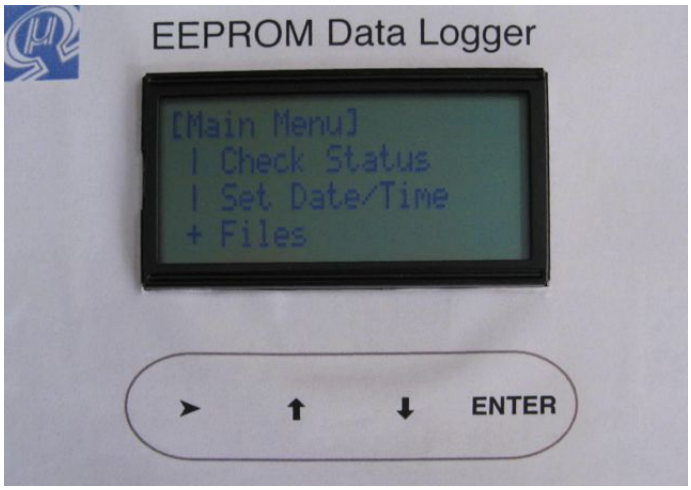
Micromega Corporation

Application Note 102
uM-FPU64

Logging GPS Data to EEPROM

Introduction

This application note provides an example of using the uM-FPU64 for logging GPS data to EEPROM. The hardware setup consists of an LCD display and four keys for the user interface, four I²C EEPROMs for data storage, and a serial input for the GPS data. This is a stand-alone application controlled entirely by the uM-FPU64. No external microcontroller is required. The front panel of the project is shown below. The keypad consists of four SMT dome switches mounted on a PCB and actuated through holes cut in the front panel. The front panel overlay is printed on a laserwriter and coated with a plastic laminate to give the appearance and functionality of a custom-built membrane switch.

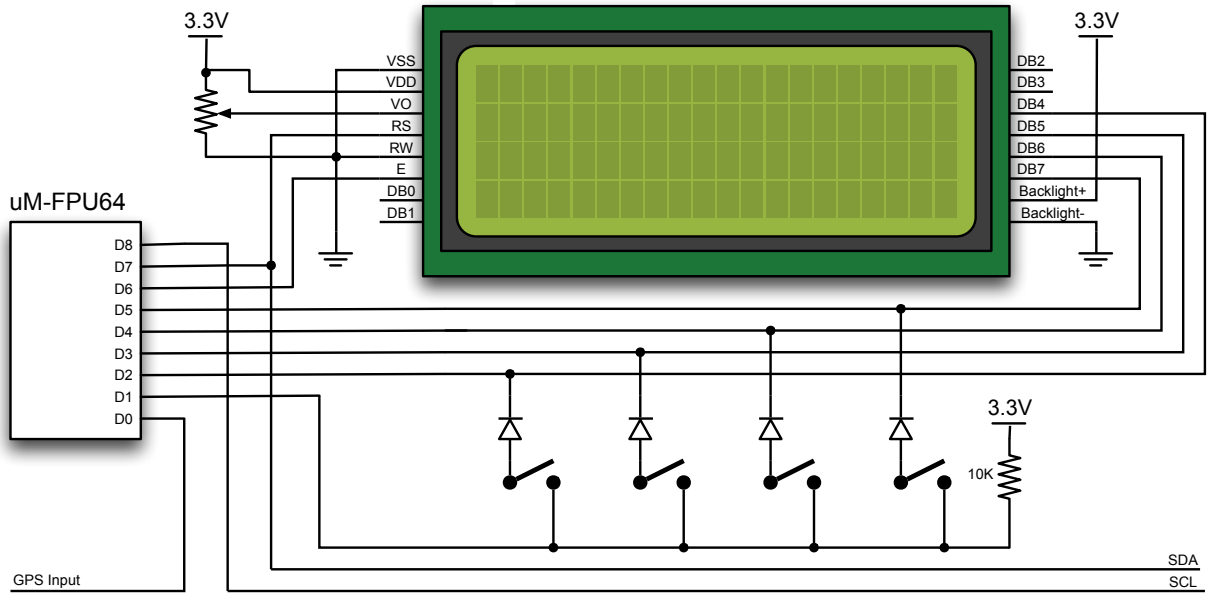


Code Examples provided by this Application Note

- `devio(LCD,…)` functions are used to interface with the LCD display.
- `devio(I2C,…)` functions are used to read and write to the I²C EEPROM.
- `digio(WRITE_BITP,…)` and `digio(READ_BITP,…)` functions are used to scan the keypad.
- `devio(FIFO1,…)` and `event(…)` functions are used to store GPS Track data.
- `rtc(…)` function is used for date/time conversions and time stamping.
- memory arrays are used for data buffers.
- `serial(…)` functions are used to parse GPS data.
- `serial(…)` functions are used to send data to the IDE *SEROUT Display* window.

Connecting the LCD, Keypad and GPS device

LCD Display and Keypad

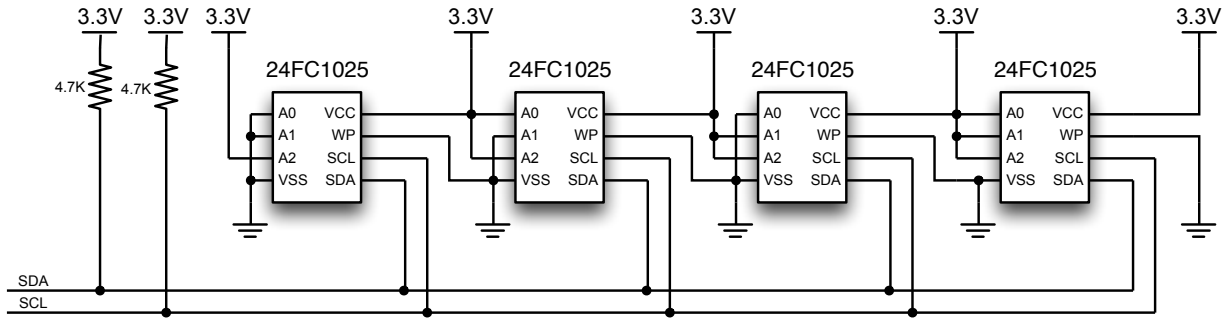


This code example uses a 4x20 LCD display and four switches for user input configured as a 4x1 switch matrix. The four data lines used by the LCD (`D2-D5`) are also used for the four column scan lines for the switch matrix. This reduces the number of pins required for the interface. Diodes are used to isolate the lines and prevent a key press from interfering with the LCD data. The switch matrix row input is connected to `D1`.

The serial input from the GPS device connected to pin `D0`.

EEPROM Storage

EEPROM Storage Array



Data is stored to EEPROM memory using an I²C interface. The example developed for this application note uses 24FC1025 EEPROM chips which can each store 1024K bits of information, or 128K bytes. The four chips are combined to provide a memory array of 512K bytes. The page size of the EEPROM is 128 bytes, so data is buffered in RAM until 128 bytes are accumulated, then the page is written to EEPROM. Partial pages are only written at the end of a file.

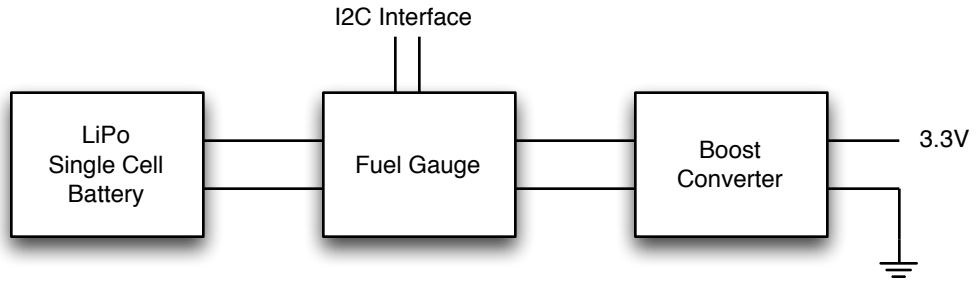
A simple file system is implemented that has a directory page stored as the first page of EEPROM memory. The directory page consists of eight, 16 byte entries that specify the file information. The 16 byte entries consist of four 32-bit words:

File Type	4 bytes	A number signifying the file type.
File Time	4 bytes	The date and time using uM-FPU64 RTC format.
File Address	4 bytes	The starting address of the first byte of the file.
File Size	4 bytes	The number of bytes in the file.

The 24FC1025 EEPROM handles the erase cycles automatically. To format the EEPROM for data storage the directory page is simply cleared to zero.

Power Supply

The project was designed to be portable. All components are housed in an enclosure with a single cell lithium battery as a power source. The battery is connected through a SparkFun LiPo Fuel Gauge (#TOL-10617) and then a SparkFun LiPower - Boost Converter (#PRT-10255) configured to provide 3.3V output. The fuel gauge provides an I²C interface to monitor the LiPo battery level.



Size of Data Logs

The 24FC1025 EEPROM memory array is 512K bytes. There are two types of log files:

Character Data (Log Serial Input)

This logs all serial input data, and can be used to collect raw data from a GPS device. The amount of data that can be logged will vary depending on the particular GPS device being used. A typical GPS device has approximately 450 characters per sample. If the update rate of the GPS is 1Hz, the maximum log duration is about 20 minutes.

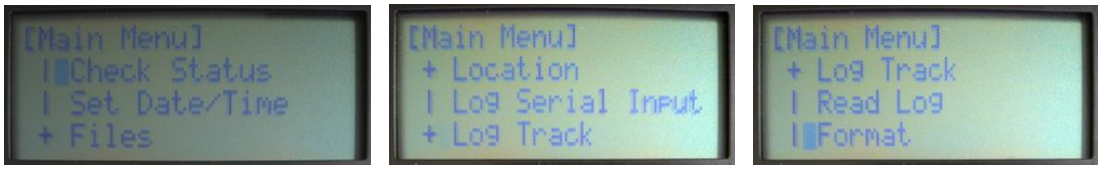
GPS Track (Log Track)

This logs the date/time, latitude, longitude and elevation for each sample received from a GPS device. The data/time value is stored as 32-bit integer, the latitude and longitude values are stored as 64-bit floating point values, and elevation is stored as a 32-bit floating point value. If the update rate of the GPS is 1Hz, the maximum log duration is just over six hours.

The application described by this application note collects all data samples. Longer log durations could be achieved by processing the data to discard samples that are very close, or if the direction of travel is approximately a straight line, midpoints can be discarded. This kind of data compression is done by handheld GPS devices.

Menu driven User Interface

The user interface is provided by the 4x20 LCD display and a 4-key keypad. The LCD Menu interface is also described in a separate document entitled *Code Example-LCD Menu*.



The following menu items are provided:

- Check Status
- Set Date/Time
- Files
- Location
- Log Serial Input
- Log Track
- Read Log
- Format

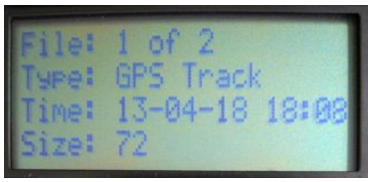
Check Status

Displays the LiPo battery level, the GPS satellite fix status, and number of satellites used in the fix. Press any key to exit.

Set Date/Time

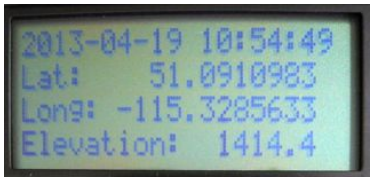
Reads the UTC (Coordinated Universal Time) from the GPS and allows the user to select a time zone. The UTC is displayed at the top of the display, and the time adjusted for the current time zone is displayed at the bottom. The *Up* key adds one hour to the time zone, the *Down* key subtracts one hour from the time zone, and the *Next* key adds half an hour to the time zone. Press the *Enter* key to exit and set the FPU RTC (Real-time Clock) with the current time. The RTC is used for setting the time in the file directory when new log files are created.

Files



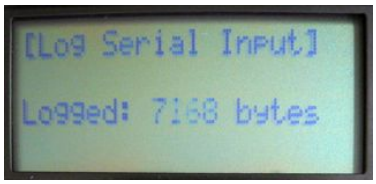
Provides a list of the log files stored on EEPROM. The *Up* and *Down* keys are used to move between file descriptions. Press the *Next* or *Enter* keys to exit.

Location



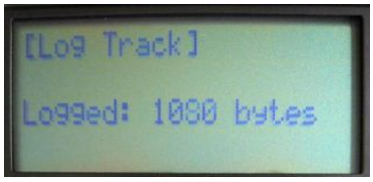
Displays the current time, latitude, longitude, and elevation received from the GPS device. Press any key to exit.

Log Serial Input



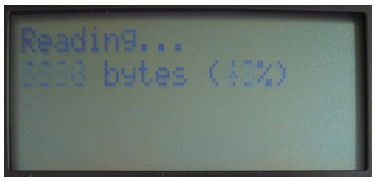
Logs all serial input received. When the EEPROM is full, logging will stop. Pressing any key will also stop logging.

Log Track



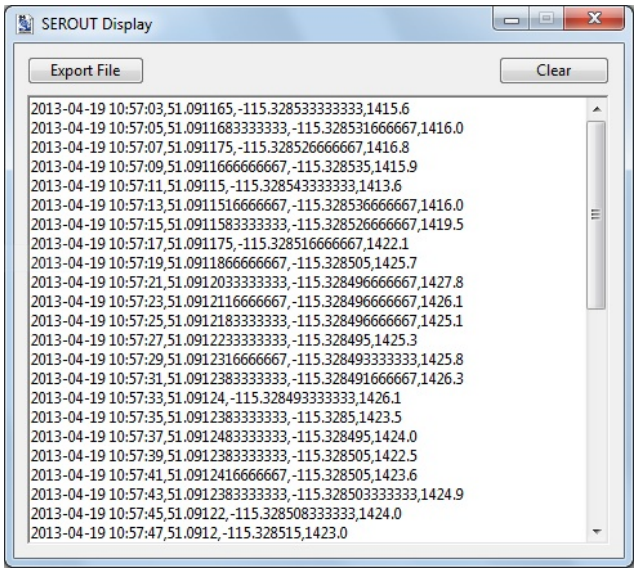
Logs the GPS track. It records the current time, latitude, longitude, and elevation received from the GPS device for each sample reported. The update rates of GPS devices can vary, but the default rate for most devices is 1 Hz. Press any key to exit. When the EEPROM is full, logging will stop. Pressing any key will also stop logging.

Read Log

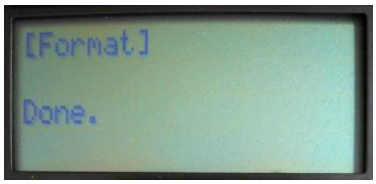


Reads data from the EEPROM and sends the data to the FPU serial output. It first displays a list of log files in the same manner as the **Files** command and allows the user to select a file. After navigating to the correct file using the *Up* and *Down* keys, select the file by pressing the *Enter* key. To cancel, press the *Next* key.

The log file is sent to the serial output. For character data files, all serial input characters are sent. For GPS track files, each sample (time, latitude, longitude, elevation) is sent as comma-separated values followed by a carriage return and linefeed. If the serial output is connected to the uM-FPU64 IDE, the data will be displayed in the *SEROUT Display* window.



Format



Clears the first page of the EEPROM to zero, which marks the EEPROM as empty. The first page is used to store a directory of file information.

FPU Functions

Source code file for this application note: *dataLogger.fp4*

`main`

Initializes the `LCD`, `FIFO1`, `ASYNC` and `I2C` devices, resets the LiPo Fuel Gauge, starts the FPU timer, and initializes registers. It then displays a startup screen, and enters the main program loop. The main program loop calls the `displayMenu` function to display the main menu on the LCD and allow the user to select a menu item. The appropriate function is then called for the menu command selected by the user.

`startupScreen`

Displays a startup screen with the program name and version.

`checkStatus`

Displays the LiPo battery level, the GPS satellite fix status and number of satellites used in the fix. The LiPo battery level is determined by reading a register on the MAX17043 LiPo fuel gauge. The satellite fix status is determined by reading the GPGGA and GPGSA sentences sent by the GPS device.

`setDateTime`

Reads the UTC (Coordinated Universal Time) from the GPS and allows the user to select a time zone. The UTC is displayed at the top of the display, and the time for the current time zone is displayed at the bottom.

`listFiles`

Provides a list of the log files on the LCD. The *Up* and *Down* keys are used to move between file descriptions. Pressing the *Next* key will return a value of -1. Pressing the *Enter* key will return the selected file number.

`location`

Displays the current time, latitude, longitude, and elevation received from the GPS device. The GPS data is received using the `serial(READ_NMEA+ASYNC)` function. The GPRMC sentence is used for date, time, latitude and longitude, and the GPGGA sentence is used for elevation. The `NMEA_to_dateTime` function is used to convert GPS the UTC date and time to an FPU date/time number. The UTC value returned is converted to the current time by simply adding the time zone adjustment. The latitude and longitude values are converted to decimal degrees by the `NMEA_to_degrees` function.

`logSerialInput`

Logs all serial input. The `fileNew` function is called to get the next available file location. The serial input data is received using the `serial(READ_CHAR+ASYNC)` function and written to a data buffer stored in RAM. When a page of data (128 bytes) is accumulated in the buffer, the page is written to the I²C

EEPROM using the `fileWrite` function. On exit, the remaining bytes in the data buffer are written, and the `fileClose` function is called.

`logTrack`

Logs the GPS track. It records the current time, latitude, longitude, and elevation received from the GPS device for each sample reported. The `fileNew` function is called to get the next available file location. The GPS data is received using the `serial(READ_NMEA+ASYNC)` function. The GPRMC sentence is used for date, time, latitude and longitude, and the GPGGA sentence is used for elevation. The `NMEA_to_dateTime` function is used to convert GPS the UTC date and time to an FPU date/time number. The UTC value returned is converted to the current time by simply adding the time zone adjustment. The latitude and longitude values are converted to decimal degrees by the `NMEA_to_degrees` function. The data/time value is stored as 32-bit integer, the latitude and longitude values are stored as 64-bit floating point values, and elevation is stored as a 32-bit floating point value. The `DEVIO(FIFO1,…)` function is used to store the data. The FIFO is used to handle the issue of the different data sizes and the page boundary. The FIFO is defined as 256 bytes, with the half-full event enabled. When a page of data (128 bytes) is accumulated by the FIFO, the page is written to the I²C EEPROM. On exit, the remaining bytes in the FIFO are written, and the `fileClose` function is called.

`readLog`

Calls the `listFiles` function to select the file to read. If no file is selected, the function exits. If the file is a character data file, the `readCharData` function is called. If the file is a GPS track file, the `readTrack` function is called.

`readCharData`

Reads character data from the EEPROM and sends the data to the FPU serial output. The `fileOpen` function is called to open the file and get the number of bytes in the file. The data is read from the EEPROM by the `fileRead` function and characters are output to the `SEROUT` pin using the `serial(WRITE_CHAR,…)` function. Progress is displayed on the LCD.

`readTrack`

Reads GPS track data from the EEPROM and sends the data to the FPU serial output. The `fileOpen` function is called to open the file and get the number of bytes in the file. Samples are read from the EEPROM using `DEVIO(I2C,…)` functions and output to the `SEROUT` pin as comma-separated values followed by a carriage return and linefeed. Progress is displayed on the LCD.

`NMEA_to_degrees(long) float64`

Uses the *field* argument to select the latitude or longitude field in the GPS sentence. It converts the value from DDDMM.MMMM format to Decimal degrees and returns the value. Decimal degrees (converted to radians) are used for navigational calculations.

`NMEA_to_dateTime(long, long) long`

Uses the field arguments to select the date and time fields in the GPS sentence. It converts the GPS date and time values to an FPU date/time number using the `RTC` function. The date/time value is returned.

`displayMenu() long`

Displays a menu on the LCD and gets user input from the keypad to select a menu item. The menu is stored as a comma-separated string in the string buffer. The first field is the menu title, and the remaining fields are the menu items. The *Up* and *Down* keys are used to move through the menu. The *Next* key returns zero, and the *Enter* key returns the number of the menu item selected.

`waitKey() long`

Calls the `readKey` function and waits until a key is pressed that is different then the last key pressed.

`readKey() long`

Returns the value of any keys pressed on the keypad. The `digio(WRITE_BITP,…)` function is used to write out sequential column values. Only one bit in each column value is 0, the rest are set to 1. For each column value `digio(READ_BITP,…)` is used to read the row value. If the selected key is pressed a 0 will be detected, and the corresponding bit is set a 32-bit register. The 32-bit value returned by the function has a 1 set for each key that is pressed on the keypad. If no keys are pressed a zero is returned.

`format`

Clears the first page of the EEPROM to zero, which marks the EEPROM as empty. The first page is used to store a directory of file information.

`fileNew(long) long`

Reads the first page of EEPROM and searches for the first available file location. The file address for storing data to EEPROM is set, and the file count is cleared to zero. The file address is returned by the function. If no files are available, zero is returned.

`fileOpen(long)`

Reads the file address and count from the EEPROM directory. The file address is returned.

`fileWrite(long)`

Writes a page of data (128 bytes) from memory to EEPROM, and updates the file address and count.

`fileWriteFIFO(long)`

Writes a page of data (128 bytes) from FIFO to EEPROM, and updates the file address and count.

`fileRead(long)`

Reads a page of data (128 bytes) from EEPROM unless the end of file is reached, in which case the remaining characters are read. The number of character read is returned.

`fileClose`

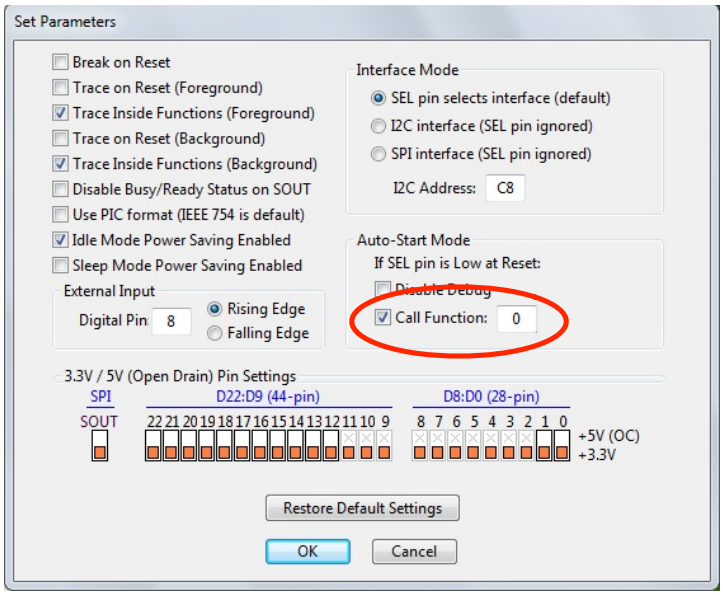
Updates the size of the file in the directory.

`EEPROM_startWrite(long)`

Starts an I²C transaction by calling the `DEVIO(START_WRITE,…)` function. The device address changes depending on the address of the page. There are two banks within each chip and four chips in the memory array. This function determines the correct device address. It also checks whether the device is ready. If a NAK is received after the start condition, the start condition will be repeated until an ACK is received, or the maximum write delay occurs.

Stand-alone uM-FPU64 Application

The uM-FPU64 can automatically call a function at Reset. To create a stand-alone application, the main function (function 0) should be called at Reset. After using the the uM-FPU64 IDE to program the data logging functions to Flash memory on the uM_FPU64, use the *Functions> Set Parameters* menu item to set *Auto-Start Mode*.



Further Information

See the Micromega website (http://www.micromegacorp.com) for additional information regarding the uM-FPU64 floating point coprocessor, including:

Application Note 101: *Reading GPS Data*
Code Example: *Interfacing Keypad Switches*
Code Example: *LCD Menu Interface*
uM-FPU64 Datasheet
uM-FPU64 Instruction Set

uM-FPU64 IDE User Manual
uM-FPU64 IDE Compiler Manual