Figure 1 is a block diagram depicting the basic operation of the ME Mega Executive. The ME is intended as an extremely rudimentary thread or task execution environment designed primarily for a Li-Ion battery power MPU system with solar power replenishment.

The basic operation begins immediately after “RESET”ing the MPU (ATmega2560 processor) and the execution of the Arduino IDE’s required setup() function in the DataReporter\_REV\_n\_n.ino source file.

The thread execution is mechanized by the code in the loop() function in the DataReporter\_REV\_n\_n.ino source file. The “for” statement block in loop(), represented by the roterary switch in the figure, executes thereafter at all times that the MPU is not “sleeping”.

The figure is intended to show that all threads, a.k.a. tasks or state machines, execute as long as their respective series switch is closed and the MPU is not sleeping.

The series switch is mechanized as a global array of boolean flags. True equals call-the-thread function and false equals don’t-call-the-thread function.

The roterary switch is mechanized as an array of pointers to the thread functions. The threads are mechanized as a function having no arguments, returning a void and containing a switch statement whose argument is the current executing state of the thread. The thread state is held in a global array of integers the state is generally controlled by the associated state machine itself.

Global, as used here, is the address space shared by the executive and all threads. All variables belonging exclusively to the threads, and referenced only by other states of the same thread are defined as static in the thread function.

To use the ME in an embedded application it is intended that you use the core set of threads and functions augmented by your specialized threads and functions added to taylor the ME to mechanize your application’s unique requirements. Some of the core threads may also have to be changed to provide further tayloring for your application’s unique requirements.

The core threads and functions are:

* setup()
* loop()
* Console
* LogData
* Monitor
* ResourceQueues
* Real Time Clock
* SD card
* Setup()

SystemLog

The Monitor thread is unconditionally scheduled at all times. The Monitor thread’s primary functions are:

* Maintain the MPU in the lowest possible current draw mode.
  + Sleep only if all threads indicate “sleep is don’t care for this associated thread”.
    - Mechanized as a(n) global integer/long 16/32 bit flags.
* Wake up for any of the following interrupts:
  + Bucket tip.
  + Watchdog timer. (approximately 9 secs.)
  + IDE console interrupt.
* Schedule other threads as required.

ME Resources shared by threads:

* System Log.
  + PushSysLog()
  + PopSysLog()
* SD Read/Writes.
  + All file objects are created at setup time.
  + Only read/writes to existing file objects are allowed.
* RTC Real Time Clock.
  + PushRtc()
  + PopRtc()

Data point buffering:

Adding a thread:

**A/D Monitor**

**(SOLAR\_OUTPUT\_MONITOR)**

**Digital Output**

**(SOLAR\_REG\_ENABLE)**

SD Card

(BOB-1140)

Cellular Shield

Sparkfun CEL-09607

(SM5100B)

5VDC Buck

Regulator

(KA278R05)

Solar Panel

Li-Ion

Battery

Li-Ion Battery Charger

Power Manager

(MCP73871)

2.5 VDC Voltage

Reference

(REF192)

5 VDC

Boost-Buck

Regulator

(S7V7F5)

3.3 VDC Buck

Regulator

(REG710)

DataReporter

Controller

Arduino Mega PRO

(ATmega2560)

Real Time Clock

(DS3234)

**SPI Bus**

**Solar Panel**

**Charge**

**Load**

**AREF**

**Antenna**

**A/D Monitor**

**(RFEGULATOR\_MONITOR)**

**RS232**

**AREF**

**A/D Monitor**

**(BATTERY\_MONITOR)**

**A/D Monitor**

**(LOAD\_MONITOR)**

**Three Digital Inputs**

**(BATTERY\_MNGR\_PGNOT**

**BATTERY\_MNGR\_STAT2BATTBATTERY\_MNGR\_STAT1\_LBO)**

**Figure 1**

**DataReporter Hardware Block Diagram**

**Arduino IDE Interface**

**Sparkfun (DEV-09873)**

**External Interrupt**

**(BUCKET\_TIP)**

**Power (VCC)**

**ATmega2560 A/D**

**Channel Voltage Monitoring Point**

**ATmega2560 General Purpose**

**Digital Output**

**ATmega2560**

**External Interrupt**

**ATmega2560 General Purpose**

**Digital Input**

**External Interrupt**

**(CONSOLE\_ACTIVE)**

Processor Control

State 1

State 2

State n

Thread 1

State 1

State 2

State n

Thread 2

State 1

State 2

State n

Thread n