GPS Logger MG USER GUIDE

Version 0.91 - March 2007
gps_logger_mg@tanundra.com

This document is a user guide for the GPS Logger MG, version 0.91. Please ensure that you read the legal notices at the end of the document.

1. BACKGROUND

Sparkfun Electronics produce the “Lassen iQ FAT16 Datalogger” device, which is a compact GPS data logger implemented with the Philips LPC2148 microcontroller, the Trimble Lassen iQ GPS module and a generic SD Card. The device is provided with basic operational firmware.

The author soon found the original firmware to be lacking and started making changes, but quickly realised that only a completely new firmware could satisfy requirements. This new firmware – henceforth referred to as “the firmware” – was born, drawing upon the original firmware only in idea and spirit.

The firmware has been tested by the author on various trips in different countries, both northern and southern hemisphere, and is now ready for release to other interested users. Further releases with new and improved features are expected.

2. DESCRIPTION

This section provides an overview of the project and its features.

2.1. Project overview

The primary objectives in the implementation of the firmware are to (a) minimise power consumption; (b) maximise testing, assurance and reliability; and (c) maximise features, options and configurability.

The firmware is provided in binary form only, without source code, until further licensing terms have been decided. The binary firmware is made available for personal royalty-free non-commercial use only: a constraint imposed by both the author and by the compiler license (Rowley CrossWorks for ARM 1.5). It has to be noted that the work is not based on or derived from the original source code, and therefore does represent a new original work for the purposes of copyright. The author has invested many hours in the work.

The author welcomes feedback, suggestions, bug reports, feature requests, and commercial queries. Donations would help prioritise feature requests and fund existing and new expenses: in particular, they would support the purchase of an upgrade to Rowley CrossWorks for ARM 1.6.

Please note that this is a hobby project and it is subject to other life priorities, so the author cannot make any guarantees about the timeliness of answers to questions, bugfixes, new features, further releases, etc.

The current plan is to produce further releases with additional bugfixes and features as identified in this document. There is no timeframe, but a release is expected within 6 months. The code may also be ported to the Sparkfun Electronics “GPS Logger v2.4” (with the US Globalsat EM-406 GPS module and other improvements).

2.2. Feature overview

There are many new features: the following is a rough categorisation.

The architectural features are largely hidden to the user, but are critical to many of the objectives.

a. **Emulation build for FreeBSD/Linux using GCC v4.2** has allowed for rapid development and more comprehensive testing of functionality. The modular software architecture has an LPC2148 Hardware Abstraction Layer (HAL) providing access to uarts, timers, SPI, LEDs, watchdog, power functions, real-time clock, etc. In the emulator, the HAL is simulated and uses POSIX/UNIX calls. The latest GCC compiler release improves verification and optimisation.

b. **Separate debug and release firmware builds** ensure a fast, efficient and optimal version for general use, but a more memory, speed and power hungry diagnostic version with extensive built-in runtime checks, unit tests and fault analysis features. If faulty behaviour or hardware is suspected, the debug firmware can be employed to perform self-tests and identify faults.

c. **Program correctness, optimisation and flexibility** is obtained by paying strict and close attention to the design and its implementation, using logic checks, compiler features and third-party tools, including:

   1. **static program correctness checking with GCC and Splint** ensures a higher degree of confidence in code correctness, and in particular, the Splint “checks” level of verification has ensured strict attention to integer sizes and conversions, removing the chance for subtle errors;
   2. **runtime program correctness checking with assertions and error code handling** in the emulator and debug firmware only, ensures that operational anomalies are extensively checked for and detected before any data or disk corruption can occur;
   3. **design, implementation and compilation optimisation** through the use of good algorithms, efficient data processing, modular and reusable design, system profiling, and high levels of compiler optimisation, provide across the board gains in reliability and efficiency;
   4. **audit and reduction in use of third-party library functions** to optimise footprint and eliminate unneeded functions and features;
   5. **highly modular software architecture** improves clarity, flexibility and the capability of changes and extensions to support future features.

d. **Extensive consideration for resource and power efficiency** pervades all aspects of the design and implementation as it is the highest objective, and achieved with the following features:

1. **immediate power-down on fault detection** whether due to software, environment or configuration ensures maximum power conservation until the fault is rectified and device restarted;
2. **use of hardware power saving features** such as power-down of unused peripheral blocks, so those that are not used, such as ADC, are never powered up and others, including SPI, uarts, real-time clocks, timers, are only powered up if and while they are needed, then down afterwards;
3. **design consideration for power saving** are employed across all other operating features.

The storage and input/output functions have received detailed attention to achieve efficiency, robustness and performance.

a. **SRAM data cache configuration up to 24K with timeout handling** allows substantial power efficiency by buffering data while the FAT code, SPI interface and SD card are in idle power save mode until a large aggregate block is ready to write. The timeout handlers can guarantee bounds on times between on writes.

b. **Highly efficient and performant FAT16 file-system module** with the following features:

   1. **suspend and resume** behaviour, at application request, which powers down the SPI block and interface, allowing the SD card to enter low-power mode, when the file system is idle in between operations;
   2. **per-file sector cache** so that multiple small sized reads and writes can be serviced from the one sector sized read or write, improving response time and, more importantly, power efficiency;
   3. **directory-entry sector cache** so that small write updates to a directory entry (such as file-size and last-access-time), or multiple directory entry reads, do not require more than one sector sized read;
   4. **FAT-cluster cache with lazy update** reduces sector sized reads for cluster allocations, and allows multiple changes (such as multiple back-to-back allocations that occur on a large block write) to be rolled into a single write update, working efficiently even with multiple files;
   5. **improved algorithms for cluster and directory searches** which reduce sector reads and data processing;
   6. **optimised processing paths for sector sized reads and writes** provide maximum efficiency for configuration reads and cached block writes;
   7. **flexible synchronous or asynchronous** updates to file directory entries after file reads or writes allow finely-detailed application level control;
   8. **multiple concurrent read or write files** support for robustness testing and future feature extension;
   9. **real-time clock based dates for directory-entry updates** for both create and access times, to improve output clarity.

c. **Fully-buffered UART receive processing under IRQ** allows the application to enter power-idle mode until woken up by IRQ, while reads on empty buffer will cause entry to power idle mode until woken by IRQ with new receive data. The

UART receive FIFO for the GPS module is configured to the maximum possible (16 bytes) which keeps the processor idle for a longer period before wake up.

The main GPS configuration and processing functions are extensive, flexible, and extensible.

a. **Text based configuration file** is user-friendly and easy to edit because of the following features:

    1. **self-descriptive entries** reduce the need to refer to the user guide when making changes;
    2. **robust parameter checking** ensures that faulty values cannot be created and do not cause faulty software behaviour;
    3. **default configuration file created if** no file exists, providing a template to start editing from.

b. **Configurable NMEA sentence processing** provides detailed control over the material delivered from the GPS module before it is processed:

    1. **NMEA sentence types** allow any combination of RMC, GGA, GLL, VTG, GSV, GSA and ZDA to be handled, although currently, the GPS module must be pre-configured to output these sentences and the firmware will strip out those not needed, otherwise the default are GGA and VTG only, but in future, the firmware will configure the GPS module itself;
    2. **NMEA sentence periods** allow update rates to be specified anywhere in seconds, minutes, hours or days: currently, the GPS module puts out sentences every 1 second, and the firmware filters out those that are not needed, but in future, the firmware will configure the GPS module itself;
    3. **NMEA checksum verification** to ensure data validity.

c. **Configurable positioning output handling** means that NMEA sentences once filtered, received, verified and decoded can be formatted and output in a variety of ways to suit requirements and achieve resource efficiencies:

    1. **output formats** as either RAW data (unaltered NMEA sentences received from the GPS module), Google Earth KML (textual csv of latitude, longitude and altitude) or configurable CSV;
    2. **tuneable output CSV format** for specification of number, type and ordering of output elements (latitude, longitude, altitude and time) and whether encoding in either text or binary format – providing up to 60% space saving and considerable increase in power efficiency;
    3. **output location** to either pass through to serial port (for capture on another computer) or write to a file (the default): this occurs after any output formatting.

d. **Configurable positioning processing** allowing the positioning data, having been extract from NMEA sentences, to be processed and functionally used:

    1. **RTC updates from NMEA ZDA sentences** make file system create and access time stamps relevant and provide more meaningful diagnostic material.

e. **Provision of tools to support output post-processing** in the platform neutral scripting language of Perl can be modified and adjusted to meet individual needs:

    1. **CSV binary output unpacker** converts binary encoded CSV output back to text encoding.

3. INSTALLATION

This section describes the packaging of the release, and the process to install and verify its contents.

3.1. Release packaging

The release is provided as a single zip file containing (a) debug and release firmware for the Sparkfun “Lassen iQ FAT16 Datalogger”, (b) debug emulator for the FreeBSD v6 ia32 platform, (c) support tools implemented as Perl scripts, and (d) userguide in Adobe PDF format. The contents can be extracted using a zip compatible utility, e.g. PowerArchiver, WinZip, unzip, etc.

```
% unzip -l gps_logger_mg-0.91
Archive:  release/gps_logger_mg-0.91.zip
  Length     Date   Time    Name
 --------    ----   ----    ----
        0  03-28-07 19:39   gps_logger_mg-0.91/
   177895  03-28-07 19:39   gps_logger_mg-0.91/firmware-
sparkfun_gpslog10-debug.hex
    81296  03-28-07 19:39   gps_logger_mg-0.91/firmware-
sparkfun_gpslog10-release.hex
   459781  03-28-07 19:39   gps_logger_mg-0.91/emulator-freebsd6_ia32
     3397  03-28-07 19:39   gps_logger_mg-0.91/csv-bin-unpack.pl
    41475  03-28-07 19:39   gps_logger_mg-0.91/examples.zip
    85912  03-28-07 19:39   gps_logger_mg-0.91/userguide.pdf
 --------                   -------
   849756                   7 files
```

3.2. Sparkfun GPS Datalogger firmware

Install the firmware with the Philips “LPC2000 Flash Utility” and the Sparkfun “LPC Serial Port Boot Loader interface” or equivalent. The release firmware should be installed as the debug firmware is for advanced fault diagnosis.

Install with the following steps. Connect the interface to the device. Switch the interface to “prog” mode and power on the device. Use the utility to “Read Device ID” to confirm communication with the LPC2148 (i.e. readable Part ID and Boot Loader ID). Select the firmware image with the utility and “Upload to Flash”. Wait until the Progress bar on the utility and the flashing LEDs on the interface have ended. Power off the device and remove the interface. Ensure that a formatted SD card is present.

Power on the device and verify that the STAT0 LED flashes periodically to indicate GPS NMEA sentences are being received from the GPS unit.

3.3. FreeBSD based emulator

Install the emulator by copying it to a FreeBSD v6 ia32 host. The emulator is statically linked so has no platform library dependencies. Invoking the binary with the '-v' option will verify that it executes successfully. For example:

```
% ./emulator-freebsd6_ia32 -v
gps_logger_mg v0.91 ("FreeBSD-6.2-STABLE-i386"): emulation mode.
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
```

3.4. Perl scripted support tools

Install the tools by copying them to any Perl environment, whether Linux, FreeBSD, Cygwin (for Windows) or otherwise. Invoking each script with the '-v' option will verify that it executes successfully. For example:

```
% ./csv-bin-unpack.pl -v
gps_logger_mg v0.91 ("freebsd"): csv unpack tool.
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
```

3.5. GPS output examples

The examples are contained within their own zip file as "examples.zip", and consist of GPS source raw material ("gps00000-raw.txt") in KML ("gps00000-kml.txt"), CSV binary ("gps00000-csv-binary.txt") and CSV text ("gps00000-csv-text.txt") formats. There is also a Google Earth KML file ("gps00000.kml").

```
% unzip -l examples.zip
Archive:  examples.zip
  Length     Date   Time    Name
 --------    ----   ----    ----
    16356  03-27-07 13:07   gps00000-csv-binary.txt
    42075  03-27-07 13:07   gps00000-csv-text.txt
    32534  03-27-07 13:07   gps00000-kml.txt
   119808  03-27-07 13:07   gps00000-raw.txt
    33099  03-27-07 13:07   gps00000.kml
 --------                   -------
   243872                   5 files
```

3.6. Adobe PDF userguide

The userguide can be read with Adobe Reader or equivalent. Presumably you have figured that out by now or wouldn't be able to read this!

4. CONFIGURATION AND OPERATION

This section describes the configuration and operation of the firmware, emulator and tools.

4.1. Sparkfun GPS Datalogger firmware

The firmware must be started with a valid FAT16 formatted SD card present. If not, then operation will abort and the STAT0/STAT1 LEDs will illuminate constantly. The debug firmware emits debugging information through the serial port at 9600bps and may be useful in tracing execution and fault analysis.

The firmware reads its configuration from the GPSCONFG.TXT file on the SD card at start-up. If the file is not present, then a default one is created, which can then be edited. The default configuration file is:

```
# gps_logger v0.91: copyright (c) 2007 gps_logger_mg@tanundra.com, all
rights reserved.
# default values generated automatically, edit as required

# name: debug - debug options
# type: string - none|trace|diag (none, output trace or diagnostics
self-test)
debug = none

# name: mode - operating mode
# type: string - file|pass (file store on card, pass through serial-
port)
mode = file

# name: format - format for NMEA GPS output data
# type: string - raw|kml|csv (raw NMEA, Google Earth KML, Comma
Separated Variables)
format = kml

# name: format_csv_content - content for CSV format output
# type: string - ... variable string lat,lon,alt,tim ...
format_csv_content = lat,lon,alt,tim

# name: format_csv_encoding - encoding for CSV format output
# type: string - text|binary (text or binary)
format_csv_encoding = text

# name: file_buffer_size - file buffer size (number of bytes of RAM to
use to cache entries before file writes)
# type: integer - 0, 64 ... 24576 bytes
file_buffer_size = 512

# name: file_buffer_timeout_normal - file buffer normal timeout (max
time before forcing file write)
# type: integer - 0 (disabled) ... [# secs, #m mins, #h hours, #d
days]
file_buffer_timeout_normal = 0

# name: file_buffer_timeout_quiet - file buffer quiet timeout (max
time with no data received before forcing file write)
# type: integer - 0 (disabled) ... [# secs, #m mins, #h hours, #d
days]
file_buffer_timeout_quiet = 0

# name: pass_serial_speed - pass output serial speed (bits per second)
# type: integer - 1200, 2400, 4800, 9600, 19200, 38400
pass_serial_speed = 9600

# name: gps_output_interval - GPS NMEA sentence output capture
interval
# type: integer - 1 ... [# secs, #m mins, #h hours, #d days]
gps_output_interval = 1

# name: gps_output_sentences - GPS NMEA sentences to be output at each
interval (comma separated)
# type: string - rmc|gga|gll|vtg|gsv|gsa|zda
gps_output_sentences = gga,zda

# name: gps_update_rtc - update the RTC from GPS (if unset, or until
first update, clock will read 1980/01/01 00:00:00)
# type: boolean
gps_update_rtc = false

# end
```

The options in more detail are:

Option	Description
debug	Controls the behaviour of the debug firmware only. With ‘none’, the default, only debug run-time assertion tests are enabled. With ‘trace’, a trace of the run-time execution is output on the serial port (9600 bps). With ‘diag’, the code level unit tests are run, with execution output on the serial port, before the firmware halts on either success or failure.
mode	Specifies the GPS processing mode. With ‘file’, the default, the GPS source is formatted and logged to a file on the SD card. With ‘pass’, the GPS source is formatted and passed through to the serial port.
format	Specifies the GPS output format type. With ‘raw’, the NMEA sentences (once verified and filtered, see below) are output without alteration. With ‘kml’, the GGA sentence is formatted as a Google Earth KML textual CSV (latitude, longitude and altitude). With ‘csv’, the GGA sentence is custom formatted and encoded according to the ‘format_csv’ options.
format_csv_content	Specifies the content of the CSV formatted output. This is a comma separated list, in any order, of one or more of the GGA variables latitude (lat), longitude (lon), altitude (alt) or time (tim).
format_csv_encoding	Specifies the encoding of the CSV formatted output. With ‘text’, each record is encoded as a single line (terminated by a CR LF) of comma separated variables. With ‘binary’, each record is bitwise encoded into a binary string and output. Binary is more compact and ultimately power efficient.
file_buffer_size	Specifies the size of the file SRAM buffer. 0 disables all

	buffering and forces synchronous writes. The default is 512 bytes, which equals a FAT16 sector size. The maximum is 24576 bytes. Larger buffer sizes improve power efficiency, but increase risk of data loss on fault or power loss.
file_buffer_timeout_normal	Specifies the maximum timeout before the file SRAM buffer is forcibly flushed. 0 disables the timeout, and default units are in seconds, but ‘m’, ‘h’ or ‘d’ can denote minutes, hours, or days, e.g. “15m”, “6h”, “1d”. This ensures that the file content is never more than a certain age, for example 15m.
file_buffer_timeout_quiet	Specifies the maximum timeout during quiet periods for the file SRAM buffer is forcibly flushed. 0 disables the timeouts, and default units are in seconds, but ‘m’, ‘h’ or ‘d’ can denote minutes, hours or days. This can be set to a low value, e.g. 5m, to ensure that when coverage is lost, e.g. because the device is brought inside.
pass_serial_speed	Specifies the serial port speed when using pass output, configurable from 1200 to 38400 bps. This only has effect in the release firmware as the debug firmware ties the rate at 9600 bps and ignores this option.
gps_output_interval	Specifies the output interval for the GPS unit. The default units are seconds and the default value is 1, but units can be specified with ‘m’, ‘h’ or ‘d’ to denote minutes, hours or days. Currently, the GPS unit remains powered up, and the firmware simply ignores all but the n’th. In future revisions, the unit’s output interval will be programmed, or it will be powered down.
gps_output_sentences	Specifies output sentences for the GPS unit. The default sentences are gga and zda, but currently the unit (unless reprogrammed) only outputs gga and vtg. This option filters all formats.
gps_update_rtc	Specifies whether the zda sentences from the GPS unit are used to update the device real time clock, and therefore, the file timestamps. The zda sentence must be specified in the gps_output_sentences option.

The LEDs provide information about the firmware’s operating state.

LED(s)	Description
POWER	Indicates that the device is powered on.
STAT0 toggle	Enabled at start of NMEA sentence reception, and disabled when a complete sentence has been read.
STAT1 toggle	Toggles between on and off for each SD card write.
STAT0 and STAT1 on	Indicates that the device has powered down due to fault, e.g. disk full, disk error, software fault, etc. Any data in the SRAM file cache not flushed to disk is lost.
STAT0 and STAT1 off	Indicates that the device has powered down successfully due to power brown-out. Any data in the SRAM file cache is cleanly flushed to disk.

In **‘pass’ mode** the formatted GPS output is written out through the serial port at the relevant speed (as configured, defaulting to 9600 bps).

In **‘file’ mode** the formatted GPS output is written to a file on the SD card. The file is only created when the GPS output first becomes valid, which is immediately for the RAW format, or when a lock occurs (and valid GGA positioning data is generated) for KML or CSV format. The file is named GPS#####.TXT where ##### is a 5-digit number increasing sequentially from 00000, e.g. GPS00004.TXT, etc. The directory is searched for the first free number. The same file is then used for all logging until the device is powered down. The ‘.TXT’ extension is used regardless of whether the output is encoded in text or binary.

In both modes, text lines are terminated with a CR/LF.

4.2. CSV binary unpack support tool

The CSV binary unpack tool command line options are seen with the ‘h’ option:

```
% ./csv-bin-unpack.pl -h
gps_logger_mg v0.91 ("freebsd"): csv unpack tool.
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
usage: ./csv-bin-unpack.pl [-v] [-h] [-d] -e <lat,lon,alt,tim>
```

The **‘-v’ and ‘-h’ options** show version and help information respectively.

The **‘-d’ option** indicates to output debug information to standard error, to aid diagnose of problems and see more verbose program execution details.

The **‘-e’ option** specifies the CSV format to unpack, as originally specified in GPSCONFG.TXT. If not specified, the tool will fail to operate.

To unpack, run the tool with the specified CSV format, with the packed CSV file as standard input, and the unpacked file as standard output. For example:

```
% ls -l gps00004.txt
-rwxr-xr-x  1 root  public  10248 Mar 20 16:34 gps00004.txt
% ./csv-bin-unpack.pl -d -e lat,lon,alt,tim \
      < gps00004.txt > gps00004.csv
gps_logger_mg v0.91 ("freebsd"): csv unpack tool.
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
elements = lat,lon,alt,tim
debug = 1
bits = 90 (byts = 12)
recs = 854
% ls -l gps00004.csv
-rw-r--r--  1 root  public  26125 Mar 20 16:34 gps00004.csv
% head -3 gps00004.csv
51.514320,-0.182911,-47,152922
51.514280,-0.182880,-47,152923
51.514270,-0.182871,-47,152924
```

```
% wc -l gps00004.csv
     854 gps00004.csv
```

In this case, the debug output shows that each compressed record consumes 12 bytes (90 bits), and there are 854 records in the file (854 x 12 = 10248 bytes, the size of the input file), and these many lines are present in the output file.

4.3. FreeBSD based emulator

The emulator command line options are seen with the '-h' option:

```
% ./emulator-freebsd6_ia32 -h
gps_logger_mg v0.91 ("FreeBSD-6.2-STABLE-i386"): emulation mode.
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
usage: ./emulator-freebsd6_ia32 [-v] [-h] [-i fat_image] [-f
gps_source] [-s timer_scale] [--cfg_<name>=<value>]
```

The **'-v' and '-h' options** show version and help information respectively.

The **'-i' option** specifies the location of the FAT16 disk image file. This is a binary image of a FAT16 file system, such as one present on an SD card. The emulator fails to operate if this image is not present. Under Linux or FreeBSD, an existing card can be copied to an image using dd, e.g. "dd if=/dev/hdb of=sdfat.img bs=512".

The **'-f' option** specifies the location of a GPS source file, which must contain raw NMEA sentences, such as one created using the firmware itself when in RAW mode. If not specified, then a randomly generated internal GPS source is used: currently generating only runs of locked or unlocked GGA sentences.

The **'-s' option** specifies the scaling of the emulator's clock in microseconds. If not specified, the clock runs without scaling: at full speed. A value of 1000000 corresponds to one real second, and the emulator will run as if operating in real time, i.e. where the GPS source is sampled at 1 second.

The **'--cfg_<name>=<value>' option** overrides items specified in GPSCONFG.TXT. For example, "--cfg_format=kml" or "--cfg_format_csv_encoding=binary". The self diagnostics can be run using "--cfg_debug=diag".

The purpose of the emulator is to aid development, but it can be used to convert GPS source files in RAW format to other formats. It is not for general purpose use and not further described here.

5. OPERATING ESTIMATES

This section provides some estimates on data volumes, log sizes, write frequencies and power consumption.

When logging in KML mode, each text record can consume between 22 bytes and 30 bytes, say 28 bytes average. The full 24K SRAM buffer, logging at one second intervals, fills in 878 seconds, i.e. 15 minutes. The 512MB SD card, with perhaps a maximum file size of 508MB, could store 5284 hours of records, i.e. 220 days.

When logging in CSV mode, using binary format, the same KML data (latitude, longitude, altitude) is a constant 10 bytes. The 24K SRAM buffer, logging at one second intervals, fills in 2458 seconds, i.e. 41 minutes. The 512MB SD card, with a maximum file size of 508MB, could store 14796 hours of records, i.e. 616 days.

When the SRAM buffer fills, in either case, whether 15 minutes or 41 minutes, the SD card and SPI block are powered up, and 48 disk blocks are written, requiring 48 file sector writes, 1 directory sector update, and less than 10 FAT sector updates. The SD card and SPI block are then powered down and the SRAM buffer is reset.

The primary purpose of the SRAM buffer and the binary logging formats is not to enable larger amounts of data to be logged, but to drastically reduce the overall amount of time that the SD card and SPI block need to be powered up, and more importantly, the number of SD card block writes that need to be made. These block writes consist of power-expensive updates to flash memory and apart from the GPS mode itself, are the largest consumer of power in the device.

Using the above examples, over a period of 14 days, perhaps on an extended road, hike or boat trip, KML text mode would make 1344 writes of ~50 sectors, but the equivalent CSV binary mode would make 491 writes, which is 63% less. Further improvements in the binary format will improve this further, probably up to 75-80%.
TODO need to include power estimates

LEGAL NOTICES

For all correspondence, send email to gps_logger_mg@tanundra.com.