory

```
00003100 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003110 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003120 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003130 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003140 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003150 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003160 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003170 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003180 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00003190 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
000031A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
000031B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
000031C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
000031D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
```

Data memory

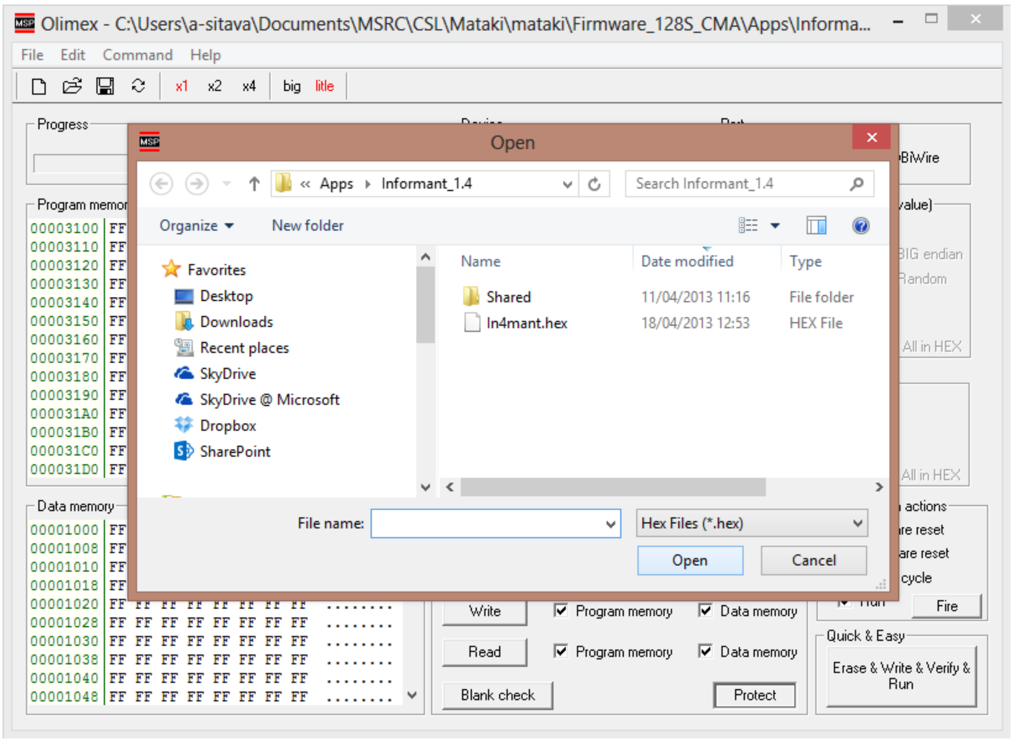
```
00001000 FF FF FF FF FF FF FF FF  ........
00001008 FF FF FF FF FF FF FF FF  ........
00001010 FF FF FF FF FF FF FF FF  ........
00001018 FF FF FF FF FF FF FF FF  ........
00001020 FF FF FF FF FF FF FF FF  ........
00001028 FF FF FF FF FF FF FF FF  ........
00001030 FF FF FF FF FF FF FF FF  ........
00001038 FF FF FF FF FF FF FF FF  ........
00001040 FF FF FF FF FF FF FF FF  ........
00001048 FF FF FF FF FF FF FF FF  ........
```

Operations: Erase ☑ Program memory ☑ Data memory; Verify ☑ Program memory ☑ Data memory; Write ☑ Program memory ☑ Data memory; Read ☑ Program memory ☑ Data memory

Post-session actions: ☐ Software reset; ☑ Hardware reset; ☐ Power cycle; ☑ Run

Quick & Easy

Figure 16 - Clear Data Memory and Program Memory before loading new program



Name	Date modified	Type
Shared	11/04/2013 11:16	File folder
In4mant.hex	18/04/2013 12:53	HEX File

File name:
Hex Files (*.hex)

Figure 17 - Loading new Program code into Olimex software

8. Default Commands - Explanation and Examples

ID

Displays/sets the current identity of the device. For example, on a device configured as number 69:

```
id
```

would return

```
[69:id] 69
```

This command can also be used to update/change the identity of a device. For example, changing device number 69 to 169, and back again:

```
id 169
[69:id] 169
id 69
[169:id] 69
```

This allows the naming of the devices to be flexible, but users should be particularly cautious not to create devices that share a common ID (causing later confusion in base-station communication, and in the resulting recorded data). However, it may be useful to name devices in a particular pattern to distinguish them more easily in the field.

APP

Outputs the name of currently installed firmware on the device. For example, on device number 69, with the ‘Tracker’ firmware installed, the command

```
app
```

may return:

```
[69:app] ‘Tracker’ 1.2 compiled 00:11:05.0000 28-08-2012
```

Alternatively, on device number 66 with the ‘Base’ firmware installed, the same command would return:

```
[66:app] ‘Base’ 1.2 compiled 00:11:05.0000 28-08-2012
```

BATT

Measures the battery voltage. For example,

```
batt
```

may return

```
[69:batt] 3.73V (raw: 2318)(7 ticks)
```

indicating that the connected battery voltage is 3.73 V. When communicating with a device without a battery attached (powered via the support board), this may result in a reading of 0 V or, if the charge switch is set to ON, to a higher voltage corresponding to the system power. When …

FORMAT

Formats the flash memory

```
format
```

READ

Display all log entries. Alternatively, parameters can be used to only display log entries in specified range. For example,

```
read
```

LOGSIZE

Provides the current number of stored log entries. For example on a recently formatted device,

```
logsize
```

may return

```
[69:logsize] {163} slots used
```

indicating that the device contains 163 logs. On a device with significant amounts of data, the same command may return

```
[69:logsize] {12499} slots used
```

indicating that the device contains 12,499 logs.

CHANNEL

Queries and reconfigures the current radio channel. For example,

```
channel
```

Should return the current radio channel of the device (e.g. 0)

```
channel 3
```

Would set the current device to radio channel 3. 128 channels are currently supported, one ‘public’ channel 0, and 127 private channels (1-128). Unless

TIME

This command queries and set the current local time of the device. For example,

```
time
```

will return the current device time, setting the time can be achieved by also entering a datetime in the syntax ‘HH:MM:SS DD.MM.YYYY’

For Example:

```
time 12:30:00 14.02.2011
```

would set the time of the current device to be 12:30pm on the 14th February 2011. For the tracker firmware, local time is continually updated from obtained satellite time, so this command will be overridden whenever a GPS fix is obtained.

UPTIME

Provides the uptime of the device since the last reset. For example, on device number 69, which had been turned on for 25 minutes (without a reset), the command

```
uptime
```

will respond with

```
[69:uptime] 00d 00:25:15
```

TIMERS

Debug command: Provides information about registered software timers (events that are scheduled to happen on the device (e.g. start trying to acquire a fix, go into sleep mode). As an example,

```
timers
```

may return:

```
[69:app] 0 timer(s)
```

if no timers are registered, or:

```
1: 0x9ca8 in 250160 ticks (30 seconds)
[69:timers] 1 timer(s)
```

if 1 event is scheduled to occur in 30 seconds.

RR

Provides information about the reset reason of the device.

```
rr
```

Returns:
0 for WD_RESET_REASON_UNKNOWN
1 for WD_RESET_REASON_WATCHDOGGUARDVIOLATION
2 for WD_RESET_REASON_REQUESTED (e.g. the "reset" command)
3 for WD_RESET_REASON_RADIO (e.g. the radio driver crashed)
4 for WD_RESET_REASON_WATCHDOG (e.g. a long running operation was too long or a function hung)
5 for WD_RESET_REASON_USER1 (unused)
6 for WD_RESET_REASON_USER2 (unused)

7 for WD_RESET_REASON_USER3 (unused)
8 for WD_RESET_REASON_USER4 (unused)

RESET

Resets the device

```
reset
```

Forces the device to reset, setting the reset reason to 2 (WD_RESET_REASON_REQUESTED), see ‘rr’

ABORT

Aborts the current schedule, preventing the devices from entering into autonomous tracking or base-station mode. Allows the user to continue to interact with the device without scheduled events occurring.

```
abort
```

HELP

Displays help on the commands available on the device. A list of commands, with brief descriptions of their function is presented. For example, on a tracking device:

```
help
```

responds with the following;

```
Commands:
 help     [0x34fa]
 id       [0x38c8] Configuration: node id
 app      [0x3ae6] Configuration: application's metadata
 uptime   [0x42ce] Uptime since last reset
 time     [0x435a] Manage local time.
 timers   [0x4778] Debug: Provides information about registered software timers.
 reset    [0x53ae] Resets the device
 rr       [0x53c4] Debug: Provides the recent reset reason
 cc       [0x5f7c] Debug: reserved for firmware developers
 ccidle   [0x5f9e] Debug: reserved for firmware developers
 ccoff    [0x5fda] Debug: reserved for firmware developers
 ccget    [0x5fe0] Debug: reserved for firmware developers
 ccfrx    [0x609a] Debug: reserved for firmware developers
 ccrx     [0x5f94] Debug: reserved for firmware developers
 channel  [0x60ee] Radio: communication channel
 light    [0x64ae] Debug: reserved for firmware developers
 batt     [0x6426] Battery voltage (correct results only when GPS is enabled)
 flashtst [0x667c] Debug: reserved for firmware developers
 gps      [0x6f7a] GPS debug interface
 logsize  [0x7266] Number of local log entries
 read     [0x7bac] Displays log entries from a specified range
 format   [0x7aae] Formats the flash memory
 request  [0x87f8] Requests a sequence of remote log entries
 fillgaps [0x86d4] Finds gaps in downloaded logs and requests missing entries from
remote devices
 showgaps [0x859a] Shows gaps in downloaded logs
 scpreq   [0x9274] Triggers temperature and pressure measurment
 f_reset  [0x9612] Formats and resets the device
 abort    [0x9646] Aborts the experiment
 base     [0x9d38] Brings the device to the 'base station' mode in the selected
channel
 setradio [0x9834] Configures the radio experiment
 suppress [0x9f08] Configures down time for the GPS (overrides setgps)
 setgps   [0x9f7c] Configures the GPS experiment
```

9. Tracking Device Commands and Configuration

Two key sets of parameters are available to configure the devices for tracking. These control the gps tracking schedule (setgps), and the radio communication schedule (setradio).

SETGPS with examples

Here, four parameters control the pattern of GPS tracking that the device engages in:

```
setgps <initial on time> <sleep time between fixes> <max time to wait
for fix> <logging time after a fix>

e.g. setgps 60 300 60 1
```

Would set the device to initially stay awake for 60 seconds then, every 300 seconds, try to get a GPS fix for 60 seconds and, once a fix is obtained, continue to record fixes for 1 second.

Example 2:

```
setgps 300 600 60 10
```

Would set the device to initially stay awake for 300 seconds (5 minutes) then, every 600 seconds (10 minutes), try to get a GPS fix for 60 seconds and, once a fix is obtained, continue to record fixes for 10 seconds.

Example 3:

```
setgps 60 300 60 0
```

Here, as 0 effectively stands for infinite duration this would initially set the device to try and obtain a fix for 60 seconds and, if this fails, wake up every 5 minutes, trying to obtain a fix for 60 seconds. However, once a fix is obtained, the device will continue recording fixes till the battery is exhausted. This may be useful if a continuous log is desired, but the device is likely to initially be in conditions where a fix may be hard to obtained (e.g. in a nest for a burrow nesting species).

Example 4:

```
setgps 0 1 0 0
```

Again, as 0 effectively stands for infinite duration this would set the device to start logging immediately, and continue recording until the battery was exhausted.

SETRADIO with examples

Here, three parameters control the pattern of Radio communication that the device engages in:

```
setradio <initial on time> <sleep time between radio heartbeats> <max
time to wait for a response>
```

Example 1:

```
setradio 60 300 10
```

Would set the device to initially stay awake (in radio contact) for 60 seconds then, every 300 seconds, send a heartbeat (see XX), and wait 10 seconds for a response.

Example 2:

```
setradio 60 0 1
```

Would set the device to initially stay awake (in radio contact) for 60 seconds then turn the radio off and maintain that state until the device is reconfigured (0 stands for infinite duration, so here the ‘sleep period’ is set to infinite).

Example 3:

```
setradio 0 1 0 0
```

Again, as 0 effectively stands for infinite duration this would set the device to start logging immediately, and continue recording until the battery was exhausted.

SETACC with examples

Here, three parameters control the pattern of accelerometer sensing that the device engages in:

```
setacc <initial on time> <sleep period between measurements>
<measurement and logging period>
```

Example 1:

```
setacc 60 300 10
```

Would set the device to initially stay awake (in accelerometer measurement and logging) for 60 seconds then, every 300 seconds, measure acceleration and log for 10 seconds.

Example 2:

```
setacc 60 0 1
```

Would set the device to initially stay awake (in accelerometer measurement and logging) for 60 seconds then turn the accelerometer off and maintain that state until the device is reconfigured (0 stands for infinite duration, so here the ‘sleep period’ is set to infinite).

Example 3:

```
setacc 0 1 0
```

Again, as 0 effectively stands for infinite duration this would set the device to start logging immediately, and continue recording until the battery was exhausted.

GPS

GPS debug interface. Typing **`gps`** results in usage information:

```
Usage: gps [operation]
 1:   Full NMEA debug mode (one way, reset to disable)
 2:   Toggle NMEA preview mode
 3:   Toggle Power of the GPS Chip
 4:   Toggle NMEA Debug Handler
 5:   Enable GPS, Start Logging (one way)
 6 X: Set the position update rate to X (X = 1, 4, 5, or 10)
```

Many of these functions are for debugging (to enable preview of the GPS NMEA data), but a key option is option **6** which allows the default update rate of the device to be modified. Here, 4 updates rates are available (1Hz, 4Hz, 5Hz or 10Hz). For example:

```
gps 6 10
```

would set the update rate of the GPS receiver to 10Hz, all future would then be attempted at 10Hz (so 1 second of logging may result in 10 positions – however, obtaining this logs per second appears easier over longer periods, perhaps due to the amount of time taken getting the initial fix).

SUPPRESS

Configures down time for the GPS (overrides setgps)

```
suppress <hh of suppress activation> <hh of suppress deactivation>
```

For example:

```
suppress 10 13
```

would result in the GPS being disabled when time >= 10 hrs and time < 13 hrs (UTC) resulting in GPS activity from 10:00:00 till 12:59:59.

`suppress` has no effect when the time isn't correct (when a correct time has not been received from GPS data, thus the suppression requires at least one correct fix to be obtained).

```
suppress 10 10
```

disables this feature (when HH are equal the condition above is always false).

BASE

Brings the device to the ‘base station’ mode on the specified channel, for example

```
base 23
```

would put the tracking device into radio communication mode and move it to channel 23:

```
base 23
[69:base] at channel {23}
```

Querying the channel shows that the device is now on channel 23:

```
channel
[69:channel] 23
```

*** Importantly, once the device has entered the base communication mode, it sets a timeout of 10 minutes (600 seconds), after which it will reset to channel 0 and continue logging. This prevents problems with scenarios where the device may only have intermittent contact with a base station, and might otherwise have been stuck in base communication mode. However, manually setting the channel causes no such timeout, and can leave devices stuck on non-public channels unable to communicate with available base-stations ***

F_RESET

Formats and resets the device

```
f_reset
```

Would cause the current device to erase all logs from it’s memory, and then reset (and therefore enter the configured logging regime). This command can be send remotely to cause devices that have been downloaded to erase previous logs and reset

*** This command is irreversible, use with caution ***

SCPREQ

On devices with a pressure sensor installed, this command triggers temperature and pressure measurement, and stores the result in the log.

SHOWGAPS

Shows gaps in downloaded logs

```
showgaps <local sequence no> <nodeid> <first remote sequence no>
<last remote sequence no>
```

For example,

```
showgaps 2300 32 0 1000
```

would search the local memory (already downloaded logs) from sequence id 2300 to identify missing logs from the sequency of logs from node 32 from 0 to 1000 (the first 1000 log entries from node 32).

FILLGAPS

Finds gaps in downloaded logs and requests missing entries from remote devices

```
fillgaps <local> <nodeid> <first> <last>
```

For example,

```
fillgaps 2300 32 0 1000
```

similarly, to **`showgaps`** would search the local memory (already downloaded logs) from sequence id 2300 to identify missing logs from the sequence of logs from node 32 from 0 to 1000 (the first 1000 log entries from node 32), and would then additionally request those missing log entries form node 32.

REQUEST

Requests a sequence of remote log entries

```
request <nodeid> <first> <last>
```

For example,

```
request 32 0 1000
```

would request the first 1000 log entries from node 32.

10. Base Station Commands and Configuration

Devices configured with the base-station firmware are configured to listen for ‘heartbeats’ from tracking devices on the public channel (channel 0). When a heartbeat is detected, the base station will request that the tracking device move to a private channel (the channel corresponds to the base stations ID, so a base station with ID 42 would request a move to private channel 42). Once on a private channel the base station will start to request and download logs from the tracking device. This process can be noisy (due to environmental factors, or radio collisions), so the base station will initially request all logs, then search for missing entries, and request those entries. This process is repeated a number of times, until the less than 5% of entries are missing (this prevents devices spending undue time and power trying to acquire a small number of missing logs).

10.1. An overview of the automatic download process

Heartbeats are send my tracking devices whenever the wake from radio sleep. Three heartbeats are sent to mitigate against possible radio packet loss. Heartbeats contain information on device ID, uptime, numbers of logs, etc. For example, the following shows three heartbeats from device 69 (received by base station 12):

```
[12:L] Heartbeat from 69 [ Uptime: 89(0h), LogSize: 1147, New Logs: 101, New
Fix: 0, Voltage 0.00V (0) ]
[12:L] Heartbeat from 69 [ Uptime: 89(0h), LogSize: 1147, New Logs: 101, New
Fix: 0, Voltage 0.00V (0) ]
[12:L] Heartbeat from 69 [ Uptime: 89(0h), LogSize: 1147, New Logs: 101, New
Fix: 0, Voltage 0.00V (0) ]
```

On detecting these heartbeats the base station (12) requests a move to a private channel and begins the process of downloading:

```
[12:L] Connecting to device id {69} on channel {12}... [00:30:59.0202 01-01-
2010]
 [12:L] Connected to device id {69} on channel {12} [00:31:04.5350 01-01-
2010]
 [12:L] Requesting newest data from {69} [00:31:06.1696 01-01-2010]
Local Sequence No:       2845
Target NodeID:           69
First Sequence No:       1047
Last Sequence No:        1147
............................................................................
.............
[12:L] Received {89} of {101} new entries from {69} [00:31:13.3368 01-01-
2010]
[12:L] Requesting missing data from {69} [00:31:13.3564 01-01-2010]
[12:L] Scheduled the operation. Wait for the schedule to complete!
[00:31:13.3759 01-01-2010]

[12:L] Processing remaining 1147 items... [00:31:14.3752 01-01-2010]
Local Sequence No:       1
Target NodeID:           69
First Sequence No:       1
Last Sequence No:        416
Missing sequence numbers (searching...):
Detected missing:        100% (416/416)
............................................................................
............................................................................
............................................................................
............................................................................
..................................................................
[12:L] Processing remaining 731 items... [00:31:20.2325 01-01-2010]
Local Sequence No:       1
Target NodeID:           69
First Sequence No:       417
Last Sequence No:        832
Missing sequence numbers (searching...):
Detected missing:        100% (416/416)
```

```
............................................................................
............................................................................
............................................................................
............................................................................
..................................................................
[12:L] Processing remaining 315 items... [00:31:24.2967 01-01-2010]
Local Sequence No:       1
Target NodeID:           69
First Sequence No:       833
Last Sequence No:        1147
Missing sequence numbers (searching...):
Detected missing:        71% (226/315)
............................................................................
............................................................................
................................................
[12:L] Schedule processing completed. [00:31:27.4229 01-01-2010]
 [12:L] Requested {1058}, received {940} ({940} in total) [00:31:27.4560 01-
01-2010]
[12:L] Requesting missing data from {69} [00:31:29.0914 01-01-2010]
[12:L] Scheduled the operation. Wait for the schedule to complete!
[00:31:29.1106 01-01-2010]

[12:L] Processing remaining 1147 items... [00:31:30.1100 01-01-2010]
Local Sequence No:       1
Target NodeID:           69
First Sequence No:       1
Last Sequence No:        416
Missing sequence numbers (searching...):
Detected missing:        11% (46/416)
..........................................
[12:L] Processing remaining 731 items... [00:31:34.6307 01-01-2010]
Local Sequence No:       1
Target NodeID:           69
First Sequence No:       417
Last Sequence No:        832
Missing sequence numbers (searching...):
Detected missing:        11% (46/416)
........................................
[12:L] Processing remaining 315 items... [00:31:37.0584 01-01-2010]
Local Sequence No:       1
Target NodeID:           69
First Sequence No:       833
Last Sequence No:        1147
Missing sequence numbers (searching...):
Detected missing:        8% (26/315)
........................
[12:L] Schedule processing completed. [00:31:39.2210 01-01-2010]
 [12:L] Requested {118}, received {106} ({1046} in total) [00:31:39.2541 01-
01-2010]
 [12:L] Requesting missing data from {69} [00:31:40.7087 01-01-2010]
[12:L] Scheduled the operation. Wait for the schedule to complete!
[00:31:40.7281 01-01-2010]

[12:L] Processing remaining 1147 items... [00:31:41.7275 01-01-2010]
Local Sequence No:       1
Target NodeID:           69
First Sequence No:       1
Last Sequence No:        416
Missing sequence numbers (searching...):
Detected missing:        0% (4/416)
....
[12:L] Processing remaining 731 items... [00:31:46.3141 01-01-2010]
Local Sequence No:       1
Target NodeID:           69
First Sequence No:       417
Last Sequence No:        832
Missing sequence numbers (searching...):
Detected missing:        1% (6/416)
.....
[12:L] Processing remaining 315 items... [00:31:48.4007 01-01-2010]
Local Sequence No:       1
Target NodeID:           69
First Sequence No:       833
Last Sequence No:        1147
Missing sequence numbers (searching...):
Detected missing:        0% (2/315)
..
```

```
[12:L] Schedule processing completed. [00:31:50.4651 01-01-2010]
 [12:L] Requested {12}, received {11} ({1057} in total) [00:31:50.4975 01-
01-2010]
[12:L] Detected less than 5 percent missing and got some data.
[00:31:50.5179 01-01-2010]
 [12:L] Returning to channel {0} [00:31:52.5594 01-01-2010]
[12:L] Got {89}/{101} new and {1057}/{1058} missing from {69}.
[00:31:52.5832 01-01-2010]
[12:L] Waiting for new trackers... [00:31:52.6031 01-01-2010]
```

A number of things are worth noting here. The first request from the base station (12) to the tracker (69) tried to obtain only the newest logs from the tracking device (each dot corresponds to a log entry):

```
[12:L] Requesting newest data from {69} [00:31:06.1696 01-01-2010]
Local Sequence No:       2845
Target NodeID:           69
First Sequence No:       1047
Last Sequence No:        1147
.........................................................................
............
[12:L] Received {89} of {101} new entries from {69} [00:31:13.3368 01-01-
2010]
```

The local sequence number is the point in the base stations memory to search for any missing entries from (here it’s at the end of the memory as this is new connection to a tracking device and no new logs have been downloaded). The target nodeID is the tracker ID, the ‘First Sequence No’ and ‘Last Sequence No’ correspond to the reported logsize in the tracker heartbeat:

```
[12:L] Heartbeat from 69 [ Uptime: 89(0h), LogSize: 1147, New Logs: 101, New
Fix: 0, Voltage 0.00V (0) ]
```

minus the number of new logs (therefore specifying the sequence numbers of the remote logs). The base station then reports that is received 89 of these 101 entries. The base station then searches for and requests missing data from the tracker. This is done in blocks of 416 logs for memory & efficiency purposes:

```
[12:L] Requesting missing data from {69} [00:31:13.3564 01-01-2010]
[12:L] Scheduled the operation. Wait for the schedule to complete!
[00:31:13.3759 01-01-2010]

[12:L] Processing remaining 1147 items... [00:31:14.3752 01-01-2010]
Local Sequence No:       1
Target NodeID:           69
First Sequence No:       1
Last Sequence No:        416
Missing sequence numbers (searching...):
Detected missing:        100% (416/416)
.........................................................................
.........................................................................
.........................................................................
.........................................................................
...................................................................
```

Here, the base station searches it’s whole memory (from sequence number 1) for any missing entries on the remote device (in the first 416 remote logs). As the tracker has not been downloaded before, all entries are detected as missing and download is attempted.

This process is repeated for each chunk of logs up to the total logsize, and then repeated until the percentage of missing logs in < 5%.

10.2. Commands

SHOWGAPS

Shows gaps in downloaded logs

```
showgaps <local sequence no> <nodeid> <first remote sequence no>
<last remote sequence no>
```

For example,

```
showgaps 2300 32 0 1000
```

would search the local memory (already downloaded logs) from sequence id 2300 to identify missing logs from the sequency of logs from node 32 from 0 to 1000 (the first 1000 log entries from node 32).

FILLGAPS

Finds gaps in downloaded logs and requests missing entries from remote devices

```
fillgaps <local> <nodeid> <first> <last>
```

For example,

```
fillgaps 2300 32 0 1000
```

behaves in a similar way as **`showgaps`**, searching the local memory (already downloaded logs) from sequence id 2300 to identify missing logs from the sequency of logs from node 32 from 0 to 1000 (the first 1000 log entries from node 32), *and would then additionally request those missing log entries form node 32*.

REQUEST

Requests a sequence of remote log entries

```
request <nodeid> <first> <last>
```

For example,

```
request 32 0 1000
```

would request the first 1000 log entries from node 32.

F_RESET

Formats and resets the device

```
f_reset
```

Would cause the current device to erase all logs from it’s memory, and then reset (and therefore enter the configured logging regime). This command can be send remotely to cause devices that have been downloaded to erase previous logs and reset

*** This command is irreversible, use with caution ***

FORCEDL

Forces a download from a selected tracker. Restarts the base station state machine.

```
forcedl <nodeid> <r.logsize> <r.newlogs>
```

```
For example:
```

```
forcedl 32 500 [r.newlogs]
```