Application Report

SLAA529A–March 2012–Revised June 2012

MSP430™ Based Lithium-Ion Polymer Battery Charging and Gauging Solution Using USB Power

Tyler Witt, Bhargavi Nisarga *MSP430 Applications*

ABSTRACT

The use of USB power to charge Lithium-Ion Polymer (LiPo) batteries in electronic systems and devices has become quite popular. To assist in the development of such circuitry in embedded system applications, this application report provides the details on the hardware and software requirements needed to not only implement a battery charging solution using the bq24230 battery charger and the MSP430™ USB device (in a switched context USB power configuration), but also to monitor the battery status using the bq27410 fuel gauge to read important battery statistics and status information.

The example cases discussed in this application report provide a basic understanding of what needs to be done at production, as well as on the application level, to achieve the aforementioned goals. The associated software provides library functions to interface and communicate with the bq27410 fuel gauge, applicable to any USB-equipped MSP430 device. The software also includes a demo application that integrates the USB stacks and the fuel gauge library functions to read the battery information from the fuel gauge and transmit it to the PC through USB Communications Device Class (CDC).

Source files and other collateral that are discussed in this application report can be downloaded from http://software-dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSP430_LIPO/latest/index_FDS.html.

Contents

1 Introduction 2
2 LiPo Battery Charging Solution Using MSP430 and bq24230 2
3 MSP430-Based Fuel Gauging Using bq27410 4
4 MSP430 USB Operation 7
5 LiPo Battery Charging and Fuel Gauging Demo Application 9
6 References 12
Appendix A Associated Code Files 13
Appendix B Demo Hardware Schematic 14

List of Figures

1 Block Diagram - Battery Charging Using MSP430 2
2 Block Diagram - MSP430-Based Fuel Gauging Using bq27410 4
3 bq27410 Fuel Gauge - Flash Data Write Flow Chart 5
4 bq27410 Fuel Gauge - Flash Data Read Flow Chart 6
5 MSP430-Based Battery Charging and Gauging Demo - High-level Flow Chart 10
6 MSP430-Based Battery Charging and Gauging Demo - Detailed Flow Chart 11
7 MSP430-Based Battery Charging and Gauging Demo - HyperTerminal Screenshot 12
8 MSP430-Based Battery Charging and Gauging Demo Schematic 14

List of Tables

1 bq24230 Battery Charger Configurations 3
2 Charging Circuit - Power Numbers 3
3 USB Connection States and Operation 9

4 Power Consumption 12
5 Associated Code Files 13

1 Introduction

This application report provides a complete solution for charging a LiPo battery through use of USB power, interfaced with an MSP430F6638 (although any USB-equipped MSP430 device can be used), and monitoring the battery using a bq27410 fuel gauge. The two main goals of this application report are:

- LiPo battery charging with USB power by a bq24230 battery charger
- Battery fuel gauging by a bq27410 fuel gauge

This application report describes the hardware chosen for this application, why it was chosen, and what software solutions are needed to most effectively achieve the goals. Power consumption numbers for each different component are also included, highlighting the system’s low-power requirements.

NOTE: A LiPo battery with 3.7-V nominal voltage and 300-mAh capacity is used for proof-of-concept and demonstration. [1]

2 LiPo Battery Charging Solution Using MSP430 and bq24230

This section explains in detail what hardware components are used for the battery charging circuitry solution (see Figure 1) and why they are selected. It also reports the power measurements of each device in the system.

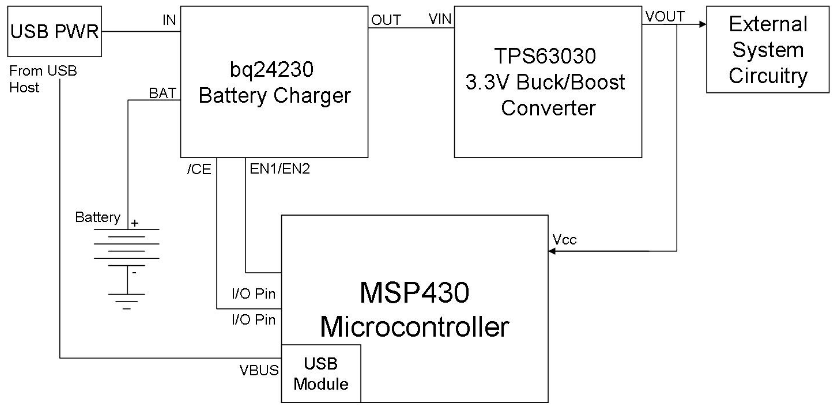


Figure 1. Block Diagram - Battery Charging Using MSP430

See the hardware schematic connections in Appendix B.

2.1 *bq24230 Battery Charger*

The bq24230 is a low power Li-Ion battery charger and power-path management IC [2]. It automatically switches the power source between USB power and battery power for supplying power to the MSP430 and the rest of the system. Depending on if the $\overline{\text{CE}}$ (Charge Enable) pin on the bq24230 is low or high, the battery charging by USB is enabled or disabled, respectively.

Both the USB power and battery are connected to the bq24230 battery charger. When the USB power is available, the charger uses the USB power to provide a regulated 4.4 V at the charger output, regardless of the battery’s connection status. When USB power is not available, the bq24230 device supplies the battery voltage at its output. Table 1 lists the various charger configurations.

Table 1. bq24230 Battery Charger Configurations

Battery Connected?	USB Connected?	bq24230 EN1, EN2	bq24230 $\overline{\text{CE}}$	Charger Configuration
Yes	Yes	LOW	LOW	Charging Battery
No	Yes	LOW	HIGH	USB Power Only
Yes	No	LOW	LOW	Battery Discharging
Yes	Yes	HIGH	LOW	Battery Discharging (USB in Suspend Mode)

The EN1, EN2, and $\overline{\text{CE}}$ pins of the bq24230 are controlled by the MSP430, and by default they are pulled high or low by the pullup or pulldown resistors, respectively, that are internal to the bq24230.

The bq24230 also has the option of being put into standby mode, which disables the charging circuitry and runs the system off of battery power. This is enabled by driving EN1 = EN2 = HIGH. This configuration is enabled by the MSP430 when the USB host puts the device into USB Suspend mode. To re-activate charging, EN1 and EN2 are driven LOW. For more detail on USB suspend mode, see Section 4.2.1. This configuration is used to ensure that the total current drawn from the USB host is less than 500 µA.

In addition to standby mode, the $\overline{\text{CE}}$ pin is used to enable and disable charging to the battery. This action is also managed by the MSP430 and is enabled in cases when only USB power is present. When the $\overline{\text{CE}}$ pin is driven HIGH, charging is disabled, and when the $\overline{\text{CE}}$ pin is driven LOW, charging is enabled. In this configuration, however, the charger is not put into standby mode, and USB is used to power the bq24230 charger regardless of the state of the $\overline{\text{CE}}$ pin.

The bq24230 was chosen because it has low standby current numbers and because it is a simple solution to charging the battery.

2.2 *TPS63030 Buck-Boost Regulator*

The TPS63030 is a high-efficiency single-inductor buck-boost converter [3]. A 3.3-V buck-boost regulator is used instead of a low dropout regulator (LDO) to ensure that regulated 3.3 V is output to supply the MSP430 and the system, without regard to the charger output voltage that is input to the buck-boost (that is, either 4.4-V regulated output from the charger when USB is connected or battery voltage, V_{BAT} = 4.2 V to 3.0 V, when USB is not connected). If 3.0 V is sufficient for system operation, then an inexpensive LDO could be used instead.

NOTE: The PS/SYNC pin of the TPS63030 is pulled LOW to enable the power save mode.

2.3 *Power Consumption*

Table 2 shows the power consumption and voltage output values for each of the devices in the charging circuitry.

Table 2. Charging Circuit - Power Numbers

	Current Consumed		Voltage Output	
Power Configuration	**Charger**	**Buck-Boost**	**Charger**	**Buck-Boost**
Only battery connected	0.3 µA	30 µA	V_{BAT}	3.3 V
Both battery and USB connected (with battery charging enabled ($\overline{\text{CE}}$ = LOW)	3 mA + battery charging current (max = 100 mA)	300 µA	4.4 V	3.3 V
Both battery and USB connected (with USB in suspend mode (EN1 = EN2 = HIGH)	0.1 µA	30 µA	V_{BAT}	3.3 V

3 MSP430-Based Fuel Gauging Using bq27410

This section describes the basics of the hardware that used for the task of fuel gauging the battery (see Figure 2) and the different protocols and procedures that are required to communicate with the gauge.

This section discusses the following topics:

- Basic overview of the hardware
- I2C communication protocol and procedure
- Other functions (battery detect and battery-low interrupt)

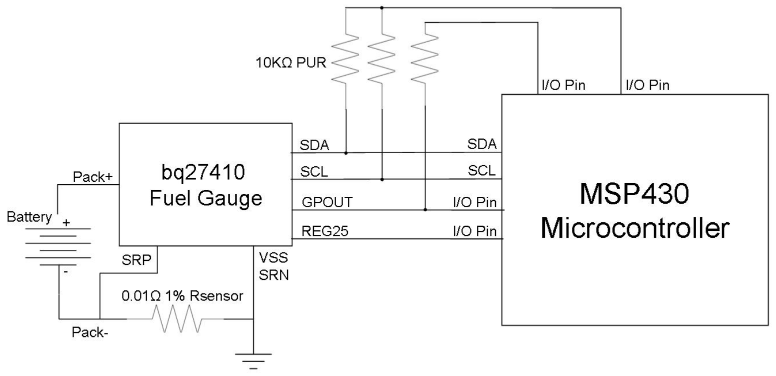


Figure 2. Block Diagram - MSP430-Based Fuel Gauging Using bq27410

3.1 *bq27410 Fuel Gauge*

The bq27410 is a system-side Impedance Track™ fuel gauge with direct battery connection. This device is used to track the vital statistics of the battery that is being used in the system. It communicates as the slave to the MSP430 using I^2C protocol. The gauge requires minimal user configurations and firmware for accurate fuel gauging. [4]

NOTE: The SDA and SCL pins of both the MSP430 and the bq27410 must be pulled up to the same voltage level. In this case, they are pulled up to the MSP430 DVCC by a GPIO pin. The SDA and SCL pins are pulled up using a GPIO pin so that they can be turned off when not being used, to save power.

3.2 *Communication Protocol and Procedure*

In addition to the basics of I^2C protocol, communicating with the bq27410 fuel gauge requires a certain protocol for successful data communication.

The data flash (accessed by the Extended Data Commands) on the bq27410 contains important values used by the fuel gauge to track the battery characteristics that show its current status. The contents of the data flash can be programmed to fit the needs of a particular battery's chemistry or altered for smaller customizable effects. Each of these values can be modified and verified as shown in the later sections of this document.

The RAM (accessed by the Standard Data Commands) on the bq27410 holds values that show the current status of the battery. Values such as the temperature, voltage, remaining capacity, and average current can all be read out from their respective RAM locations using the correct command code as specified in the data sheet and in the following sections.

NOTE: For more information on other functions and values that can be set or read in the bq27410's data flash or RAM, see the device-specific data sheet [4].

Interfacing with the bq27410 data flash and RAM are treated separately, as they have different procedures.

3.2.1 Writing and Reading bq27410 Data Flash

For a particular battery selected, the data flash values remain the same, and therefore, writing to and reading back from the data flash is done only at a production level and are customized to the battery's chemistry.

3.2.1.1 Writing to bq27410 Data Flash

Figure 3 shows the procedure to write the data flash on the bq27410.

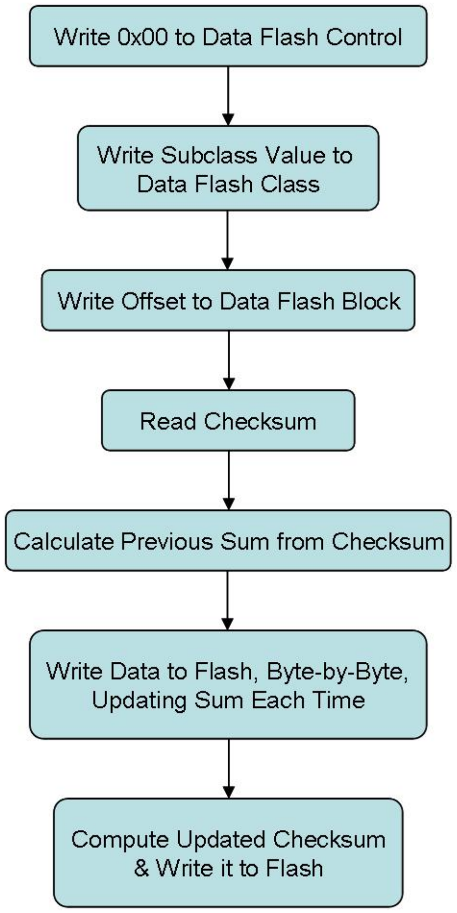

Write 0x00 to Data Flash Control
Write Subclass Value to Data Flash Class
Write Offset to Data Flash Block
Read Checksum
Calculate Previous Sum from Checksum
Write Data to Flash, Byte-by-Byte, Updating Sum Each Time
Compute Updated Checksum & Write it to Flash

Figure 3. bq27410 Fuel Gauge - Flash Data Write Flow Chart

What is most important to this procedure is ensuring that the correct checksum is being written. Without the correct checksum written, the data flash block is not updated and retains the previously programmed values. To ensure that the update happens correctly, the firmware (bq27410_config.c and bq27410_config.h) reads the previous checksum, calculates the previous sum from this value, and updates the sum byte-by-byte. By subtracting the previous read value (see Section 3.2.1.2) and adding the new written value to the running sum of the data block, an accurate calculation value of the checksum is ensured.

Another key point to writing the data flash is the offset. The offset that is in the data sheet is a data byte offset, not a block offset. As such, this offset value is assessed within the loop to write new data. For more information, see the firmware associated with this application report.

3.2.1.2 Reading from Data Flash

Figure 4 shows the steps for reading the data flash.

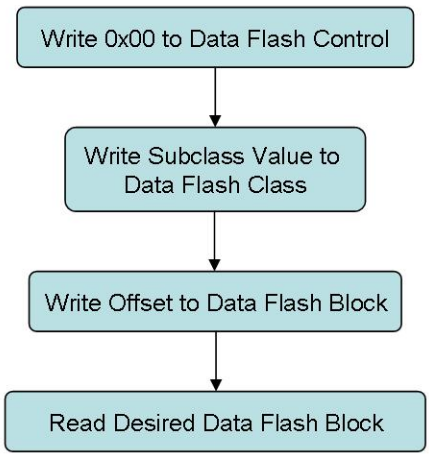

Write 0x00 to Data Flash Control
Write Subclass Value to Data Flash Class
Write Offset to Data Flash Block
Read Desired Data Flash Block

Figure 4. bq27410 Fuel Gauge - Flash Data Read Flow Chart

3.2.2 Writing and Reading the bq27410 RAM

As mentioned before, writing to and reading from the RAM of the bq27410 is different from writing to and reading from the data flash. The method for reading and writing the device RAM is a much more direct process.

To access the RAM values, only one byte of data has to be written. Following this, the corresponding data values can be directly read back with a read of the sufficient amount of data bytes.

The command bytes that are used to access the bq27410 RAM are referred to as the "Standard Data Commands" in the device data sheet [4]. These values can be read out individually or as an entire block of data.

3.3 Battery Insertion and Removal Detection

As mentioned in the section regarding the battery charger (bq24230), a combination of commands and interactions between devices is necessary to accurately detect a battery connection event and to efficiently handle such an event. To do this successfully, the application uses the $\overline{\text{CE}}$ functionality of the bq24230 in conjunction with polling the REG25 pin of the bq27410 to test if a battery is attached or not.

When the battery is removed, the REG25 pin of the bq27410 goes LOW, causing the falling edge event and an interrupt on the related MSP430 GPIO pin. In the GPIO ISR, $\overline{\text{CE}}$ is set HIGH to disable charging and conserve current and then the interrupt edge select for the GPIO pin is toggled.

When the battery is inserted, the REG25 pin of the bq27410 goes HIGH, causing the rising edge event and an interrupt to be triggered. In the GPIO ISR, $\overline{\text{CE}}$ is set LOW to re-enable charging, and the interrupt edge is toggled again. A flag is also then set by the ISR that allows the BAT_INSERT command to be sent prior to the next access of the bq27410 RAM contents.

NOTE:

- If USB power is not present in the system, a battery removal is treated as a loss of power to the system. Upon reinserting the battery, a power on reset (POR) is triggered for the MSP430, thus resetting the device.
- The BAT_INSERT command should only be sent to the bq27410 when a battery insert event is detected and not every time the RAM is accessed. While sending the command for RAM accesses does not cause any communication problems, it is done in best practice to reduce power consumption of the system.
- Removing the battery from the system when the USB is still connected is not fully supported in this solution. If the battery is removed when the MSP430 is communicating with the fuel gauge through I2C, the communication hangs and the MSP430 requires a reset or power cycle to resume operation. This configuration can be accounted for by adding an I2C time-out (planned for support in a future release).

3.4 Battery State of Charge Interrupt or Battery-Low Detect

The GPOUT pin of the bq27410 fuel gauge can be used for either of the two following functions: state of charge (SOC) interrupt or battery-low detect interrupt.

For this application, the state of charge interrupt function of the GPOUT pin was chosen over the battery-low detect function. This was chosen because the internal battery-low detect thresholds of the device were too high to be applicable to the 300-mAh Tenergy battery [1] used in the demo application.

The state of charge interrupt helps to chart the battery’s charge from a high-level perspective. That is, the complete battery charge range (0 to 100%) is divided into "x" sections (also referred to as "bars" in this application report) with each section representing "100 / x"% charge. When the battery’s charge reaches a limit of one of these sections, an SOC interrupt is generated on the GPOUT pin. The GPOUT pin is connected to an interrupt capable GPIO pin on the MSP430, which increments or decrements the "bars" counter based on if the battery is charging or discharging, respectively. Also, upon a battery insert event, this counter is refreshed using the most recent SOC reading from the bq27410 RAM to re-establish the battery’s status.

4 MSP430 USB Operation

This section describes the basics of the different USB power configurations and connection states, as well as how each USB state is handled in this particular application. It also describes the power consumption values of individual devices in the system and the entire system in each of the different USB power configurations.

4.1 USB Power Configuration Basics

When a USB device (meaning not just the MSP430, but the entire physical device) is connected to a USB host, the host provides 5-V power over the USB cable; this power source is referred to as VBUS. For devices that are permanently "tethered" to the USB host (that is, devices that function only when attached to the host), this "VBUS" may eliminate the need for a battery or other local source. Such a device is said to be *bus-powered*. By strict definition, a bus-powered device derives all of its power from VBUS.

In applications where the USB device is not permanently tethered to the host (that is, the device has an option to be disconnected and reconnected back to the host intermittently), the device might have its own local power source (battery or some other external source). A device that derives some of its power from a local source while connected to a USB host is called a self-powered device.

A USB device with its own local power source often powers itself completely from the local source when not attached to the host but takes advantage of VBUS when attached. This is often necessary because keeping an active connection to a USB host drains considerable power, adding additional load on the device’s battery. However, because the USB connection requires very little power when the USB device is suspended by the host, the device may be designed to revert to local power during suspend.

The design used in this application report performs similar to this. When not attached to a host, it powers itself from the battery. When there is an active USB connection to the host, it derives all its power from the host. When the host suspends the device, most power is once again derived from the battery, even though attached to the host. The only power that continues to be derived from the host is the power for the MSP430's USB module, drawn through the device's VBUS pin (see Section 4.2.1).

4.2 USB Connection States

There are six different USB connection states possible for the MSP430 USB device:

- Disconnected – USB device not connected to the USB host.
- Connected, No Enumeration – USB device is physically connected to host, but the device has not yet asserted the D+ pullup to alert the host of its presence, and thus the host has not yet enumerated it.
- Enumeration in Progress – the host is in the process of enumerating the device.
- Enumerated, Active – enumeration has completed, USB device is active and ready for communication.
- Enumerated, Suspended – device is enumerated, but the USB host has suspended it (often done by PCs when they are put into standby mode).
- Suspended, No Enumeration – USB device enters this state when it is connected to host that was already in standby at the time of connection, or when attached to a USB hub that is not connected to a host but is self-powered. In the latter case, the USB device sees VBUS power and, therefore, believes a host is present; however, because there is no signaling on D+ or D-, the device becomes suspended.

For more information about how these states are accessed and how the different events for changing states are handled, see the *MSP430 USB API Programmer's Guide* that accompanies the USB Developer's Package [5].

4.2.1 USB Suspend Mode

USB suspend mode is one of the USB connection states in which the USB host can suspend the USB device at any time. When the USB device is placed in suspend mode, no communication can take place and the power drawn from the 5-V USB power (VBUS) should be less than 500 µA.

The charging solution provided in this application report accounts for this mode by driving the EN1 and EN2 pins of the bq24230 battery charger HIGH to disable battery charging by USB and to allow the system to run off battery power alone. In this state, the only power that the system draws from VBUS is the power for the MSP430's USB module, drawn through the VBUS pin. This power is less than 500 µA, including the approximately 200 µA that is required to maintain the pullup on D+.

4.2.1.1 "USB Power Only" Configuration

When only USB power is input to the bq24230 charger (that is, no battery connected to bq24230 BAT pin), with EN1 = EN2 = LOW, the bq24230 can source up to 100 mA of current from USB bus power. If the USB host enters suspend mode in this power configuration, the application firmware should not drive bq24230 EN1 and EN2 pins HIGH, because there is no battery present in the system to power the system. In such application cases, the user must account for this configuration by reducing the system power (drawn from bq24230) to be less than 500 µA to comply with the USB-IF specification.

Also, during MSP430 USB enumeration, the USB host sends USB Suspend and Resume commands and, to account for this particular power configuration, the application firmware should not set EN1 = EN2 = HIGH to disable charging and allow the system to run off battery power.

5 LiPo Battery Charging and Fuel Gauging Demo Application

This section discusses the demo application as a whole. It combines all the information from the previous sections to give a full description of the final demo application.

5.1 Demo Application Hardware

The LiPo battery charging and fuel gauging circuit should be connected to the MSP430F5638 or MSP430F6638 target board. See Appendix B for the related schematic.

5.2 Demo Application Operation

The basic operation of the demo application is shown in Table 3, Figure 5, and Figure 6. When the MSP430 is powered up, the device initializes all the pertinent peripherals and checks if a battery is connected. If the battery is present, then RAM contents of the bq27410 fuel gauge are read and stored locally. The device then checks the USB status. If the USB is connected and enumerated, the device sends the fuel gauge information by USB Communications Device Class (CDC) to the PC.

NOTE: Using USB CDC protocol generates a virtual COM port on the PC and allows a UART-like connection between the PC and the device.

Table 3 describes the actions taken in each of the different USB connection states.

Table 3. USB Connection States and Operation

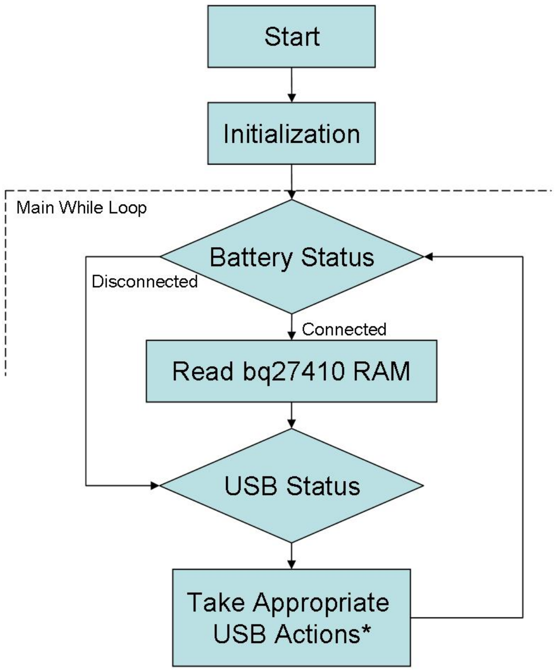
USB Connection State	Actions Taken in Demo Application
Connected, No Enumeration	Assert the D+ pullup (alert host of presence), go to sleep for 5 seconds, check if battery is connected
Enumerated, Active	Send fuel gauge RAM data through USB CDC in background, go to sleep for 5 seconds, check if battery is connected
Enumerated, Suspend	Put charger in standby mode, sleep for 10 seconds, check if battery is connected
Enumeration in Progress	Do not go to sleep, check if battery is connected
Suspend, No Enumeration	Sleep for 10 seconds, check if battery is connected
Disconnected	Sleep for 5 seconds, check if battery is connected

In each of these cases, if there is a battery connected, the RAM contents of the bq27410 fuel gauge and then the USB state are evaluated.

For more detailed regarding the demo firmware and the associated code files, see Appendix A.

5.3 Application Flowcharts

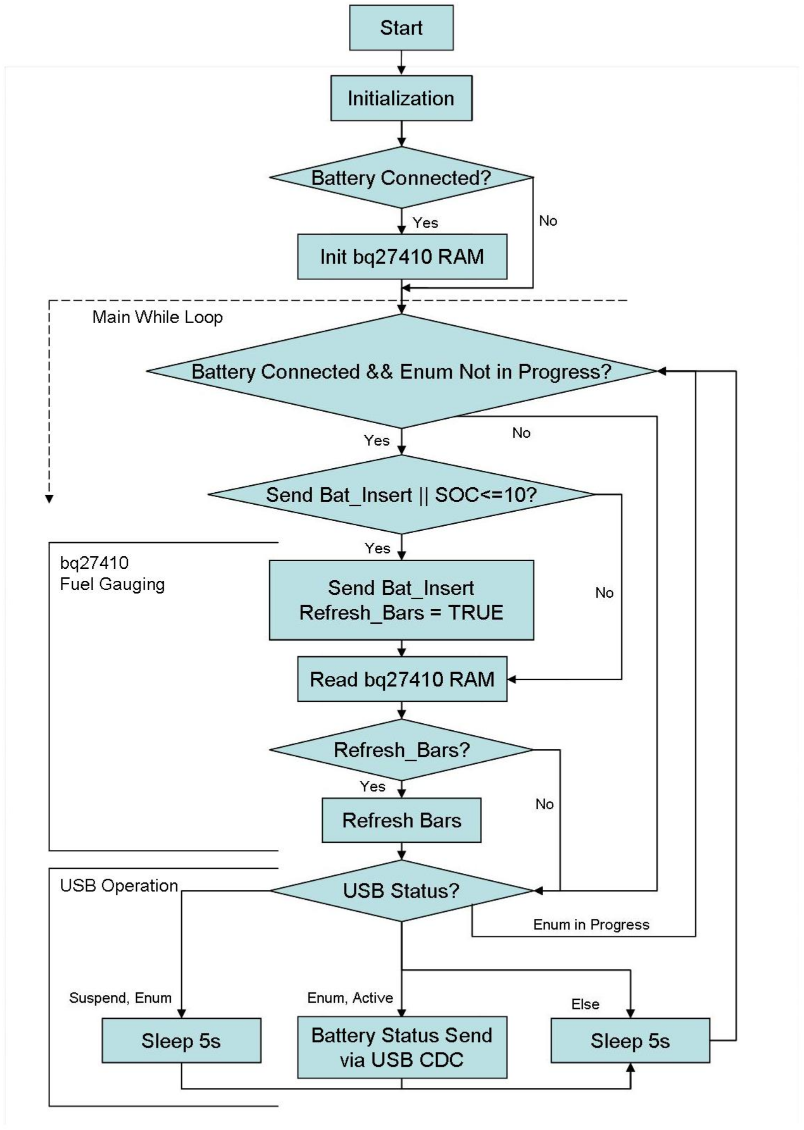
Figure 5 shows a high-level flow chart of the demo application.



NOTE: See Table 3 for the "Appropriate USB Actions" for each of the different USB configurations.

Figure 5. MSP430-Based Battery Charging and Gauging Demo - High-level Flow Chart

Figure 6 is a more detailed flowchart of the demo application, delving into the finer details of the demo firmware.

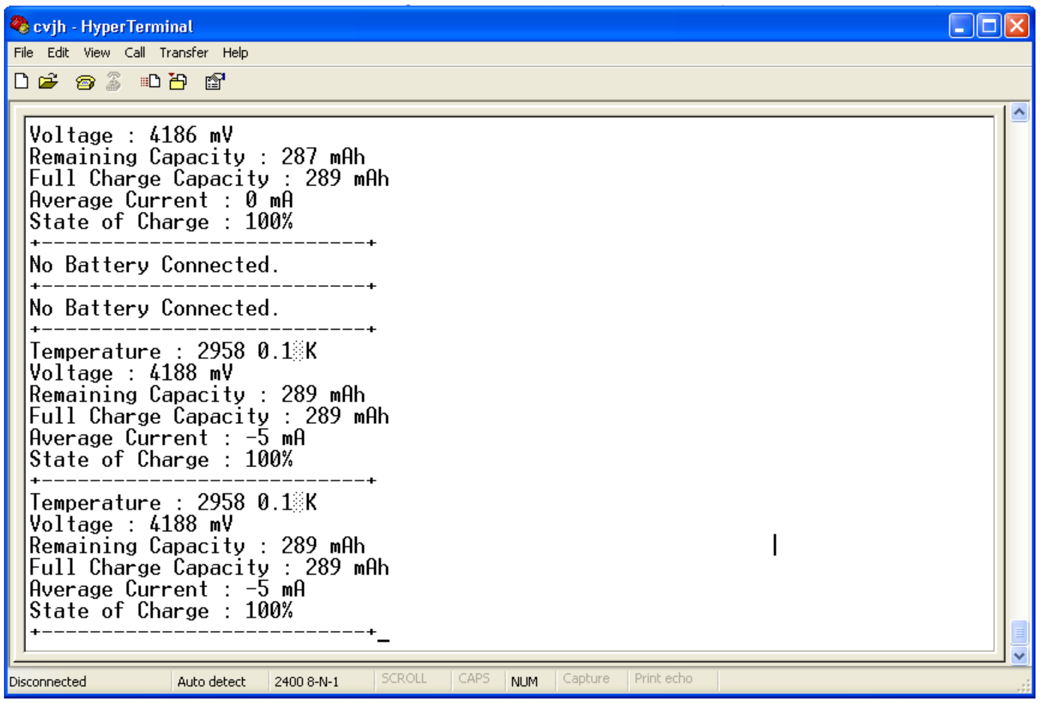


NOTE: When using the demo example, ensure that a battery is attached before power up to ensure that the system is initialized properly.

Figure 6. MSP430-Based Battery Charging and Gauging Demo - Detailed Flow Chart

5.4 Demo Application Output

Figure 7 is a screenshot of the PC software (HyperTerminal) while the demo application is running. When a battery is present in the system, the HyperTerminal window displays the bq27410 RAM values every five seconds. When a battery is removed from the system, the phrase "No Battery Connected" is displayed.



```
Voltage : 4186 mV
Remaining Capacity : 287 mAh
Full Charge Capacity : 289 mAh
Average Current : 0 mA
State of Charge : 100%
+---------------------------+
No Battery Connected.
+---------------------------+
No Battery Connected.
+---------------------------+
Temperature : 2958 0.1░K
Voltage : 4188 mV
Remaining Capacity : 289 mAh
Full Charge Capacity : 289 mAh
Average Current : -5 mA
State of Charge : 100%
+---------------------------+
Temperature : 2958 0.1░K
Voltage : 4188 mV
Remaining Capacity : 289 mAh
Full Charge Capacity : 289 mAh
Average Current : -5 mA
State of Charge : 100%
+---------------------------+_
```

Figure 7. MSP430-Based Battery Charging and Gauging Demo - HyperTerminal Screenshot

5.5 *Power Consumption*

Table 4 shows the current consumption for the system as a whole while using USB power.

Table 4. Power Consumption

Parameter	Battery Charging Mode	Battery Fully Charged or bq24230 Termination Mode	USB Suspend OR Battery Discharging Mode
Current on VBUS	36.5 mA	12 mA	405 µA
Current at Charger	96.9 mA	48 µA	46.4 µ A
Current at Buck-Boost	30 µA	30 µA	30 µ A
VOUT Charger	4.4 V	4.4 V	V_{BAT}
VOUT Buck-Boost	3.3 V	3.3 V	3.3 V

NOTE: When in Battery Charging Mode, the bq24230 is in USB100 mode. Up to 100-mA charging current may be drawn when charging.

6 References

[1] Tenergy 3.7V 300mAh Lithium-Ion Polymer Battery (http://www.tenergy.com/Tenergy-Lithium-Ion-Polymer-3-7V-300mAh-561540-Rechargeable-Battery)

[2] bq24230 Battery Charger (SLUS821)

[3] TPS63030 3.3-V Buck-Boost Converter (SLVS696)

[4] bq27410 Fuel Gauge (SLUSAF4)

[5] MSP430 USB Developers Package (http://www.ti.com/tool/msp430usbdevpack)

Appendix A Associated Code Files

Table 5 shows which files are required for each project that is included with this application report. Each project's function and application use is also discussed.

Table 5. Associated Code Files

C Source Files and Folders	Comments
Production Code	"bq27410 Flash Configuration at Production" project folder MSP430_bq27410_prod.c – handles the bq27410 flash configuration at production FuelGauge_API.c and .h, HAL_FuelGauge_ProductionCode.c and .h – library of bq27410 flash read/write commands
Demo Application	"LiPo Charging/Fuel Gauging Demo Application" project folder MSP430_bq27410_CDC.c – application level firmware that manages the USB, charger, fuel gauge interfaces FuelGauge_API, HAL_FuelGauge.c and .h – library of bq27410 RAM read/write commands F5xx_F6xx_Core_Lib, USB_API, USB_config, usbConstructs.c and .h, usbEventHandling.c – USB stack referenced files and folders
Folders: driverlib, inc	driverlib and inc folder contains all the driverlib .c and .h files used FuelGauge_API, HAL_FuelGauge.c and .h and the MSP430_bq27410_prod.c files Driverlib files are used to manage the low-level I2C communication with the fuel gauge, clock, PMM configuration, TLV data access, and so on.
To use the latest Driverlib version	If using the associated application firmware as is, there is no need to include the driverlib files separately. However, it is recommended to use the latest driverlib files. **NOTE:** Only MSP430Ware version v1.10.00.00 or later works with the associated application firmware. To include the latest driver lib functions into the projects: 1. Download the MSP430Ware as a standalone package (http://www.ti.com/tool/msp430ware) and replace the "driverlib" and "inc" folders in the project path with the ones from the MSP430Ware folder. OR Update the MSP430Ware on the IDE by checking for updates (in CCS, click Help > Check for Updates). 2. Clean the project and Rebuild.

Appendix B Demo Hardware Schematic

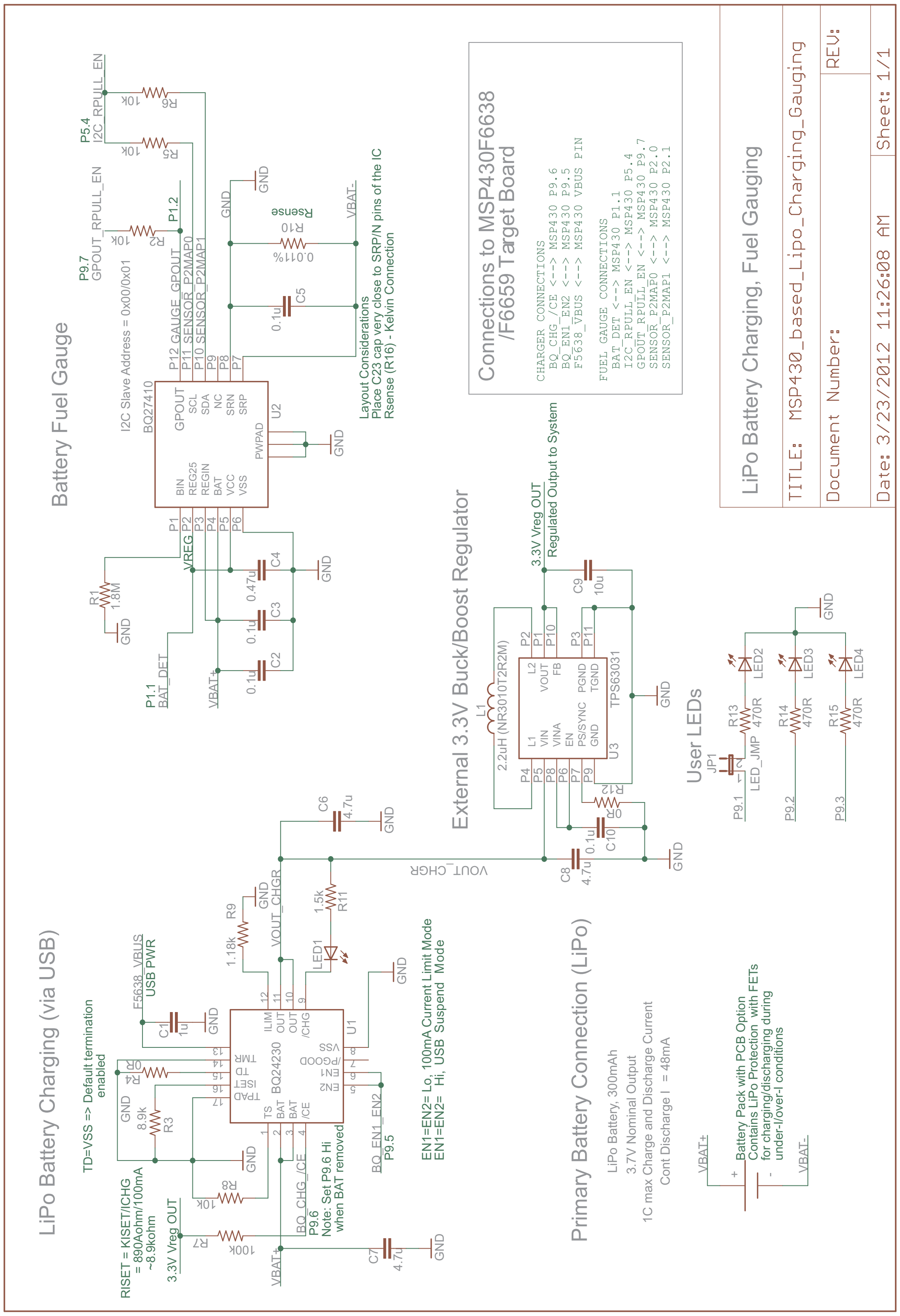

Battery Fuel Gauge
BQ27410
U2
I2C Slave Address = 0x00/0x01
P9.7 GPOUT_RPULL_EN
P5.4 I2C_RPULL_EN
P1.2
R5 10K
R6 10K
R2 10K
R10 0.011% Rsense
C5 0.1u
VBAT-
GND
Layout Considerations
Place C23 cap very close to SRP/N pins of the IC
Rsense (R16) - Kelvin Connection
R1 1.8M
C4 0.47u
C3 0.1u
C2 0.1u
P1.1 BAT_DET
VBAT+
VREG
Connections to MSP430F6638 /F6659 Target Board
CHARGER CONNECTIONS
BQ_CHG_/CE <--> MSP430 P9.6
BQ_EN1_EN2 <--> MSP430 P9.5
F5638_VBUS <--> MSP430 VBUS PIN
FUEL GAUGE CONNECTIONS
BAT_DET <--> MSP430 P1.1
I2C_RPULL_EN <--> MSP430 P5.4
GPOUT_RPULL_EN <--> MSP430 P9.7
SENSOR_P2MAP0 <--> MSP430 P2.0
SENSOR_P2MAP1 <--> MSP430 P2.1
External 3.3V Buck/Boost Regulator
L1 2.2uH (NR3010T2R2M)
TPS63031
U3
3.3V Vreg OUT
Regulated Output to System
C9 10u
C10 0.1u
C8 4.7u
R12 0R
VOUT_CHGR
User LEDs
JP1 LED_JMP
R13 470R
R14 470R
R15 470R
LED2
LED3
LED4
P9.1
P9.2
P9.3
LiPo Battery Charging (via USB)
BQ24230
U1
TD=VSS => Default termination enabled
RISET = KISET/ICHG = 890Aohm/100mA ~8.9kohm
F5638_VBUS USB PWR
C1 1u
R3 8.9k
R4 0R
R8 10K
R7 100K
3.3V Vreg OUT
R9 1.18k
R11 1.5k
LED1
C6 4.7u
C7 4.7u
BQ_CHG_/CE P9.6
Note: Set P9.6 Hi when BAT removed
BQ_EN1_EN2 P9.5
EN1=EN2= Lo, 100mA Current Limit Mode
EN1=EN2= Hi, USB Suspend Mode
Primary Battery Connection (LiPo)
LiPo Battery, 300mAh
3.7V Nominal Output
1C max Charge and Discharge Current
Cont Discharge I = 48mA
Battery Pack with PCB Option
Contains LiPo Protection with FETs
for charging/discharging during
under-I/over-I conditions
LiPo Battery Charging, Fuel Gauging
TITLE: MSP430_based_Lipo_Charging_Gauging
Document Number:
REV:
Date: 3/23/2012 11:26:08 AM
Sheet: 1/1

Figure 8. MSP430-Based Battery Charging and Gauging Demo Schematic