Circuit Cellar & Texas Instruments Stellaris 2010 Design Contest Project

Registration number: **TI2835**

Abstract

Lithium-ion Battery Module with Battery Control Unit (BCU)

Current engineering trends are increasingly dealing with an increased interest, both financially and otherwise, in energy harvesting, storage, and waste reduction. One need not look much further than industry leaders in IC technology, such as Texas Instruments, to see the upswing in energy storage-related solutions. Energy storage is an especially hot engineering topic, examples being the current advent of emerging new methods of storing sunlight for later use and hybrid and fully electric vehicles.

In short, lithium-ion batteries provide a higher energy density per volume when compared to old lead-acid batteries; however, their current cost (up to several hundred dollars for current high-capacity models!) and need for careful monitoring to prevent damage or possible safety hazards presents the need for not just sufficient technology, but *good and reliable* technology to regulate the use of multi-cell battery modules.

Knowing this, when the 2010 Stellaris Design Contest opportunity presented itself, I found the idea –nay, the *pleasure*-of demonstrating, using an actual working prototype, how several design flaws in some current Li-ion systems could be overcome with some additional hardware and the right microprocessor solution. The Texas Instruments Luminary Micro LM3S9B96 microcontroller provides a great resource to implement a battery management controller (BCU) with a safety-certified real time operating system (RTOS) with compact and efficient battery monitoring, data reporting, and circuit protection.

The Goal of the Design: Current Solutions and an Alternative Approach

If we were to review some current battery management solutions, we would find surprising inefficiency & high cost. Some vulnerable points of some systems (which shall rename nameless) are:

1. Inability to handle an external power supply loss and implement a safe shutdown, ensuring the battery supply contactor is opened or **opened as intended regardless of external power**
2. Wasteful power conversion, or an inability to derive power from battery cell module being monitored, or to selectively employ power sources
3. Large physical size
4. No available safety-qualified operating system, if available at all on the target BCU hardware. This is especially a concern for human-controlled vehicles or systems involving human life
5. Inefficient circuit protection actuator/solenoid hardware (example: some require a large, always-on power source to keep a circuit protection contactor powered)

We shall demonstrate here how all of the above can be remedied in order to improve safety & reliability.

Figure 1 shows a comparison of two systems: the Stellaris design approach (employing SafeRTOS, also) vs. a traditional, inefficient and somewhat unsafe solution.

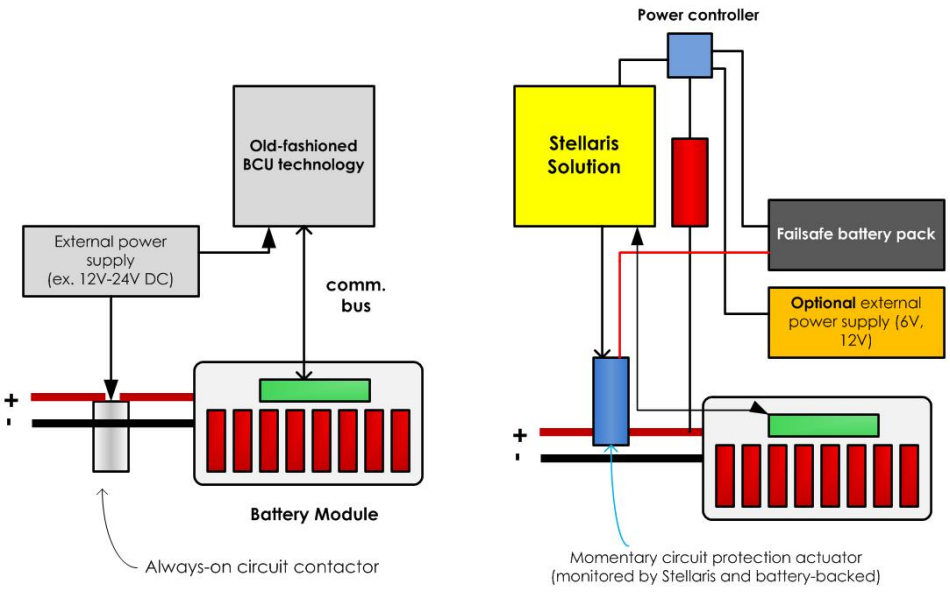

Figure 1. A conventional design versus the Stellaris-based approach

Reviewing Figure 1, we can see that if the conventional system loses its external power source, the entire BCU and control circuitry can fail-and will not fail *gracefully* by any means**! A better engineering solution would use a more safe & reliable approach** that does not necessarily mean extreme costs.

In the Stellaris design based approach, we solve the aforementioned issues (some of which can present safety-critical scenarios!) using the following:

1. We allow an external power supply to be used *where applications allow for it*, such as lab/test use or in home/industrial applications; otherwise, we allow the battery module and BCU to run from a step-down supply powered by the battery module(s) being monitored. Using the battery-backup module we can guarantee a power source for the circuit protection actuator and the BCU for emergencies
2. Power conversion requirements can be reduced because, with the exception of the cell stack power supply, **we only use power supplies momentarily and as really needed**. Battery-backed circuit protection supplies can use lower-cost and high-reliability linear voltage regulator circuits. Logic control circuits are minimum current draw and where possible zero-off-current designs
3. With the exception of some additional interface hardware such as our communication hardware, power boards, and other physical hardware, the design of the BCU board itself does not need to be much larger than the tiny EK-LM3S9B96 kit!
4. The Stellaris LM3S9B96 microcontroller provides a Wittenstein safety-proven high-reliability RTOS. SafeRTOS™ is a read-only memory (ROM) –based built-in preemptive round-robin operating system supporting priority-based functional tasks and ensures that safety-critical operations go unhampered
5. This design provides circuit protection using a *momentary* circuit protection actuator, consisting of a model system (for prototyping purposes) based upon a 12V DC solenoid that switches contacts only when needed. Additionally, the BCU monitors the auxiliary contacts to ensure the position of the contactor is known, and works with the battery-backup module to always ensure power is available for circuit –open functionality.

In the case of dying or undercharged backup batteries, the BCU places the battery module in a safe state, and if necessary, opens the circuit protection contactor and shuts down the module. The battery voltages are constantly monitored for safe working values. In other words, it's a "smart system" with a fail-safe approach to shutdown and safety monitoring.

Implementing the Project Design: The Hows, Whats, and Whys

The design in question certainly does require additional external hardware. A single-board solution simply is not feasible for the following reasons:

- Currently the Stellaris LM3Sxxx microcontrollers provide only 10-bit analog-to-digital converter (ADC) hardware. For accurate battery cell balancing and precise monitoring, we must use an external battery cell monitoring board
- Power source selection, and ensuring that a power loss does not occur (in addition to providing the means for the Stellaris microcontroller to sense the current power supply status voltages) will require a "power source controller" board
- Circuit protection will require a driver board, or "actuator controller board" to handle switching the solenoid driver. Also, control inputs will be optically isolated to prevent damage to the Stellaris from induced voltages from the actuator solenoid
- Current & temperature sensing: the BCU must be able to detect and report current battery cell module current (positive or negative, where negative means the module is in a recharge state) and operating temperature
- Data reporting: the user must have a means of receiving data in real-time for knowing the operational parameters including cell voltages. We will use an optional custom LCD monitor to do this, along with UART (RS232) and controller area networking (CAN) mediums
- Prototyping: to obtain a "proof of concept" via real-world and operational hardware, the entire system shall be built and debugged to the best extent possible. We need an assembled, working, and debugged battery module to do this
- As real high-capacity li-ion battery cells are both dangerous and expensive, we shall duplicate the functionality on a smaller scale using more readily available 18650 cells that share identical cell voltages to our ideal cells. Essentially, **we will safely and inexpensively simulate 50 ampere-hour** (Ah) cells within our budget, time, and parts-on-hand constraints
- Software implementation/RTOS: we shall use the SafeRTOS to implement our desired software safety-based system and fulfill the design rules of the Stellaris 2010 Design Contest guidelines

Making it Happen: A Module is Born!

Following the above guidelines has resulted in a complete (with the exception of some communication software not implemented before deadline, see the *Project Design Description* for more info) and working 8-cell li-ion battery module with custom boards, BCU, circuit protection contactor, and LCD monitor display.

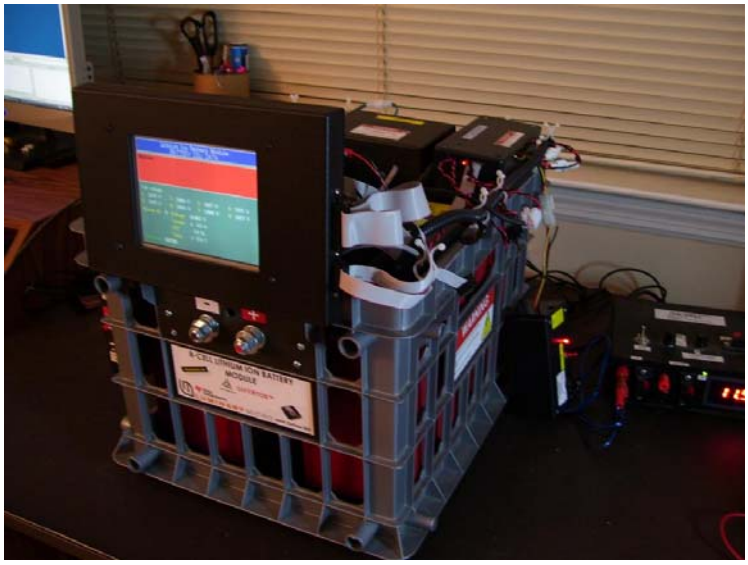

Figure 2: The up-and-running battery module, with operating SafeRTOS-based BCU

Figures 3 & 4 are provided for the reader in order to better understand the LCD display (visual data reporting) and additional custom hardware in the module.

Figure 3. The LCD monitor displaying module parameters including cell/ambient temperature, load current, individual cell voltages, total module voltages, real-time contactor position (open or closed), and any alarms as they appear. Note that the StellarisWare GRLIB graphics library is used to draw graphics and text. Custom drivers were developed to control the Solomon Systech SSD1963-based LCD driver board

Note that the State of Charge (SOC) functionality is implemented only in ASCII string conversion currently due to design deadline constraints. (Additional information in the Project Design Description)

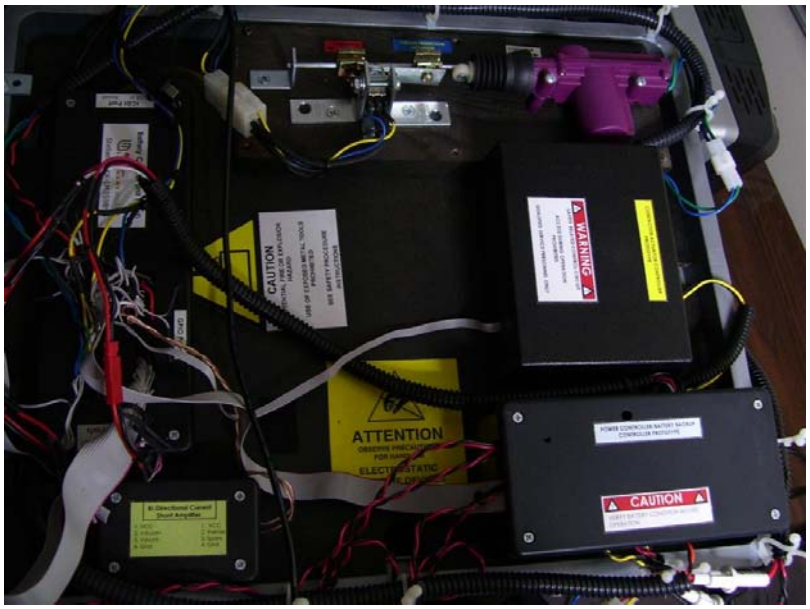

Figure 4. The top view: the various custom prototype hardware boards, in their enclosures, as required for operation. Top: circuit protection contactor (functional model); middle right: actuator controller board. Lower right: power controller board. Lower left: current shunt monitor board. Upper left: the Stellaris-based BCU board in its enclosure & wiring.

Functional Block Diagram: The System In a Nutshell

Figure 5 is an architectural/functional system block diagram of the design.

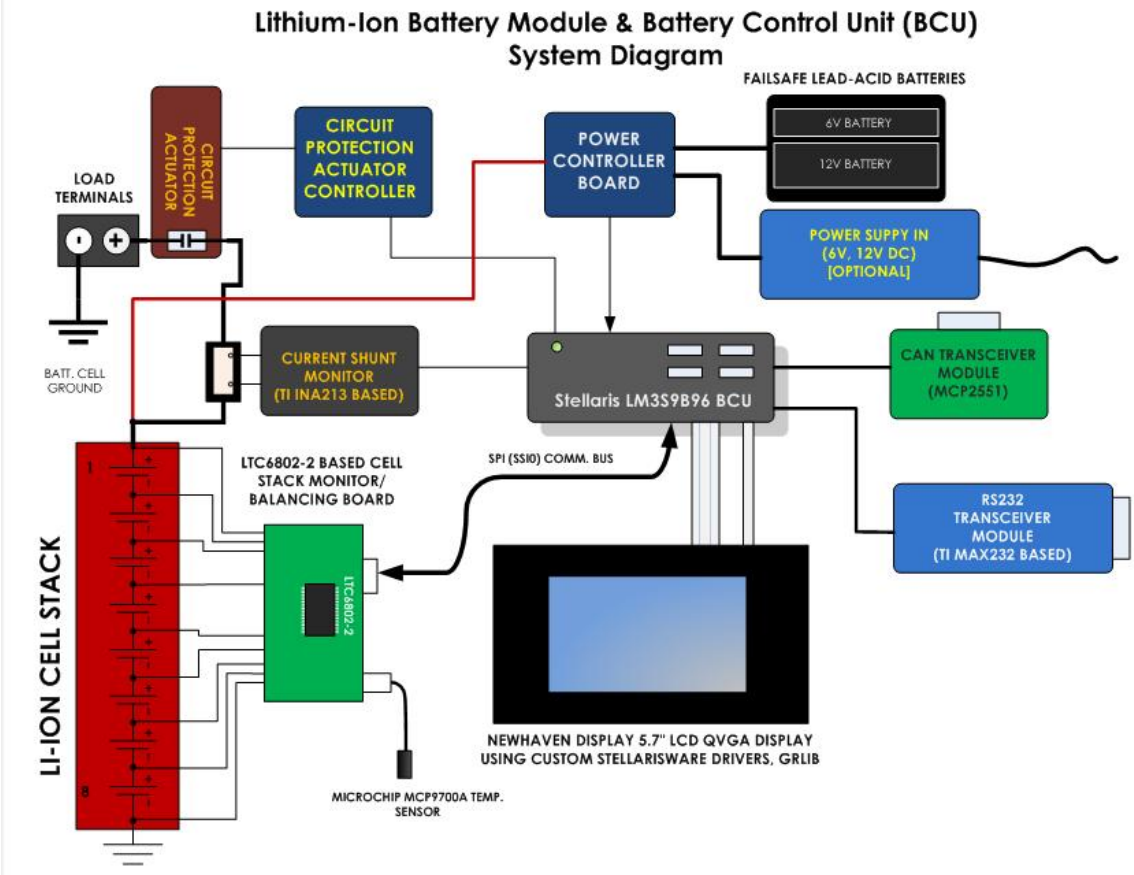

Figure 5. The system design overview. Note that additional information is provided in the Project Design Description

Code Samples & the Software Program: What Makes It SysTick?

The "nuts and bolts" of the software program can be best briefly described with the following:

- The current design employs two "tasks" (RTOS scheduler-called software functions, which do not exit) to operate: `LED_task and LTC6802_MODULE_task`
- `LED_task` is a simple "heartbeat" indicator for the BCU board, while the heart of the design is `LTC6802_MODULE_task`: a looping, iterative task to communicate with the custom battery cell stack monitor board, handle alarms, display data, and contactor responsibilities
- The SafeRTOS operating system operates based on the SysTick timer; this is a peripheral feature of the ARM CORTEX M3 Stellaris microcontroller, used to create a regular interval "tick" for an RTOS scheduler like ours to switch tasks as needed
- Module-critical functionality and alarm-handling is carried out in the `LTC6802_MODULE_task`, hence the name

Basically, the design works around a Linear Technologies LTC6802-2 integrated circuit (IC) which allows up to 12 li-ion battery cells to be measured and balanced, as desired by the communication bus "master." Our BCU acts as the master via the serial peripheral interface (SPI) bus or "SSI" as called in the Texas Instruments literature. We use peripheral SSI0 at as speed of 250kilobits/second (250Kbps) currently.

The LTC6802-2 provides a 12-bit ADC at 1.5mV/bit accuracy. This is much better than what the LM3S9B96 can currently provide, and allows for accurate cell monitoring and balancing control.

Important software functionality samples are:

(Start the battery cell voltage ADC)

```
LTC6802_broadcast_start_tempADC(&SPI0bufferTX[0]);
```

(After a brief delay, read back the cell voltage measurement results, which must be rearranged into values and stored in 16-bit variables)

```
for( i = 0; i < 16; i++)
{
   if(LTC6802_activeslaves[i] == TRUE)
   {
      error_temp_byte = LTC6802_address_read_temp_group(&SPI0bufferTX[0], &SPI0bufferRX[0],
                                                        &LTC6802_temperatures[i * 3], i);

      /*  Track communication errors.  If one was found, increment the error count var.          */
      if(error_temp_byte == COMM_ERROR_YES)
      {  /*  Is this the first error to appear? if so, reset counter, as this will allow us     */
         /*  to check the error count when the cycle loop limit is reached, to avoid rolling    */
         /*  over beyond the max. variable value                                                */
         if(LTC6802_commerror_count == 0)
         {
            ModuleLoopCycleCount = 0;
         }

         LTC6802_commerror_count++;
      }
   }
}
```

Additionally, we show here how the SafeRTOS tasks are "born." Note that the two tasks are created, and the SafeRTOS system can *only* run after the scheduler is started. Should the scheduler fail, the code will crash (and hence no run state):

```
/*  Create the SafeRTOS tasks, as needed; note any actions taken if the creation    */
/*  fails (returns value other than 0)                                               */
if(LEDTaskStart() != 0)
{
   while(1)
   {
     __nop();
   }
}
```

```
if(LTC6802_MODULE_TaskStart() != 0)
{
   while(1)
   {
      __nop();
   }
}

/*  Start the RTOS scheduler.  When done, we should have no way of returning back  */
/*  this point                                                                      */
xTaskStartScheduler(pdTRUE);
```

The central task `LTC6802_MODULE_task` must loop continuously; this is because the LTC6802-2 has a watchdog feature, and a communication loss for greater than 2.5 seconds will allow it to go into low-power "sleep" mode and lose all configuration values. Also, we wish to continuously monitor cell status in real-time to prevent any under or over-voltage conditions to the cells during charge, discharge, or any other time the BCU is active and the contactor is closed (module cell stack voltage applied to the connected load at the terminals).

Schematic Sample: A Sensible Current Sensor

Note that due to the multiple designs and complexity of some of the designs, only one schematic can be shown reasonably due to size and viewable scale constraints. Figure 6 illustrates the bi-directional current shunt monitor board, which allows the LM3S9B96 to measure and scale expended current (in amps) of the module.

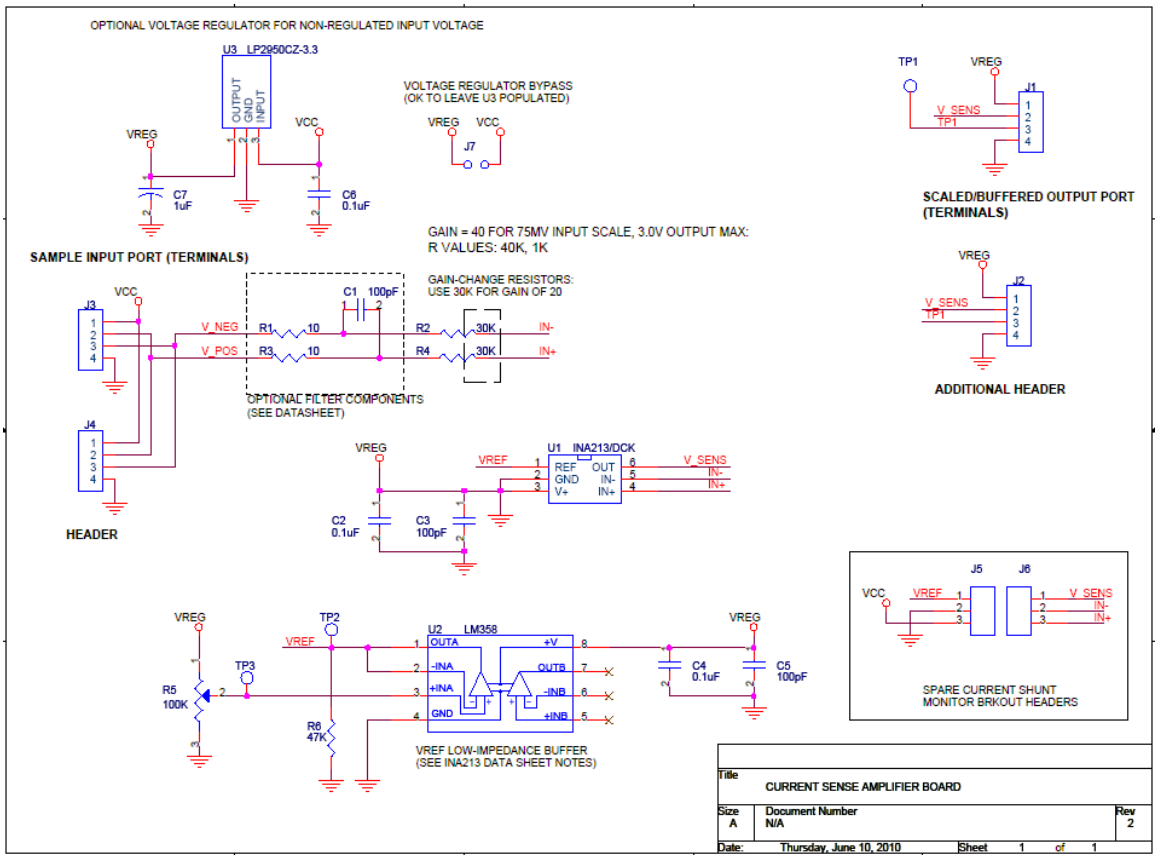

Figure 6. The bi-directional current shunt monitor board, based on a Texas Instruments INA213 with altered gain value. Note that a DC-biased scheme is used in the place of a negative voltage for negatives current readings. A reference voltage is used and buffered to set the zero-current output of the circuit to 1.5V.

In the real world, we must have a means of measuring battery cell current discharge, be it positive or negative. The Texas Instruments INA213-based current shunt monitor sense board is a simple but cost effective means to sense current from the cell stack. Here, due to scaled values, we are limited to a 2A current shunt (with voltage of 50mV at 2A) measured via the microcontroller's ADC peripheral and converted to an ASCII display string for data reporting.