To: LFEV-ESCM Team

From: Naing Minn Htet

Date: 24 March 2014

Subject: Issues with BMS Firmware

Abstract:

This memo describes the issues found with the LFEV'13 design of Battery Management System (BMS) firmware. There are two main bugs or issues with the current firmware for the BMS slave PIC controller. They are :-

1) The data returned is incorrect or corrupted. Usually, the first response from the slave PIC is different from what is expected.

2) Once around every ten I2C messages, the slave PIC will crash and require manual reset. Instead of crashing, the slave PIC could also perform erratically such as changing the board address or turning the bypass.

Technical Findings:

The firmware uses interrupt routine mainly. Interrupt routine is called when -

1) an I2C request addressed to the slave PIC needs to be serviced

2) the systems needs to gather and update data such as voltage and temperature. This is set with a timer.

The design handles all those interrupts in the interrupt function (isr) while the main function does little to no actual work. The isr() cannot be interrupted and any interrupts caused during isr() will be handled later.

Every time a byte is received from the master, the slave hardware sets the i2c interrupt. The software then determines the appropriate action within the interrupt function. The actions include storing the bytes if the whole message is not completed or transmitting a byte at a time if it is a read request.

The problem with the design is that the slave firmware does not ask the master to wait till it finishes processing the request. Thus, it could be performing the second interrupt function of isr which is updating data when the I2C request is received. Since the slave PIC does not ask the master to wait and the master does not know that its request has not been serviced yet, it continues to send more bytes. This could lead to erratic behavior as what the slave PIC was not able to record the bytes fast enough and could interpret the request wrongly.

Another problem is the concurrency issues found with the interrupt routine. The concurrency issue arises when a particular data or memory is read while it is being written or changed. In the design, while the main function is accessing the memory, it could be interrupted and the memory it was accessing could be changed. The design should prevent this from happening as this could lead to wrong or corrupted data being returned to the I2C master.

Recommendations and Decisions:

To solve the issues, I recommend two changes to the BMS firmware.

1) The slave PIC should ask the master to wait till next byte if the I2C request is still being serviced.
2) The design should prevent concurrency issues from happening by making sure that no data or memory could be changed while it is being read.

For the first change, SSP1CON2bits.SEN should be set to allow clock stretching. Setting this bit will allow the slave hardware to hold the I2C clock line after every byte, preventing the master from sending additional bytes until the slave firmware releases the clock line by setting SSP1CON1bits.CKP. Thus, the change conisists of setting SSP1CON2bits.SEN and then in the interrupt funciton (isr) , SSP1CON1bits.CKP should be set after the request is being handled.

For the second change, I recommend moving the i2c_complete section of the main to the interrupt function (isr). The i2c_complete section reads the received bytes (rxByte). Since it is inside the main function, rxByte could be changed by the interrupt function while it is being read by the main function. The slave PIC does not allow isr() to be interrupted. By moving the i2c_complete section into isr(), we can assure that rxByte would not be written while being read.

I also recommend loading the transmit bytes only once. In the current design, the slave PIC loads the requested data into txByte array and then sends the first byte. Then, it loads the data again and sends the second byte. If the data is changed between sending the first byte and the second byte, the bytes would not be consistent. To prevent this, I recommend only loading the transmit bytes once by adding “if (it == 0)” in front of “loadTransBuff(command_lower_4)”.

Attached Documents And Useful Information:

PIC16LF1827 Data Sheet - Section 25 - Page 233
http://ww1.microchip.com/downloads/en/DeviceDoc/41391B.pdf

Firmware.c - LFEV'13 source code for BMS firmware

http://sites.lafayette.edu/ece492-sp13/files/2013/05/firmware.zip