debug innovations

Mataki-Lite User Guide

Version 2

This document provides an in depth guide to the Mataki-Lite GPS tracker, including descriptions of hardware features and how to effectively use it for experiments.

Version History

Version	Date	Changes
1	16 March 2018	First Release
2	23 October 2019	Updated for firmware V1.2.3

Related Documents

PuTTY Installation and Configuration
Mataki-Lite Programming Guide
Mataki-Lite EMBASIC Reference
Mataki Support Board User Guide

Contents

1. Introduction

Mataki-Lite tags are the lightest member of the Mataki family weighing in at around 3.5g excluding a battery. Unlike Mataki-Classic, this is a tracker only device. It can upload its log data to a Mataki-Classic or Pi-Base base station.

The tracking and logging behaviour of Mataki-Lite tags is entirely controlled by a user-supplied script giving total control over the logging process and allowing species-specific behaviour.

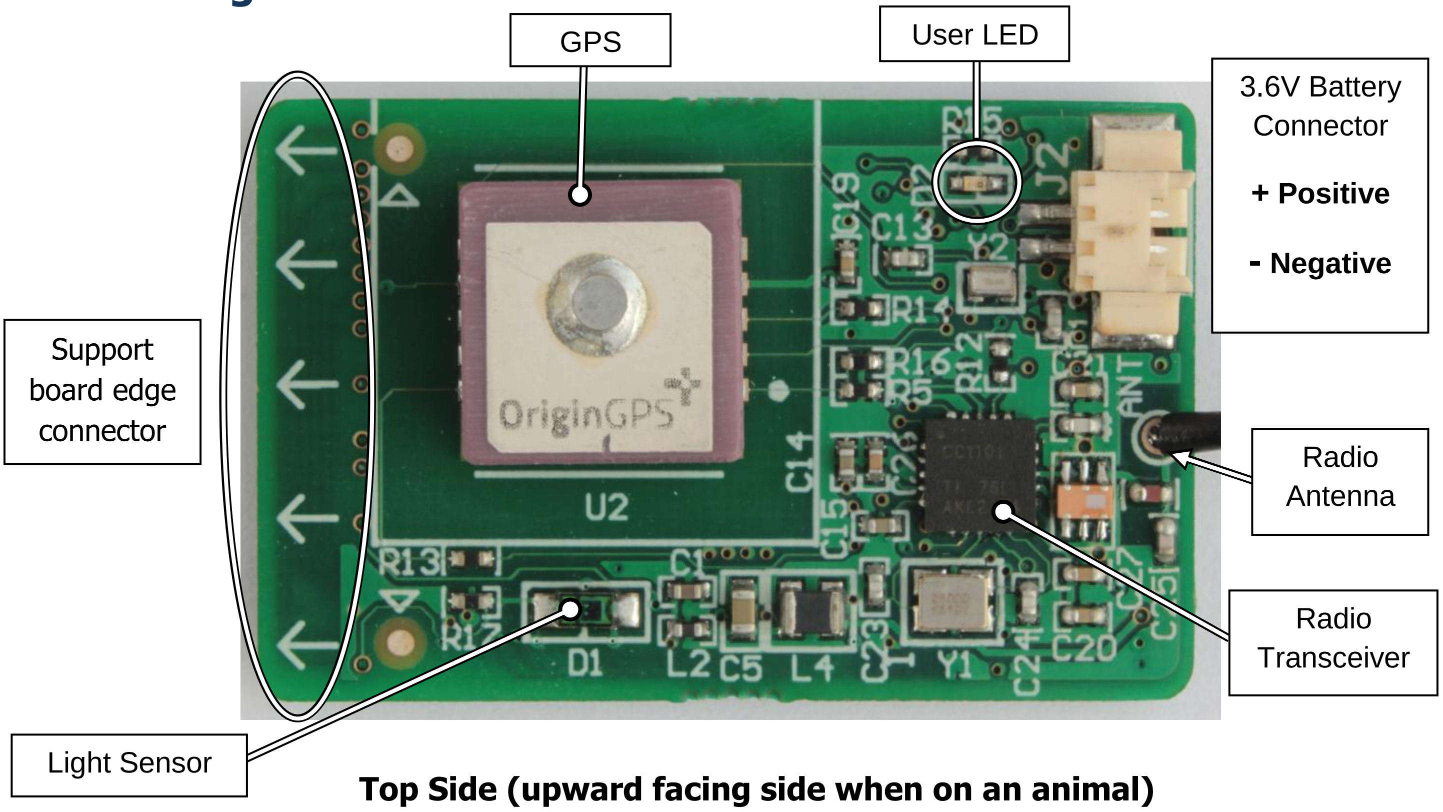
Unless otherwise stated, this document refers to V1.2 PCBs running V1.2.3 firmware.

2. Specifications

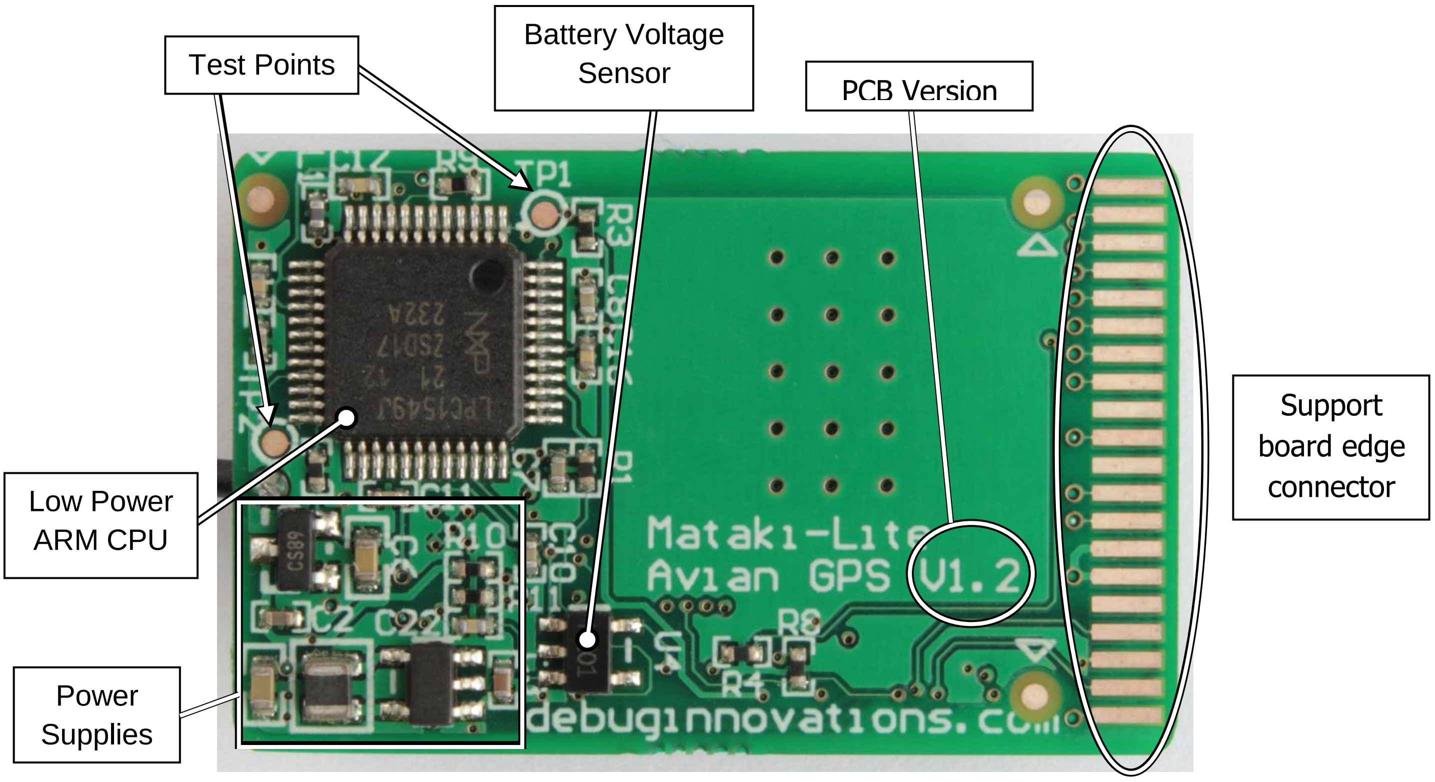
Size	34 x 21.75mm
Weight	3.5g ± 0.2g
Radio Base Frequency	868MHz (Europe) 916MHz (USA)
Max. Transmit Power (at antenna)	10mW (+10dBm)
Battery (not supplied)	3.6V Lithium
Log Capacity	3184 Entries
Max. Script Size	12KB
Battery Voltage Sensor	✓
GPS Position	✓
Light Sensor	✓
Fully Scriptable	✓
Script Upload Over Radio *	✓
Current Consumption **	Typical Figures
Sleeping (`_GPS = -1`)	< 10 µA
Sleeping (`_GPS = 0`)	40 µA
Obtaining Fix (`_GPS = 1`)	30-40 mA
Radio Comms (`_GPS = 0`)	20 mA

* Requires a Pi-Base base station
** Older V1.1 PCBs consume around 20µA more

3. Tag Hardware



Top Side (upward facing side when on an animal)



Bottom Side (side touching the animal)

3.1. Board Orientation

Referring to the photos above, components are laid out such that the bottom side is mostly flat and can rest on an animal.

The GPS antenna (the metal square on top of the GPS receiver) must be able to 'see sky' to receive a satellite signal so it is on the top side.

The light sensor obviously needs to be located on the top side too. Take care not to cover the light sensor with wrapping.

3.2. Comparison with Mataki-Classic

Mataki-Lite shares some of its design with Mataki-Classic which we developed previously. Mataki-Lite uses the same support board as Mataki-Classic so the width of the tag is the same but Mataki-Lite is 10mm shorter in length. The support board edge connector is the same as are most of the pin functions, see Mataki Support Board User Guide.

Mataki-Lite doesn't have an On/Off switch. As a result, it can only be powered from its battery or the support board charger. The 3.8V battery replacement power is not used by Mataki-Lite.

As the name suggests, Mataki-Lite is around one third of the weight of Mataki-Classic. This has been achieved mainly by using a smaller GPS module although the same board space is required due to the GPS antenna clearance requirements. The GPS is powered from a 1.8V supply and this is not a high enough voltage to light the GPS light on the support board.

Mataki-Classic has a large flash memory chip for log storage. This is required in the base station role in order to contain all the logs from all the tracker tags. Since Mataki-Lite can't be used as a base station, it requires a lot less flash, so the CPU's internal flash storage is used.

Mataki-Classic has an accelerometer and air pressure/temperature sensor. Mataki-Lite is designed primarily for GPS tracking so doesn't have these sensors. However, it does retain the light sensor and battery voltage measurement features.

Both tags use a UART interface to communicate with the PC and a PuTTY terminal as a command interface. However, on Mataki-Classic tags, the command interface is used to set logging parameters while the logging and radio communications continues in the background. On Mataki-Lite the command interface is used to write and run a script which does the logging when it is running. Consequently at the command prompt, no logging is happening and the tag will not contact the base station. See section 4 for more information on scripts.

3.3. Electronics

Power
Tag power is supplied from a 3.6V battery connected to J2 (see section 3.4). This is regulated down to 3.3V for the CPU and 1.8V for the GPS. The regulators are located on the bottom side with further filtering components on the top side.

The battery voltage can be read by the script to modify the tag's behaviour if the battery is getting low. If a battery is connected to a tag, it can be charged by plugging the tag into a support board. The 3.8V 'battery substitute' on the support board is not used.

CPU
The CPU is an ARM Cortex-M3 with 256KB of flash memory. 128KB is used for the firmware, 112KB is used for log entries, 12KB is used for the script which is loaded into RAM when the tag starts up and 4KB is used for persistent settings such as the node ID.

GPS
The GPS is an Origin ORG1411 'Nano Hornet'. It emits standard NMEA messages which can be seen in GPS debug mode (`_DBGGPS=1`). There is more information on the GPS in the Mataki-Lite EMBASIC Reference.

Light and Sea Sensors
The light sensor can be read by the script and used to detect when an animal goes into a burrow for example. A pin on the edge connector is used as an analogue input to detect sea water conduction to GND or other pins. See Mataki-Lite EMBASIC Reference.

Radio and Antenna
See section 3.5.

User LED
There is a red LED on the top of the board. This comes on when the board is reset and can be controlled by the script. You may not want to use this in a tracking script to save battery power and so as not to affect the animal.

Test Points
There are two test points on the bottom of the board which can be set by the script. They are both outputs and by default, they are set high. They have 10K pull-ups on them so, if they are set low, the tag will consume more power. There is more information on test points in the Mataki-Lite EMBASIC Reference.

3.4. Battery

Mataki-Lite is supplied as a circuit board. You will need a battery for each tag and the means to attach it to your animal of choice. Because the animal dictates the attachment method and the size of battery, we do not supply these parts.

The power is supplied from a 3.6V Lithium-Ion or Lithium-Polymer rechargeable battery. The battery connector is a Molex Pico-Blade series (Farnell part number 112-5373). This connector is widely used on small batteries for model aircraft, drones etc. However, the pins are not always the polarity shown.

It is recommended that pre-made cables with connectors are bought for this, such as: https://www.digikey.com/short/qjd0nz. Batteries with the right connector can be bought, but these often have the polarity of the pins incorrect. These can still be used by following the instructions below...

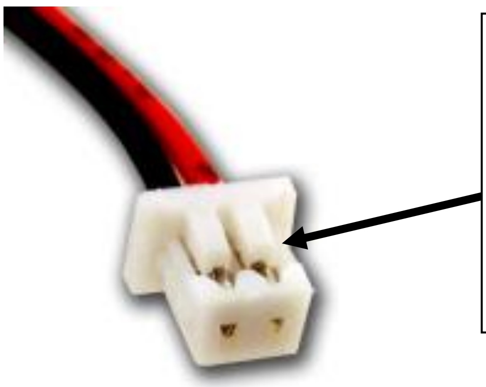


PHOTO SHOWS INCORRECT POLARITY

To use with Mataki-Lite, insert a very small screwdriver or pair of tweezers under these plastic latches and lift them up. The wires can then be removed and inserted in the opposite pin order to reverse the polarity.

Molex Pico-Blade Connector

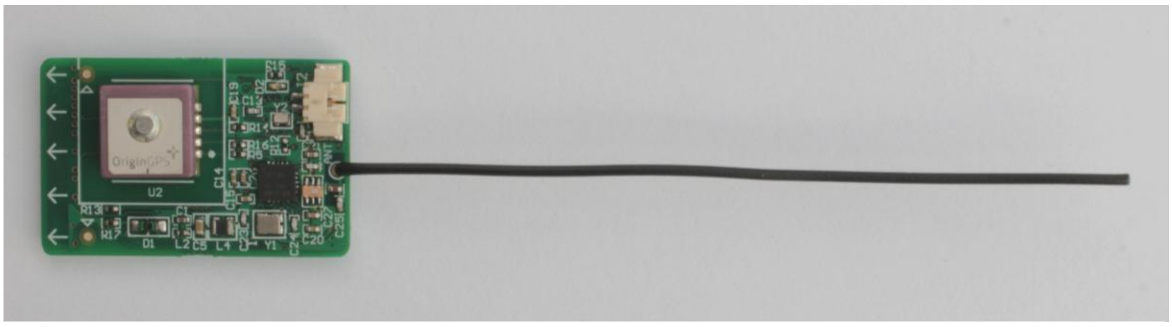
WARNING

Check the polarity of the battery very carefully before inserting as inserting it the wrong way round is likely to cause permanent damage to the tag

3.5. Radio

The radio transceiver operates in the ISM (Industrial, Scientific & Medical) band. No license is required to transmit in this band provided you stick to the published rules. In Europe this band is 868-870MHz (generally known as 868MHz) and in America it is 902-982MHz (generally known as 915MHz). You must use the correct tag settings for the region in which you are using it.

Most new tags will come with a wire antenna soldered to the hole provided (shown below). Tags with a frequency sticker have been factory set to that frequency and the antenna will have been cut to the correct length. Tags without a sticker will be set to 868MHz and the antenna will be cut for 868MHz. To change the frequency, see section 6.2.



Mataki-Lite with Supplied Antenna

It is possible to attach your own antenna as an alternative to the wire antenna. There is no DC blocking capacitor on the PCB. If you intend to use certain types of antenna or long cable runs, we recommend putting a 100nF ceramic capacitor at the PCB end. You must also be careful not to exceed the radiated power limits at the antenna. If you connect a high gain antenna for example, you may be operating outside the ISM rules. The tags we supply will operate within the ISM rules if used in the correct region and with the supplied wire antenna.

The firmware supports 3 different radio modes, all of which are controlled by the tag's script (for further details see Mataki-Lite EMBASIC Reference):

- Automatic Mode, which uploads logs to a base station and receives an updated script if the base station has one
- Private Message Mode, used to send and receive small messages between scripts on other tags
- Test Mode, for generating a carrier signal. In principle this could also be used to generate CW 'pips' in a script but be aware of local regulations.

3.6. Radio Compatibility

Mataki-Lite radio messages are compatible with Mataki-Classic tags and mixed operation is supported (Lite and Classic operating at the same time using the same base station). The logs produced by Mataki-Lite and Mataki-Classic tags are identical.

All tags, whatever the type, must have unique IDs or the base station logs will mix up the tags. Compatibility between tags is only possible if the firmware versions are compatible. The following table shows which firmware versions are compatible with which PCBs and Mataki-Classic base stations.

Mataki-Lite Firmware Version	Mataki-Lite PCB Version	Mataki-Classic Firmware Version
1.1.1	1.1	5.3.3
1.2.2	1.1 & 1.2	5.4.4
1.2.3	1.1 & 1.2	5.4.4

In general we recommend running the latest firmware on all tags. See Mataki-Lite Programming Guide for instructions on how to load new firmware.

Mataki-Lite tags running V1.2.2 firmware onwards are also compatible with Pi-Base base stations.

4. Scripts

Mataki-Lite tags need to have a script loaded onto them before deployment. Scripts are written in the EMBASIC language, a dialect of BASIC. See the Mataki-Lite EMBASIC Reference for details on the language.

Initially all scripts are loaded from a .bas (BASIC) file on a PC using the load_script utility with the tag plugged into a support board. The script loader cannot be used at the same time as PuTTY is connected as they use the same COM port. Once loaded, the script can be overwritten by a base station over the radio or saved locally with the `SAVE` command.

4.1. Script Load Process

The script is stored in the flash memory and loaded into RAM on start up. When the script is loaded, text like this is shown on the PuTTY terminal...

```
EMBASIC V1.4  Copyright (c) Debug Innovations 2017-2019

Stored script loaded...
File     : hardware_test.bas
Size     : 4215 bytes (34%)
Modified : 10:47:21 01-02-2018 (UTC)
Username : Dave
Uploaded : 16:52:30 18-02-2018 (UTC) via support board
Base Up  : Allowed
Auto-run : OFF
```

If the script is successfully loaded, `Stored script loaded...` is displayed and the details are listed. If the script cannot be loaded, an error is displayed and a new script must be put on the tag.

File is the name of the file that the script came from when it was loaded from the PC or base station, or the name it was given when it was saved locally using the `SAVE` command.

Size is the size of the tokenised script in bytes and as a percentage of the 12KB space available. This is not the same as the original file size.

Modified is the time the script was last changed. It can be used to compare versions of scripts between tags as identical scripts will have the same time, regardless of when they were put on the tag. If the script was loaded from a PC with the script loader or from a base station over the radio, then this is the same as the operating system file modification time. If the file was saved locally, this is the tag's time when saved.

Username is the login name on the PC the script was loaded from. If the script was loaded by a base station, it contains the type of base station e.g. Pi-Base. If the script was saved locally, it will contain a dash.

Uploaded is the time it was put on the tag. After the time it says how the script got on the tag, either from a PC when it was plugged into a support board or from a base station over the radio. If the script was saved locally, it will say 'Local SAVE'.

Base Up says whether a base station that supports uploading new scripts can overwrite the script. Normally this would say 'Allowed'. If it says 'Not allowed (SCRIPT LOCKED)', the base station will not overwrite it (see `_SCRIPTLOCK` in Mataki-Lite EMBASIC Reference).

If the **Auto-run** setting is 'ON', the script will start 10 seconds after starting the tag (unless cancelled from the PuTTY terminal) and a message like this will be shown...

```
[1:L] About to start script, press C to cancel
        -> Start in 9 secs
```

When a script is loaded using the script loader tool, Auto-run can be set to ON or OFF. This is the only way a user can change the Auto-run setting.

When a script is loaded by a base station over the radio, Auto-run is set to ON. This is because the currently running script needs to be stopped while it is overwritten, then the new script is copied to flash and the base station resets the tag. If Auto-run was OFF, the new script would fail to run, effectively disabling the tag.

When a script is saved locally, the current Auto-run setting is left unchanged. This is so a developer can save many times without altering whichever setting he/she wants to use.

Note: It is important to have Auto-run set to ON when deploying tags, otherwise the tag will sit idle until the battery goes flat.

4.2. Script Hardware Limitations

The maximum size of a script is 12KB. This sounds small but it is enough for quite a large BASIC program. Note that the file size of the script when stored on a PC may be larger than 12K because the BASIC keywords are tokenised - the script loader shows the actual size of the stored script.

The maximum amount of RAM space for BASIC variables is around 4KB. This is plenty for normal variables but you have to be careful with arrays and strings. The emulation has the same memory limits so there should be no surprises when putting scripts on to tags.

Some important elements that need to be considered whilst writing scripts are given below.

4.2.1. Getting a Valid Clock Time

It is crucial that the tag's real-time clock is set so that the EMBASIC time functions work properly, logs have the correct timestamps and firmware functions that rely on the time work properly.

There is no way to manually set the time on Mataki-Lite. The time can come from either the GPS or a base station. Make sure your base station has the correct UTC time.

- All times are in UTC. Don't use local time - it will stop things working.
- The time from EMBASIC functions will 'jump' when the time is updated. This is especially troublesome when the GPS is on, see Mataki-Lite EMBASIC Reference.
- A tag reset from the script or a base station command will not change the clock. A hard reset e.g. red button on the support board or power cycling the tag will reset the clock.

4.2.2. Using the GPS Module

The GPS module consumes a lot of current (30-40mA). It is crucial to battery life that the GPS is only on for the minimum time necessary to get a fix and that it isn't on if there is little or no chance of getting a fix.

- Don't turn on the radio when trying to get a fix - it will degrade the GPS radio performance.
- Fix intervals should be tuned to suit your needs. More regular fixes provide a better map of the path taken at the expense of battery life.
- The time of day, light level and `_GPSSIGNAL` should be used to avoid leaving the GPS on for a long time when it's unlikely to get a fix. Consider the animal's natural behaviour, is it in a burrow/cave/hole etc?
- First fixes can take quite a long time to get (sometimes minutes even in good weather conditions) so adjustments might need to be made for the first fix.
- You might want to ignore fixes that fall beneath a quality threshold.
- A tag reset from the script or a base station command will not turn off the GPS voltage regulator. It will only put the GPS module to sleep. A hard reset or power cycle will turn off the GPS voltage regulator.

4.2.3. Radio and Base Station Contact

Every now and again, the tag needs to contact the base station to upload its logs and possibly receive a new script. The radio consumes about half the current of the GPS (20mA), so it is important that the radio is only on for the time required.

Radio messages may or may not be received due to range, other radio signals, weather and many other reasons. Write your script with this in mind.

- Use the `_CONTACT` variable to help control timeouts on base contact, rather than leaving the radio on for a fixed time
- Once contact has been made, the base station controls the tag, see section 7. When finished, the tag is told to reset itself. Write scripts in such a way that a reset is expected but not guaranteed after making contact with a base station.

4.2.4. Battery Management

The GPS and radio take significant amounts of power. Therefore the script should minimise the time these parts are powered on. The Mataki-Lite EMBASIC Reference covers this in detail.

Just running round a loop in the script still takes a medium amount of current (5mA). To maximise battery life, turn off the GPS and radio and enter `_SLEEP` state for as long as possible.

For some use cases it may be beneficial for the script to look at the battery voltage and to have different behaviour in low battery situations, e.g. stop looking for fixes and start trying to upload the last of the data on the tag to a base station.

4.2.5. Log Space

As mentioned earlier, Mataki-Lite only has space for 3184 log entries. Most tags only use a small battery for weight reasons. Therefore, if you only log GPS fixes, it is very difficult to run out of log space before the battery goes flat. However, if you use a large battery or you log lots of other information (light and sea sensor, battery voltage, anything the script stores etc.) you may run out of log space before the battery goes flat. We suggest you estimate the log usage when writing scripts.

There is a lot more information and script examples in the Mataki-Lite EMBASIC Reference. In addition, Debug Innovations provide an example tracker script which is typically loaded on to new tags. This can be used as a starting point for your own scripts.

4.3. Recovery Mode

If an error occurs in a script, EMBASIC stops running and returns to the BASIC prompt. If this happens after a tag has been deployed, the script will never have a chance to upload the log to a base station. In this situation the log entries that have not been collected will be lost unless the tag is recovered.

As of V1.2.3, a feature called "recovery mode" has been introduced. When a script error occurs, the tag enters into this mode in order to recover as many logs as possible and attempt to get a new script from the base station (if supported).

In Recovery Mode, the tag will attempt to contact the base station and upload its log without the need for the script. All other scripted activities will no longer occur but the log will be sent to the base station if it is in range and the battery has sufficient charge. The base station will then reset the tag. Depending on the script error, the script may be able to continue running, for example some problems may only occur in specific circumstances such as a division by zero error at zero latitude.

If the tag is connected to a support board and being actively interacted with (user is typing in commands) then the behaviour is as before - errors cause the script to stop. However, if the tag has been deployed, script errors will cause entry to Recovery Mode.

5. Tag Start Up

5.1. Powering a Tag

Mataki-Lite tags can be powered from two sources...

- A Battery
    - Simply plug the battery into the tag (checking the polarity first!)
    - The loaded script will start approx. 10 seconds later (if Auto-run is ON)
- A Support Board
    - Insert the tag into the support board
    - Switch SW2 (Charger) into the On position to provide power to the tag (not necessary if a battery is already attached)
    - If a battery is attached, SW2 (Charger) will charge it
    - SW1 is not used for Mataki-Lite

5.2. Start Up Messages

On start up, a large amount of text is displayed on the PuTTY terminal. We will explain it in sections. Some lines are time-stamped. The text below has had the time stamps removed for clarity. A lot can be learned from the start up messages - we recommend reading through this section as it contains information that is not explained elsewhere.

```
Mataki-Lite Radio Tracker Platform V1.2.3
```

The first line is the sign on message and firmware version.

V1.2.3 can be interpreted as follows: V1.2 means 'supports PCB version up to and including PCB V1.2' and the .3 on the end means this is the second release of firmware. So the next release will be V1.2.4 unless it supports another board e.g. V1.3.4. The last number never resets, so V1.1.27 has the same functionality as V1.2.27 except it supports V1.2 PCBs.

If the last number is zero, it is a development build, not a release build, and it may have unusual behaviour!

The version number can be obtained at any time from a PuTTY terminal...

The `VER` command displays the base code version - in this case V1.2.3
The `_VER$` variable can be used to get the version from an EMBASIC script

```
###############################
###    DEVELOPMENT BUILD    ###
###############################
```

If this is a development build (version number ends in zero) a warning sign is displayed. Development builds may have unexpected behaviour. On release builds this message is not shown.

```
Copyright (c) 2016-2019 Debug Innovations Ltd.
Portions Copyright (c) Microsoft Corporation.
Portions Copyright (c) 2012 Freie Universitaet Berlin,
Tomasz Naumowicz.
All rights reserved.
```

The copyright information.

```
PCB : V1.2
```

The Mataki-Lite PCB version which matches the PCB version on the back of the board. The firmware supports multiple PCB versions and can auto-detect which version it is running on and alter its behaviour accordingly.

```
[0:D]  Radio: Registered handler for protocol 2
[0:D]  Radio: Registered handler for protocol 3
```

The part of the firmware that deals with radio messages supports different message protocols. Handlers receive messages and there are more registered handler messages later. Some of the protocols are only used for sending - this table describes all of them:

Protocol	Function
1	Used to send/receive log text
2	Used for remote command requests
3	Used for remote command responses
10	Used for requesting missing logs to fill gaps
11	Used for sending missing logs for gap requests
12	Used for requesting a range of missing logs
13	Used for sending missing logs for range requests
100	Used to send heartbeats
101	Used to receive base station setup instructions
110	Starts a script upload
111	Used to send script data
112	Ends a script upload
200*	For private messages

* This is registered by the firmware the first time private messages are used

```
Flash Init...
Part ID: 0x1549
Using LPC1549 Flash driver
Drive 0: Settings (4K bytes)
Drive 1: BASIC Script (12K bytes)
Drive 2: Main Log (112K bytes)
Flash Init: OK
```

This detects the flash device (on Mataki-Lite it is inside the CPU chip, an LPC1549 from NXP), then partitions the flash into separate drives so the filing system can read and write the flash. Drive 2 contains the user's log. When a user or script writes to the `_LOGCLEAR` PS variable, this erases drive 2.

```
[0:D]  Log: Total slots 3184 (slot size 36)
[0:D]  Log: Slots used 0
[0:L] Log Engine Init: OK
```

The log engine starts up, displays the log capacity and counts how many existing log entries there are in Drive 2. It is responsible for adding new entries.

```
[1:L] Node ID: 1
```

The node ID and all the other settings are stored in Drive 0. Note that the `[0:L]` changes to `[1:L]` at this point to reflect the Node ID, which up until this point had not been read.

```
[1:L] Radio 0 On
[1:D]  Radio 0 (CC1101): Part Number 0, Version 20
```

The radio chip is turned on and the part and version registers are read. The driver expects to see a CC1101 device.

```
[1:D]  Radio base freq: 868000000Hz
[1:D]  Radio freq. cal: -21000Hz
```

The radio's frequency is set. The base frequency comes from the `_FREQBASE` setting and the frequency calibration comes from the `_FREQCAL` setting. The two values are added together, then loaded into the radio chip. The calibration setting is factory set to a value that generates an 869MHz carrier signal with a base frequency of 868MHz when set to channel 20 (the centre of the 2MHz range). This primarily corrects for the crystal oscillator offset. In the example above, the oscillator is about 600Hz higher than it should be. We recommend that you do not change the `_FREQCAL` setting unless you have the necessary equipment to set it again.

```
[1:L] Radio 0 is in RX
[1:D]  Radio: Registered handler for protocol 101
[1:D]  Radio: Registered handler for protocol 110
[1:D]  Radio: Registered handler for protocol 111
[1:D]  Radio: Registered handler for protocol 112
[1:D]  Radio: Registered handler for protocol 1
[1:D]  Radio: Registered handler for protocol 12
[1:D]  Radio: Registered handler for protocol 10
```

The radio is in receive (RX) mode and more handlers are registered.

```
Log Sync Init: OK
System init complete
```

All the protocols needed to synchronise logs with a base station have been loaded. The system is initialised (ready).

```
Notifying base of tag startup
```

Mataki-Lite now sends out a radio message announcing it has started up. If there is a base station within range, the base station display shows a message with the node ID and signal strength plus the tag type and firmware version.

```
Base Time is 14:35:59.000 15-03-2018
```

If there is a base station within range, it will reply with the current time.

```
Tag time set to base time
```

If the time is reasonable (looks like it has been set at the base station) and is significantly different to the tag time, this message is shown highlighted in blue and the tag time is set to the base station time.

At this point, the only significant thing that hasn't happened is the EMBASIC interpreter hasn't loaded or started running the script.

This is to allow the tag to pause for a start up delay (see `_STDLY` in Mataki-Lite EMBASIC Reference).

```
[1:L] Startup delay set, press C twice to cancel
        -> Start in 10 mins
```

If a start up delay is set, the message above is displayed followed by a countdown timer that ticks down every minute until zero where the following is displayed instead:

```
        Startup timer complete
```

If the start up delay is cancelled, the following message is displayed:

```
        Startup timer cancelled
```

If a start up delay is set, regardless of whether it was cancelled, the tag will once again announce that it has started up and may receive the base station time. This is so that, after a long start up delay (could be days), the fact that a tag has started up is useful information to anyone looking at the base station display and it is another opportunity for the tag to get the time if it was out of range on the first occasion. Note that typically, the tag ignores the second base time as it already has an accurate time.

```
EMBASIC V1.4  Copyright (c) Debug Innovations 2017-2019
```

The EMBASIC interpreter now starts. The version number displayed above relates to the interpreter itself and is not related to the firmware version.

```
Stored script loaded...
File     : hardware_test.bas
Size     : 4215 bytes (34%)
Modified : 10:47:21 01-02-2018 (UTC)
Username : Dave
Uploaded : 16:52:30 18-02-2018 (UTC) via support board
Base Up  : Allowed
Auto-run : OFF
```

The EMBASIC interpreter now loads the script from flash into RAM (see section 4). It also loads any PS variables that are stored as settings in the flash.

```
>
```

If Auto-run is OFF, the EMBASIC prompt will be displayed as shown above. Commands can now be typed in. Note that the script will have been loaded and can be seen by typing `LIST`.

If Auto-run is ON, the following message will be displayed:

```
[1:L] About to start script, press C to cancel
        -> Start in 10 secs
```

A 10 second timer counts down every second until zero where the following is displayed instead:

```
Starting Script...
```

If the script start is cancelled, the following message is displayed:

```
Script start cancelled
```

Followed by the EMBASIC prompt:

```
>
```

6. Important Settings

Please check the following settings before tag deployment, as errors in these may cause issues for the tags functionality.

6.1. Node ID

It is important to set a unique ID for each tag before deployment, or radio communication will be impaired and the base station will mix up the logs from tags with the same ID.

To assign a tag an ID, use the `_ID` command on the EMBASIC command prompt or inside a script. This value is stored in the flash and will be saved between device resets. See the Mataki-Lite EMBASIC Reference for more information on how to do this.

6.2. Radio Base Frequency

The radio base frequency (used as channel 0) must be set depending on the region where the tag will be deployed, see section 3.5.

To set the base frequency, use the `_FREQBASE` command on the EMBASIC command prompt or inside a script. The device requires a reset for the base frequency to be set. This value is stored in the flash and will be saved between device resets. See the Mataki-Lite EMBASIC Reference for more information on how to do this.

6.3. Supplementary Data

Mataki-Lite tags are also capable of sending supplementary data detailing extra information about the tag, helping identification and sorting retrieved data. Currently, there are two types of supplementary data that should be set on a tag:

- Owner
  - Stored in the EMBASIC PS variable `_OWNER$`
  - Typically the name of the organisation/institute that owns the tag
- Study
  - Stored in the EMBASIC PS variable `_STUDY$`
  - Helps sort the tag into groups based on what sort of study they have been used for, e.g. "British Gulls 4"

6.4. Script Lock

The script may be locked in order to stop a base station from overwriting it over the radio. This is mostly useful for testing, as leaving the script locked during deployment means any potential script changes will never have the opportunity to be overwritten.

To lock or unlock the script on a tag, use the `_SCRIPTLOCK` command on the EMBASIC command prompt or inside a script. This value is stored in the flash and will be saved between device resets. See the Mataki-Lite EMBASIC Reference for more information on how to do this.

The script loader tool ignores this value and always overwrites the script on a tag.

6.5. Start Up Delay

In order to save battery, a start up delay can be programmed onto the tag. This causes the tag to go to sleep for the specified time, only waking up when it is time to start the script.

This could be used, for example, to set a 1 hour delay while walking up to a site where the tags will be deployed with the batteries inserted instead of having to plug the batteries into the tags at the deployment site.

It may be cancelled through the PuTTY terminal by pressing the 'C' key twice.

To set the start up delay, use the `_STDLY` command on the EMBASIC command prompt or inside a script to specify the delay time in minutes. This value is stored in the flash and will be saved between device resets. See the Mataki-Lite EMBASIC Reference for more information.

6.6. Auto-run

Auto-run enables the script to start automatically after a reset. It is very important that this is enabled on deployed tags because tags are reset after successful contact with a base station. If it was disabled, then the device would never start the script after being reset by a base station, leaving it idle until the battery was drained.

Auto-run is always enabled when a Pi-Base uploads a new script.

To set this value, use the '-r' switch when using the script loader tool.

7. Base Station Communication

When a tag contacts a base station, a lot happens in a short space of time. This section is an overview of the process. Note that we are only describing the automated contact process (`_radio = 1`).

Two radio channels are used to communicate with the base station. The actual frequencies depend on the base frequency setting but the channel numbers are the same and are used in the following description. The basic idea is that tags contact the base station on one channel then switch to a different channel to upload their logs. During this process, other tags which try to contact the base station cannot interfere with the first tag's data transfer.

Messages can be broadcast (sent to all tags) or sent to a specific tag (its node ID). In the description below, we explicitly state when a message is broadcast, other messages are sent to a specific tag.

Initially the tag's radio is off. When the script sets `_radio = 1` the radio chip is turned on and a heartbeat is sent out...

- The tag broadcasts a heartbeat message on channel 16. The heartbeat contains basic information about the tag (ID, type, firmware version etc) and the number of log entries it has.
- The base station uses this information to compare with its own copy of the tag's logs and decide which log entries it doesn't have. To avoid searching its log every time, the base station keeps statistics on every tag which are built from the log when the base station starts up. If the base station has all the tag's logs, it sends a reset message to the tag causing the tag to reset itself.
- If there are logs to get, the base station sends a message to the tag to switch to channel 20. Then the base station pings the tag on the new channel to check the tag got the message and has switched channels.
- The ping message contains information about the base station (ID, type, firmware version etc). When the tag receives a ping message, it sends a ping response back.
- Depending on the type and capabilities of the base station, further information may be sent with the ping response. Base stations that support script uploading will be sent details of the current script (name, size, CRC etc.). This is used by the base station to decide whether the tag has the latest script. The `_OWNER$` and `_STUDY$` information are also sent at this stage and used by advanced base stations to name the directory structure for the log data and identify maps generated by a Mataki server.

- Once a connection has been established on channel 20, the base station sends a 'keep alive' message to the tag. This indicates to the tag how long it should wait before giving up if contact is lost. On Mataki-Lite this sets the `_CONTACT` variable. Different types of base station send different values due to the way they work. For example a Mataki-Classic base station sends 10 minutes as it might spend 10 minutes searching its own log without sending a radio message, whereas a Pi-Base base station sends 10 seconds as it has enough RAM to cache the whole log.

The base station then asks for the log entries it doesn't have. This is a two step process...

- In the first step, a 'range request' message is sent from the base station to the tag. This basically has a start and end point for the logs it wants e.g. Logs 101-225. The base station asks for every log between the highest entry it has and the tag's log size reported in the heartbeat message.
- The tag then sends back a message for each missing log entry with a short delay between each message. As each log entry is received, the base station adds the missing entry to its own log and updates the log statistics. The base station may not receive all of the logs but the next step takes care of the problem.
- In the second step, the base station sends a 'gap request' to the tag. This consists of a log entry number and a bitmap of the logs after that number identifying which logs the tag needs to send (gaps in the base station's log for this tag). Unlike a range request, this can quickly patch up the holes in the log because not all the logs are requested. However, due to the maximum size of the radio messages, the bitmap is limited to about 400 logs so multiple gap requests are required to patch up holes that are a large number of logs apart.
- As in range requests, the tag sends the requested logs to the base station.
- At the end of the process, the base station evaluates the data it has received and decides whether to ask again for any log entries it hasn't received. If not, the next time this tag is seen, the base station will try again.
- Note that the base station's log isn't in the same order as the tag's log and it contains logs for all the tags it has contacted. This jumbled log is unscrambled by the log converter script on a Mataki-Classic base station. On a Pi-Base base station, the logs are sent to a Mataki server which unscrambles the data and plots separate maps on its web pages for the different tags as the logs are received.

If the tag or base station doesn't support script uploads or either end doesn't have or doesn't need/want a new script, then the base station tells the tag to reset itself. The base station switches back to channel 16 and waits for other tags.

If the tag is a Mataki-Lite and the base station supports script uploads...

- If 'Base Up' is set to 'Allowed' (see section 4.1), the current script details from the ping response are compared to the latest script on the base station and if they are different, the base station tries to upload the new script to the tag.
- A 'Script Upload Start' message is sent to the tag. This stops the current script from running. The message contains the size and CRC for the new script. The tag acknowledges receipt of the message.
- The new script is sent piece by piece with 'Script Data' messages. Included in the data is the metadata for the script (filename, modified time etc.). Each data packet is acknowledged by the tag. Any that are not acknowledged are re-sent.
- A 'Script Upload End' message is sent to the tag. This causes the tag to calculate the CRC of the data it has received and compare it to the size and CRC sent in the start message. If all is ok, the new script is written to the flash and the tag is reset. If there is a problem, the new script is ignored and the tag is reset, causing the old script to be re-used. The base station will try again at the next contact.
- If the tag goes out of range of the base station during the upload process, the tag simply resets itself and re-runs the old script. This is detected by a lack of radio packets after the upload has started.

Regardless of whether the tag got the new script, the base station tells the tag to reset itself. The base station switches back to channel 16 and waits for other tags.