GPS Logger MG

USER GUIDE

Version 0.94 – June 2007

gps_logger_mg@tanundra.com

Please ensure that you read the legal notices at the end of the document

1. BACKGROUND

"GPS Logger MG" is an independently developed firmware for use with the Spark Fun Electronics range of GPS Dataloggers. Neither the author nor the project is associated with Spark Fun Electronics. The author built the firmware from scratch to support features, and address limitations, in the original Spark Fun firmware. The new firmware is not a derived work and so has its own copyright and license.

The firmware is currently compatible with the Spark Fun **"Lassen iQ FAT16 Datalogger"**, using the Trimble Lassen iQ GPS module, and the Spark Fun **"GPS Logger v2.4"**, using the US Globalsat EM-406 GPS module. Both have a Philips LPC2138/2148 ARM microprocessor and use an SD card for storage. They are powered from a 6v supply, typically a battery pack.

The firmware is also provided with Linux and FreeBSD based emulators, command line utilities, various example files, and this documentation. The release has been thoroughly tested and also used by the author on trips in different countries (both northern and southern hemisphere). It is regularly maintained as a hobby project.

There is an project blog for further information: http://gpslogger.blogspot.com.

2. DESCRIPTION

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

2.1. Project overview

The firmware was developed to provide features not available, nor easily created, in the factory shipped firmware (authored by Spark Fun Electronics). For this reason, the firmware in GPS Logger MG was created entirely from scratch with a new more modular, efficient and testable architecture.

The primary objectives in developing the firmware have been, in this order, to (a) minimise power consumption; (b) maximise testing, assurance and reliability; and (c) maximise features, options and configurability. At the current stage of development, these have been realised to certain degrees, but work to go further is ongoing.

The firmware is provided in binary form only, without source code (except for select utilities), until further licensing terms have been decided. The binary firmware is available strictly for personal royalty-free and non-commercial use only: this is a constraint imposed by the author and by the compiler license (Rowley CrossWorks for ARM 1.5 under a personal license).

The author has invested many hours in developing the firmware and welcomes feedback, suggestions, bug reports, feature requests, and commercial queries. Donations would help prioritise feature requests and fund existing and new expenses: the author would like to 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 at about every 6 months. The project blog provides regular updates and information on the projects development and schedule.

The current roadmap outlook is:

- **Release v0.94 (this release):** support V1.0 and V2.4 hardware; provide output compression; various code level fixes and optimisations.
- **Release v0.96 (Q3/Q4 07):** rework and extend the GPS data handling with GPS module programming, SparkFun firmware features, utility for standalone GPS format conversion, etc; minor code and performance optimisations.

- **Release v0.98 (Q4 07, Q1 08):** serial download and command line interface; extensive focus on code and power performance and optimisations. Perhaps other features TBD.
- **Release v1.00 (Q2 08):** a milestone release once all bugs and outstanding issues are resolved. Perhaps other features TBD.

2.2. Feature overview

The features are roughly categorised, and highlighted if they are new to this release.

2.2.1. Structural (architectural and environmental) features

These are largely hidden to the user, but are critical to success of the firmware. In some cases, they are cross-cutting features that pervade other functional features.

a. **Emulators for FreeBSD release 6 and Linux kernel 2.6** for rapid development and testing, using a modular LPC21xx hardware abstraction layer, built on POSIX threads and UNIX libraries. Built with GCC V4.3 and high levels of optimisation and warnings. Distributed as statically linked ELF binary. IMPROVED IN V0.94

b. **Debug and Release builds produced separately** for a fast, efficient and optimal version for general use, but a memory, speed and power hungry diagnostic version with extension built-in runtime checks, self-tests and fault analysis features.

c. **Program correctness, optimisation and flexibility** 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 correctness checking with GCC and Splint**, especially the Splint “checks” level of verification that pays close attention to integer sizes and conversions, reducing chance of subtle errors; IMPROVED IN V0.94
    2. **runtime correctness checking with assertions and error code handling** in the emulator and debug firmware ensuring that operational anomalies are extensively checked for and detected before any chance of data corruption;
    3. **design, implementation and compilation optimisation** through good algorithms, efficient data processing, modular and reusable design, system profiling, and high levels of compiler optimisation;
    4. **analysis of and reduction in the use of third-party library functions** to optimise footprint and eliminate unnecessary baggage;
    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** pervading all aspects of the design and implementation as the highest objective, through:

    1. **immediate power-down on fault detection** whether due to software, environment or configuration ensures 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 (e.g. ADC), and demand power-up and power-down of other non-essential blocks (e.g. SPI, uarts, real-time clocks, timers);

3. **design considerations for power saving** are employed across all other operating features.

e. **Automated build and test framework** to build executables, run code checks, run self-tests, run tests against sample sets, generate output formats and statistics, build release packages, and manage the version control repository. NEW IN V0.94

2.2.2. Subsystem (execution, storage and I/O) features

These infrastructure modules have received detailed attention to achieve efficiency, robustness and performance.

a. **SRAM data cache configurable up to 24KiB with timeout handling** for 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. **Generic LZARI compression layer** to provide further decreases input data before delivery into the SRAM data cache, which enables greater power efficiency and storage capacities by reducing data write amounts and sizes. NEW IN V0.94

c. **Highly efficient and performant FAT16 file-system module** with:

   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; IMPROVED IN V0.94
   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.

d. **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.

2.2.3. Application (GPS configuration and processing) features

The application modules have received detailed attention to achieve configurability, extensibility and flexibility.

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 positioning data input handling**:
    1. **capture periodicity** allows positioning data to be captured at any interval, whether seconds, minutes, hours or days, defaulting to 1 second.

c. **Configurable positioning data output formats**:
    1. **raw output format** to process unmodified content as received from the GPS module; NEW IN V0.94
    2. **NMEA output format** to filter (by any sentence type), verify (by checksum and structural content) and process conformant sentences;
    3. **KML output format** containing longitude, latitude and altitude for use with popular geo-tools including Google Earth;
    4. **CSV output format** for configurable specification of positioning elements (longitude, latitude, altitude, time and date), stored as text or a compressed binary format for considerable space saving. IMPROVED IN V0.94

d. **Configurable positioning data output modes**:
    1. **pass mode** to send output format through serial port in real-time, at a configurable baud rate; NEW IN V0.94
    2. **file mode** to store output format to a file, whether compressed and/or cached, as a FAT16 file on the SD card.

e. **Configurable positioning data runtime processing**:
    1. **RTC updates from NMEA ZDA/RMC sentences** make file system create and access timestamps relevant and enable meaningful diagnostic data.

f. **Utilities for positioning data output post-processing** in platform neutral source code, including:
    1. **Generic LZARI unpack utility** provided in ‘C’ source-code to unpack LZARI compressed data; NEW IN V0.94
    2. **CSV-binary format unpack utility** provided in Perl source-code to unpack CSV binary encoding back to text encoding. IMPROVED IN V0.94

3. INSTALLATION

This section describes the release packaging, installation and verification.

3.1. Packaging

The release is provided as a single zip file having (a) debug & release firmware for both Spark Fun “Lassen iQ FAT16 Datalogger” and “GPS Logger v2.4” devices, (b) debug emulators for both “FreeBSD release 6” and “Linux kernel 2.6” IA-32 hosts, (c) support utilities provided in source code form, (d) example config and output log files, (e) userguide in Adobe PDF format, and (f) a release README and LICENSE.

Extract the contents using a zip compatible utility, e.g. WinZip, unzip, etc.

Please read the README and LICENSE as they have important information.

```
% unzip -l gps_logger_mg-0.94
Archive:  gps_logger_mg-0.94.zip
  Length     Date   Time    Name
 --------    ----   ----    ----
        0  06-26-07 22:26   gps_logger_mg-0.94/
   493336  06-26-07 22:26   gps_logger_mg-0.94/emulator-freebsd_6.2-
stable_i386
  1005313  06-26-07 22:26   gps_logger_mg-0.94/emulator-linux_2.6.18-4-
686_i686
   243723  06-26-07 22:26   gps_logger_mg-0.94/firmware-
sparkfun_gpslog10-debug.hex
   129205  06-26-07 22:26   gps_logger_mg-0.94/firmware-
sparkfun_gpslog10-release.hex
   245036  06-26-07 22:26   gps_logger_mg-0.94/firmware-
sparkfun_gpslog24-debug.hex
   130825  06-26-07 22:26   gps_logger_mg-0.94/firmware-
sparkfun_gpslog24-release.hex
    14606  06-26-07 22:26   gps_logger_mg-0.94/utility-lzari-unpack.c
     5106  06-26-07 22:26   gps_logger_mg-0.94/utility-csvbin-unpack.pl
   129047  06-26-07 22:26   gps_logger_mg-0.94/examples.zip
   112554  06-26-07 22:26   gps_logger_mg-0.94/userguide.pdf
     1433  06-26-07 22:26   gps_logger_mg-0.94/README
      422  06-26-07 22:26   gps_logger_mg-0.94/LICENSE
 --------                   -------
  2510606                   13 files
```

3.2. Installation of firmware on Spark Fun GPS Datalogger devices

Install the firmware with the Philips “LPC2000 Flash Utility” and the Sparkfun “LPC Serial Port Boot Loader interface” or equivalents. The release firmware is preferable as the debug firmware is only for advanced fault diagnosis. Use the following steps:

a. Connect the boot loader interface to the data logger device.
b. Switch the interface to “prog” mode and power on the device.
c. Use the flash utility to “Read Device ID” to confirm communication with the LPC2138/2148 (i.e. the Part ID and Boot Loader ID are readable).
d. Select the firmware image with the utility and “Upload to Flash”.

e. Observe that uploading is occurring with the utility Progress bar moving forward and the interface LEDs flashing.
f. Wait until progress has completed and the LEDs have stopped flashing.
g. Power off the device and remove the interface.
h. Insert a formatted and empty SD card into the device.
i. Power on the device and verify that the status LED flashes periodically to indicate GPS NMEA sentences are being received from the GPS unit.
j. Power off the device and remove the SD card.
k. Inspect the contents of the SD card to verify that a default config file was created.

3.3. Installation of emulators for FreeBSD 6 or Linux 2.6 IA-32 hosts

Install the relevant emulator by copying it to either a “FreeBSD release 6” or a “Linux kernel 2.6” IA-32 host. The emulators are statically linked and have no host library dependencies. Although they were built for specific system and kernel releases, they should work on related releases (e.g. any FreeBSD release 6, not just 6.2). Invoke the emulator with the ‘-h’ option to verify that it executes successfully.

```
% uname -a
FreeBSD clare 6.2-STABLE FreeBSD 6.2-STABLE #3: Tue Mar 27 11:08:51 GMT
2007     root@clare:/opt/build/src/sys/CLARE  i386
% ./emulator-freebsd_6.2-stable_i386 -h
gps_logger_mg v0.94: firmware emulator (freebsd_6.2-stable_i386).
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
usage: ./emulator-freebsd_6.2-stable_i386 [-v] [-h] [-i fat_image|-d
file_dir] [-f gps_source] [-s timer_scale] [--cfg_<name>=<value>]
```

```
% uname -a
Linux debian 2.6.18-4-686 #1 SMP Mon Mar 26 17:17:36 UTC 2007 i686
GNU/Linux
% ./emulator-linux_2.6.18-4-686_i686 -h
gps_logger_mg v0.94: firmware emulator (linux_2.6.18-4-686_i686).
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
usage: ./emulator-linux_2.6.18-4-686_i686 [-v] [-h] [-i fat_image|-d
file_dir] [-f gps_source] [-s timer_scale] [--cfg_<name>=<value>]
```

3.4. Installation of utilities for compressed output unpacking

Install the Perl based csvbin unpack utility by copying it to a Perl environment: Linux, FreeBSD, Cygwin (for Windows) or otherwise. Invoke the utility with the ‘-h’ option to verify that it executes successfully. If it does not, you may need to install additional Perl modules, e.g. POSIX and Getopt::Long, refer to ‘cpan’.

```
% ./utility-csvbin-unpack.pl -h
gps_logger_mg v0.94: csv unpack tool.
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
usage: ./utility-csvbin-unpack.pl [-v] [-n] [-h] [-d] -e
<lon,lat,alt,tim,dat>
```

Install the ‘C’ based lzari unpack utility by compiling it with a standard ‘C’ compiler (e.g. gcc) and copying it to your local environment. Invoke the utility with no input to verify that it executes successfully.

```
% gcc utility-lzari-unpack.c -o utility-lzari-unpack
% ./utility-lzari-unpack < /dev/null
gps_logger_mg v0.94: lzari pack tool.
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
(256/96/2; 3708) :: read = 0 bytes, write = 0 bytes :: ratio = nan
```

3.5. Installation of example config and output files

The example files are contained within their own zip file named “examples.zip”, and consist of an example config file (as “gpsconfg.txt”), and two sets of output files (with “gps00000” and “gps00001” prefixes) with corresponding sample KML files (for direct use with Google Earth). The benefits of binary or generic compression can already be seen: for example, 123KiB of NMEA GGA sentences is reduced to 6KiB using KML output mode with compression, or 32KiB without compression.

```
% unzip -l examples.zip
Archive:  examples.zip
  Length     Date   Time    Name
 --------    ----   ----    ----
      458  06-25-07 20:44   gps00000-csv-binary-compress.txt
      513  06-25-07 20:44   gps00000-csv-binary.txt
     1055  06-25-07 20:44   gps00000-csv-text-compress.txt
     4864  06-25-07 20:44   gps00000-csv-text.txt
      761  06-25-07 20:44   gps00000-kml-compress.txt
     3800  06-25-07 20:44   gps00000-kml.txt
     2515  06-25-07 20:44   gps00000-nmea-compress.txt
    57704  06-25-07 20:43   gps00000-nmea.txt
     5502  06-25-07 20:43   gps00000-raw-compress.txt
    97794  06-25-07 20:43   gps00000-raw.txt
     4359  06-25-07 20:44   gps00000.kml
     3138  06-25-07 20:44   gps00001-csv-binary-compress.txt
     4455  06-25-07 20:44   gps00001-csv-binary.txt
     8972  06-25-07 20:44   gps00001-csv-text-compress.txt
    42075  06-25-07 20:44   gps00001-csv-text.txt
     6587  06-25-07 20:44   gps00001-kml-compress.txt
    32534  06-25-07 20:44   gps00001-kml.txt
    14945  06-25-07 20:44   gps00001-nmea-compress.txt
   125944  06-25-07 20:44   gps00001-nmea.txt
    14949  06-25-07 20:44   gps00001-raw-compress.txt
   125954  06-25-07 20:44   gps00001-raw.txt
    33093  06-25-07 20:44   gps00001.kml
     2254  06-25-07 20:44   gpsconfg.txt
 --------                   -------
   594225                   23 files
```

3.6. Installation of userguide in Adobe PDF format

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 configuration and operation of firmware, emulator and tools.

4.1. Firmware on Spark Fun GPS Datalogger devices

4.1.1. Start-up

The firmware must be started with a valid FAT16 formatted SD card present. If a valid card is not present, then execution will abort and the LEDs will report an error condition (see below). The debug firmware emits debugging information to the serial port at 9600 bps which may useful in tracing execution and fault analysis.

4.1.2. Configuration

The firmware starts up and reads its configuration from the GPSCONFG.TXT file on the SD card. If the file is not present then a default one is created, which contains:

```
# gps_logger_mg v0.94: firmware configuration file.
# default values generated automatically, edit as required.

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

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

# 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|nmea|kml|csv (raw data, NMEA data, Google Earth
KML, Comma Separated Variables)
format = kml

# name: format_nmea_sentences - GPS NMEA sentences to select with NMEA
format (comma separated)
# type: string - rmc|gga|gll|vtg|gsv|gsa|zda (supported NMEA sentences)
format_nmea_sentences = gga,zda

# name: format_csv_content - content of each entry for CSV format
output
# type: string - lon|lat|alt|tim|dat (refer to documentation)
format_csv_content = lon,lat,alt,tim

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

# name: compress - perform compression (lzari based) on output data
(reduces buffer size by 8K)
# type: boolean
compress = false
```

```
# 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
# type: integer - 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
(bits per second, 8N1)
pass_serial_speed = 4800

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

# end
```

The options in the configuration file are as follows:

Option	Description
debug	Relevant for the debug firmware only. 'none' only enables run-time assertion tests. 'trace', the default, also turns on run-time tracing log, output to the serial port at 9600 bps. 'diag' forces the code-level unit-test suite to be run, with assertions enable and output to the serial port, after which the firmware will halt on a success or failure condition.
interval	Specifies the recording interval for GPS positioning data. The default units are seconds and the default value is 1. Units can be specified with 'm', 'h' or 'd' to denote minutes, hours or days. Currently, the GPS unit is remains powered up, and the firmware simply ignores all but the 'n'th record, but this will be optimised in future releases.
mode	Specifies the operating mode. 'file', the default, causes GPS positioning data to be formatted and output to a file on the SD card. 'pass', also processes and formats data, but only outputs it to the serial port.
format	Specifies the output format for GPS positioning data. 'raw' performs no formatting and outputs data as-received. 'nmea' accepts only NMEA sentences (filtered and checksum verified). 'kml', the default, turns each NMEA GGA sentences into a Google Earth KML textual CSV (longitude, latitude and altitude). 'csv' turns either GGA or RMC sentences into a custom CSV.

format_nmea_sentences	Specifies which NMEA sentences to accept with the 'nmea' format, and all others are ignored. Sentences currently supported are RMC, GGA, GLL, VTG, GSV, GSA and ZDA. The default is GGA and ZDA.
format_csv_content	Specifies the content of each output CSV record with the 'csv' format. This is a comma separated list, in any order, of one or more of the GGA/RMC variables longitude (lon), latitude (lat), altitude (alt), time (tim) or date (dat). The default is 'lon,lat,alt,tim'.
format_csv_encoding	Specifies the encoding of the output CSV record with the 'csv' format. 'text', the default, encodes each record as a single line (terminated by CR/LF) of comma separated variables. 'binary' encodes each record as a compressed binary string, which is space and power efficient.
compress	Specifies compression for the output data. This may be true or false, the default. The compression is based on LZARI and requires 8KiB of internal RAM, reducing the maximum file buffer size from 24KiB to 16KiB.
file_buffer_size	Specifies the size of the SRAM buffer for the output file in 'file' mode. 0 disables all buffering and forces synchronous writes, but the default is 512 bytes, which equals a FAT16 sector size, and the maximum is 24576 bytes (or 16384 if compression is enabled). Larger buffer sizes improve power efficiency, but increase risk of loss on fault or power loss.
file_buffer_timeout_normal	Specifies the maximum timeout before the SRAM buffer is forcibly flushed in 'file' mode. 0, the default, disables the timeout, and units are in seconds unless specified with 's', 'm', 'h' or 'd' for seconds, minutes, hours, or days, e.g. "15m", "6h", "1d". Use to provide a guarantee that the file content is updated with a specified time period, e.g. 15m.
file_buffer_timeout_quiet	Specifies the maximum timeout during quiet periods before the SRAM buffer is forcibly flushed in 'file' mode. 0, the default, disables the timeout, and units are in seconds unless specified with 's', 'm', 'h' or 'd' for seconds, minutes, hours or days. Use to ensure that if coverage is lost, e.g. because the device is brought inside, the file content is flushed within a given time period, e.g. 2m.
pass_serial_speed	Specifies the serial port speed in bits per second for output in 'pass' mode. This may be either 1200, 2400, 4800, 9600, 19200, 38400, 57600 or 115200. The default is 4800. Note that this only has effect in the release firmware, as the debug firmware clamps output to 9600.
update_rtc	Specifies whether the microprocessor real-time-clock (rtc) should be updated from the GPS positioning data. This only affects file output timestamps. Either the ZDA or RMC sentence must be provided by the GPS unit.

It is allowable for configuration entries to not be present, in which case the internal defaults are used, however it is not allowable for entries to be present but invalid,

whether invalid because of syntax (e.g. “wrue” rather than “true”) or value (e.g. a file buffer size of 32KiB). If an option is specified for a mode that is not being used (e.g. the ‘pass_serial_speed’ option even though the current ‘mode’ is set to ‘file’), then it must still be valid. When invalid entries are detected, execution will halt with an error.

4.1.3. Operating status via. LEDs

The LEDs provide information about the firmware’s operating state, but differ slightly between the “Lassen iQ FAT16 Datalogger”, which has three single-state red coloured LEDs, and the “GPS Logger v2.4”, which has a single tri-state (blue, green and red cololured) LED. Note that the EM-406 on the “GPS Logger v2.4” has a red power LED that illuminates whenever it is powered-up.

LED states for firmware v1.0 (Lassen iQ FAT16 Datalogger)

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. Indicates that data is being received successfully.
STAT1 toggle	Toggles between on and off for each SD card write (e.g. if the file cache is set to 24KiB, then the LED toggles for each 24KiB block that is written). Indicates that data is being written successfully.
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.

LED states for firmware v2.4 (GPS Logger v2.4)

LED(s)	Description
RED-COLOUR toggle	Enabled at start of NMEA sentence reception, and disabled when a complete sentence has been read. Indicates that data is being received successfully.
GREEN-COLOUR toggle	Toggles between on and off for each SD card write (e.g. if the file cache is set to 24KiB, then the LED toggles for each 24KiB block that is written). Indicates that data is being written successfully.
BLUE-COLOUR toggle	Not currently used.
all (RED-, GREEN- & BLUE- COLOUR) 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.
all (RED-, GREEN- & BLUE- COLOUR) off	Indicates that the device has powered down successfully due to stop button pressed or power brown-out. Any data in the SRAM file cache is cleanly flushed to disk.

4.1.4. Operating in ‘pass’ mode

In **‘pass’ mode** the formatted GPS output is written out through the serial port at the configured speed (defaulting to 4800 bps). Text lines are terminated with a CR/LF (DOS format line-ending).

This mode has been verified to work with the PC applications listed below. The default serial speed of 4800 bps should be used for full compatibility. The output format should be set to **‘raw’** or preferably **‘nmea’** which provides better compatibility for applications that may be unhappy with any GPS module proprietary output (as is generated by the EM-406). The test environment consisted of Windows XP Pro SP2, on a Dell Latitude D620, with Intel T2600 CPU at 2.16GHz, and 1.00 GiB of RAM.

There is currently no input handling, but future releases may support a command interface.

Mapping applications

Application	Description
Microsoft AutoRoute 2007 with GPS Locator	Popular road trip planning and real-time directions.
Google Earth Plus	Geo-imaging with real-time GPS, configured for Magellan/Serial mode with NMEA.
GooPs	Provides real-time GPS for basic Google Earth (non Plus).
BUNGEE	Provides LIVE GPS tracking for basic Google Earth (non Plus).
Earth Bridge	Provides a GPS bridge for basic Google Earth (non Plus). Note: does not work with default v1.0 kit as the iQ does not emit RMC sentences. Fine with v2.4 where the EM-406 does!
MeHere	Networked GPS tracker working with both Google Maps and basic Google Earth (non Plus).
TopoFusion	GPS mapping software.
GPS TrackMaker	Provides real-time navigation.
Memory-map V5 European Edition	Route-planning and mapping software.

Wireless network applications

Application	Description
NetStumbler	Wireless network analyser tool.
WiFi Hopper	Wireless network discovery and site survey.
WirelessMon	802.11 wireless monitoring tool.

Diagnostic utitilies

Application	Description
GPSDiag	Performs simple interpretation and presentation of GPS information from NMEA strings.
GPS Utility	Manage, manipulate and map GPS information.
GPS Express	Variety of reception, interpretation and mapping functions.
VisualGPS	Command monitor and graphical view of NMEA data.
NMEA Sentence Logger	File and network logging of NMEA data.

Timekeeping applications

Application	Description
GPS Time and Test	Provides clock-synchronisation. Note: fails to work with default v1.0 kit due to lack of ZDA or RMC sentences. Fine with v2.4 where the EM-406 does.

4.1.5. Operating in 'file' mode

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 files are 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, and once the file with that number is opened, it is 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 or is compressed. Text lines are terminated with a CR/LF (DOS format line-ending).

It is worth improving power-efficiency by choosing the most optimal recording format and enabling compression. Unfortunately v1.0 hardware has no stop-button, so when logging is to finish, it is advisable to note that disk led state (STAT1), and wait until it toggles indicating that the last valid data was flushed to disk. Setting the quiet timeout reduces the flush delay if the device is not receiving any more valid data, for example when it has been brought indoors. This problem does not existing for v2.4 hardware, in which case the stop-button causes an orderly shutdown.

4.1.6. Using debug features in the debug firmware

The debug firmware can be loaded to assist with fault finding. It contains substantial run-time assertion checking, built-in diagnostic trace output and built-in self tests. By default, according to the debug configuration option, the **'trace'** output will emit to the serial port at 9600 bps: the speed cannot be changed and is not affected by other settings which may set the serial port speed (they are ignored). If the **'diag'** option is specified, then as the firmware starts up, it will run through its internal suite of self-tests, with pass/fail criteria emitted to the serial port. When the test is complete, the firmware will shutdown.

4.2. Support utility for CSV-binary unpacking

The CSV-binary unpack utility invocation options are seen with the '-h' option:

```
% ./utility-csvbin-unpack.pl -h
gps_logger_mg v0.94: csv unpack tool.
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
usage: ./utility-csvbin-unpack.pl [-v] [-h] [-d] [-n] -e
<lon,lat,alt,tim,dat>
```

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 **‘-n’ option** specifies that both CR/LF line endings (DOS format) should be used, rather than the default LF line ending (UNIX format).

The **‘-e’ option** specifies the CSV format (i.e. the elements) to be unpacked, as originally specified by the ‘format_csv_content’ option. This option is mandatory.

The standard input and output are used for file input and output respectively.

To unpack a file, run the utility with the specified CSV format, having the packed file on standard input, and the unpacked file on standard output. For example:

```
% ls -l gps00000-csv-binary.txt
-rw-r-xr-x   513 Jun 25 20:44 gps00000-csv-binary.txt
% ./utility-csvbin-unpack.pl -n -e lon,lat,alt,tim \
        < gps00000-csv-binary.txt > gps00000-csv-binary.unpacked
gps_logger_mg v0.94: csv unpack tool.
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
% ls -l gps00000-csv-binary.unpacked gps00000-csv-text.txt
-rw-r--r--  4864 Jun 25 21:02 gps00000-csv-binary.unpacked
-rw-r-xr-x  4864 Jun 25 20:44 gps00000-csv-text.txt
% cmp gps00000-csv-binary.unpacked gps00000-csv-text.txt
% wc -l gps00000-csv-binary.unpacked
     152 gps00000-csv-binary.unpacked
% head -3 gps00000-csv-binary.unpacked
-0.179930,51.513215,-47,172526
-0.179665,51.513276,-46,172534
-0.179698,51.513265,-47,172535
```

In this case, 152 records in CSV-binary format were packed at 513 bytes and unpacked to CSV-text format at 4864 bytes, a packed ratio of about 10:1.

4.3. Support utility for LZARI generic unpacking

The LZARI generic unpack utility has no invocation options.

The standard input and output are used for file input and output respectively.

To unpack a file, run the utility, having the packed file on standard input, and the unpacked file on standard output. For example:

```
% ls -l gps00000-kml-compress.txt
-rw-r-xr-x   761 Jun 25 20:44 gps00000-kml-compress.txt
% ./utility-lzari-unpack \
        < gps00000-kml-compress.txt > gps00000-kml-compress.unpacked
gps_logger_mg v0.94: lzari pack tool.
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
(256/96/2; 3708) :: read = 761 bytes, write = 3800 bytes :: ratio =
4.993430
% ls -l gps00000-kml-compress.unpacked gps00000-kml.txt
-rw-r--r--  3800 Jun 25 21:06 gps00000-kml-compress.unpacked
-rw-r-xr-x  3800 Jun 25 20:44 gps00000-kml.txt
% cmp gps00000-kml-compress.unpacked gps00000-kml.txt
% wc -l gps00000-kml-compress.unpacked
     152 gps00000-kml-compress.unpacked
% head -3 gps00000-kml-compress.unpacked
-0.179930,51.513215,-47
-0.179665,51.513276,-46
-0.179698,51.513265,-47
```

In this case, 152 records in KML format were packed at 761 bytes and unpacked to 3800 bytes. The utility indicates that the pack ratio was about 5:1.

4.4. Emulators for FreeBSD 6 or Linux 2.6 IA-32 hosts

The emulator invocation options are seen with the ‘-h’ option:

```
% ./emulator-freebsd_6.2-stable_i386 -h
gps_logger_mg v0.94: firmware emulator (freebsd_6.2-stable_i386).
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
usage: ./emulator-freebsd_6.2-stable_i386 [-v] [-h] [-i fat_image|-d
file_dir] [-f gps_source] [-s timer_scale] [--] [--cfg_<name>=<value>]
```

```
% ./emulator-linux_2.6.18-4-686_i686 -h
gps_logger_mg v0.94: firmware emulator (linux_2.6.18-4-686_i686).
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
usage: ./emulator-linux_2.6.18-4-686_i686 [-v] [-h] [-i fat_image|-d
file_dir] [-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. Under Linux or FreeBSD, an existing card can be copied to an image using dd, e.g. “dd if=/dev/hdx of=sdfat.img bs=512”.

The **‘-d’ option** specifies the location of the file system directory. This allows the emulator to read and write files to the native file system rather than a FAT16 disk image.

Note that it is mandatory to specify either an image or a directory.

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 **‘--' option** is optional, but must be used if any subsequent ‘--' based options are being used (as follows).

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

The emulator is primarily used for development purposes but it can convert formats as in the following. In subsequent releases, this conversion code will be factored out into its own standalone utility.

```
% ./emulator-freebsd_6.2-stable_i386 -d . -f gps00000-raw.txt \
        -- --cfg_format=kml > out.log
gps_logger_mg v0.94: firmware emulator (freebsd_6.2-stable_i386).
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.

DOWN
% ls -l GPS00000.TXT
-rw-r-xr-x  3800 Jun 25 21:10 GPS00000.TXT
% ls -l gps00000-kml.txt
-rw-r-xr-x  3800 Jun 25 20:44 gps00000-kml.txt
% cmp GPS00000.TXT gps00000-kml.txt
% head out.log
lpc21XX_init
gps_logger_mg v0.94: system firmware (hardware v2.4).
copyright (c) 2007 gps_logger_mg@tanundra.com. all rights reserved.
app_init
cfg_init: default config; filename = GPSCONFG.TXT, length = 2254
cfg_item_decode: parsing tokens: <format>, <kml>
cfg_item_decode_string: got 'kml'
gps_init
gps_mode = file
cachefile_init: length=512, timeout_n=0, timeout_q=0
% tail out.log
--GPS--> $GPVTG,,,,,,,,,N*30
 <----
--GPS--> $GPGG
 <----
gps_nmea_decode: checksum verification failed
--GPS-->
 <----
app_term
gps_term
lpc21XX_term
```

5. OPERATING ESTIMATES

This section provides some estimates on data sizes, write frequencies and power consumption. These are by no means exact, and actual results depend you're your specific circumstances, so you should use the following as guidance, but do your own evaluations.

5.1. Data and read/write sizes for particular output formats

In the following cases, GPS sample data was collected from 19 unique locations on the internet, and represents test logs, personal data logger recordings, flights of test equipment, boat trips, car journeys, etc. In total, the 550 files contain 700MiB of raw data, which reduces down to 329MiB of valid NMEA sentences.

Case 1: Large aggregate sample

In the first case, the GPS sample files were collated into a single large file and then used to generate all output format types. The figures clearly show the benefits of the KML or CSV formats, especially CSV format with binary encoding. In all cases, generic LZARI compression further reduces output size, but this comes at an expense of processing power and onboard RAM. Note that 24KiB file buffer was used for the uncompressed results and a 16KiB file buffer for the compressed results (the remaining 8KiB was used by the LZARI packer).

format	Size (uncompressed / compressed)	sector reads (uncompressed / compressed)	sector writes (uncompressed / compressed)
NMEA (GGA sentences)	189.2MiB / 26.8MiB (14.1%)	154 / 31 (20.1%)	411794 / 60067 (14.6%)
KML	59.5MiB / 11.4MiB (19.2%)	55 / 19 (35.5%)	129482 / 25446 (19.7%)
CSV-text (lon,lat,alt)	59.5MiB / 11.4MiB (19.2%)	55 / 19 (35.5%)	129482 / 25446 (19.7%)
CSV-binary (lon,lat,alt)	7.9MiB / 5.8MiB (73.4%)	16 / 16 (100%)	17178 / 13038 (75.9%)

The most effective trade-off is CSV-binary without compression, which provides compact output but avoids the memory and processing overhead of the LZARI packer. The results also reveal that KML output is 31.4% size of NMEA, and CSV-binary is 13.2% size of KML, or 4.1% size of NMEA. These compact outputs result in reduced SD card access and write times, leading to lower power consumption.

Case 2: Large sample collection

In the second case, the GPS sample files were individually processed to generate each output format type and averages were calculated across all results. For these results, CSV output elements contained "lon,alt,alt,tim".

The first analysis on the performance of generic LZARI compression shows that compressing NMEA format output reduces it to 16.0% of its original size, KML to 22.1%, CSV-text to 23.8% and CSV-binary to 82.2%. The increasing percentages

reflect the increasing information density of the formatted data, but show that all types of output are still compressible.

The second analysis compares the performance of each formats output to the raw NMEA output (GGA sentences only) itself, showing that KML is 27.1% the size, CSV-text is 35.1% the size, and CSV-binary is 4.3% the size. If the outputs are compared after they have been compressed with LZARI, the results are 5.8%, 8.2% and 3.6% respectively. This illustrates the dramatic saving possible with CSV-binary, producing positioning data that is up to 25x smaller that raw NMEA material itself.

The third analysis investigates the relative performance in using the CSV-text or CSV-binary formats, showing that CSV-binary results in output 13.1% the size of CSV-text.

5.2. Capacity limits and maximum recording lifetimes

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

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

Typically in CSV mode, the binary format compression results in smaller output sizes. For example, the sample data (above) containing 2638358 records was encoded with CSV binary to 7.9MiB, representing approximately 3.14 bytes per record, i.e. 3.14 bytes per 1 second. In this case, a maximum file size of 508MiB would be realised in 47126 hours, i.e. 1963 days or 5.3 years.

In any of these cases, the SRAM buffer fills, whether 15 minutes, 41 minutes or otherwise, at which point 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, say 52 sectors. The SD card and SPI block are then powered down and the SRAM buffer is reset.

The primary purpose of the SRAM buffer, the binary formats, and the compression options, is not enable larger amounts of data to be logged. It is 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 module itself, are the largest consumer of power in the device.

Using the above example, over a period of 14 days (60 x 60 x 24 x 14 = 1209600 records), perhaps on an extended road, hike or boat trip, KML text mode would make 1378 writes (1209600 * 28 / 24576) of say 52 sectors, but the equivalent CSV binary mode may only make 154 writes (1209600 * 3.14 / 24576) of say 52 sectors. That’s an 88.8% reduction in SD card sector writes.

5.3. Power consumption, efficiency and battery lifetimes

To be completed in subsequent release: anecdotal evidence is good so far though!

LEGAL NOTICES

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