The steps to change 18650 cells and reset a battery.

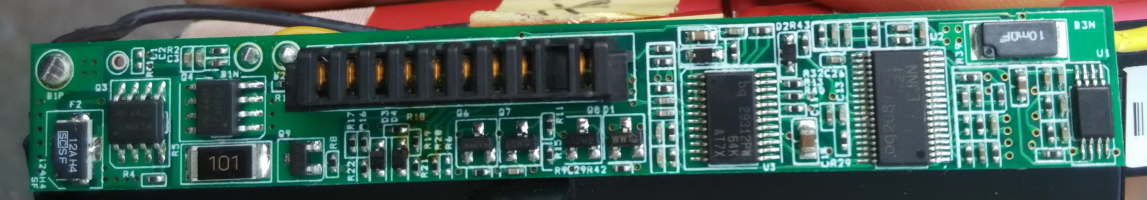
You Xiaojie (yxj790222@163.com)

This article records my steps to change 18650 cells for ECS G401 laptop. The model is EM-G400L2S, 11.1V 4800mAH.

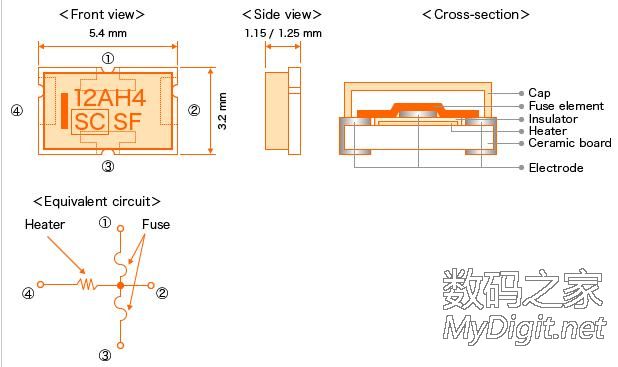
1. Hardware info and connexion

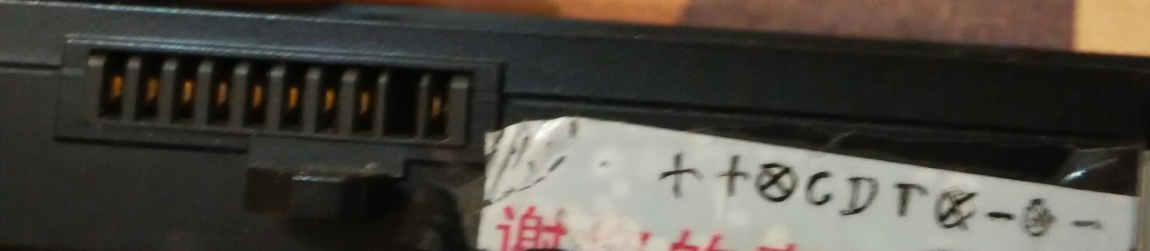
Disassambling batteries, there are 18 clips around. No undamaged methods only force, then we can get chips.

Central controller bq2084drt(u2), MOS driver and cell voltage balance controller bq29312pw(u3) cell voltage monitor bq29401(u1), the datasheet can be downloaded from ti website.



F2 is a 12A fuse,if no output check F2 first. Q3 Q4 are MOSFETs controlling charging/discharging.

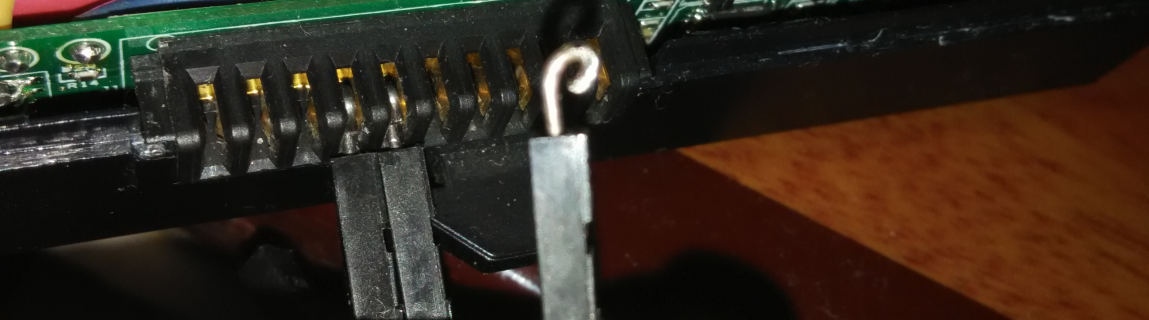




vcc, vcc, nc, scl, sda, temp, nc, gnd, nc, gnd

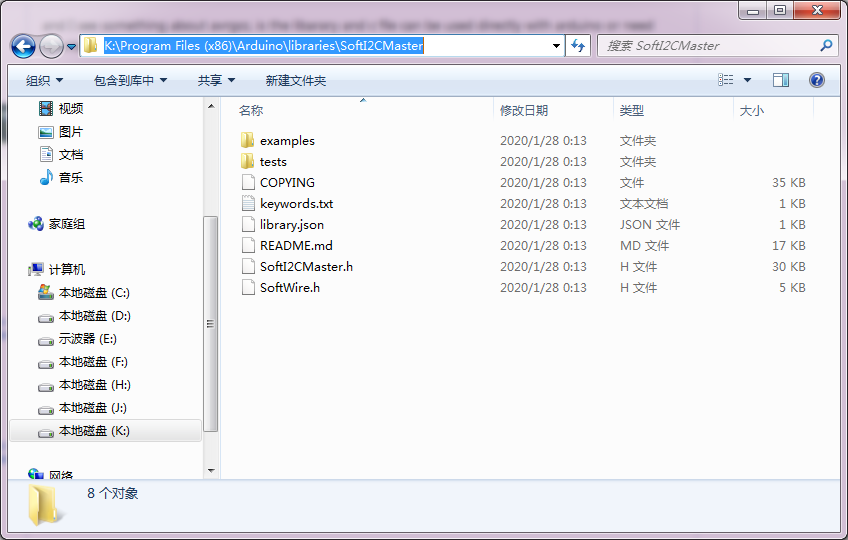
I use Arduino Uno R3 with default SDA,SCL which are identical to A4 A5, other similar atmega 328/168 derivative will be ok, modifying \*.ino file for pin definition.

Connect arduino SDA,SCL and GND with corresponding pin of batteries. Some batteries have SYS\_PRES pin, connected to GND through a resister.





To compile and upload SmartBatteryHack arduino firmware, using official arduino IDE. First download SoftI2CMaster <https://github.com/felias-fogg/SoftI2CMaster> ,Put this library inside the "<Your installed dir>\Arduino\libraries\SoftI2CMaster" folder, for me: K:\Program Files (x86)\Arduino\libraries\SoftI2CMaster. Then compile and upload firmware normally.



Though there is a pre-compiled SmartBatteryHack.exe in \GUI\SmartBatteryHack\SmartBatteryHack\bin\Debug, you can run “c:\Windows\Microsoft.NET\Framework\v3.5\MSBuild SmartBatteryHack2008.sln” to compile GUI. If v3.5 fail, try v4.0 with other \*.sln file.

1. Introduction of register and byte order

The register is the function we called, so we write parameters to register and read the return value. Some register can be read only.

The byte order is the transfer and store sequence for a series of bytes.By default the byte-order is reversed both in the Arduino firmware and the Windows GUI. This is the most common setting in fuel gauge ICs so I went with it. If you want to write 0x2673 into the 0x00 register, then just type 2673 into the GUI and the Arduino will first write 0x73 then 0x26.

1. Checking Seal State and Permanent Failure of batteries.

The batteries are in sealed mode when put it into market. In this mode we can only read a few registers. Firstly to check seal state and permanent fail bit.

Looking for datasheet for seal state, the result is register 0x2f:

* Pack Status and Pack Configuration (0x2f)

This function returns the pack status and pack configuration registers. The pack status register contains a number of status bits relating to bq2084 operation. The pack status register is the **least significant byte of the word**. The pack configuration register reflects how the bq2084 is configured as defined by the value programmed in Pack Configuration in DF 0x28.

The pack status register consists of the following bits:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| PRES | EDV2 | SS | VDQ | AFEFAIL | PF | CVOV | CVUV |

PRES

The PRES bit indicates that the bq2084 has been inserted into the system.

0 bq2084 is out of the system

1 bq2084 is inserted into a system

SS

The SS bit indicates the seal state of the bq2084.

0 The bq2084 is in the unsealed state

1 The bq2084 is in the sealed state

VDQ

The VDQ bit indicates if the present discharge cycle is valid for an FCC update.

0 Discharge cycle not valid

1 Discharge cycle valid

PF

The PF bit indicates that the bq2084 PF Flag has been set. It is cleared only when the PF Flag has been cleared

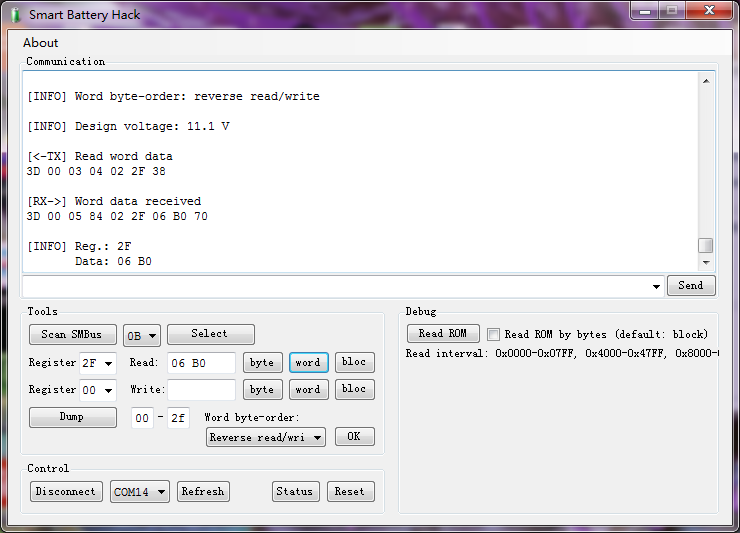
0 bq2084 PF Flag = 0x00

1 bq2084 PF Flag = 0x66

Decode the Battery for Seal State and Permanent Failure:

Read register 0x2f in GUI:

Tools->register: select 2f, read:(readonly, the value read shown here) , click word button, the read value is:06b0



Ok, 0x06b0=0b0000 0110 1011 0000 the **least significant byte of the word** = 1011 0000

So bit5 SS=1 bit2 PF=0

1. Unseal batteries

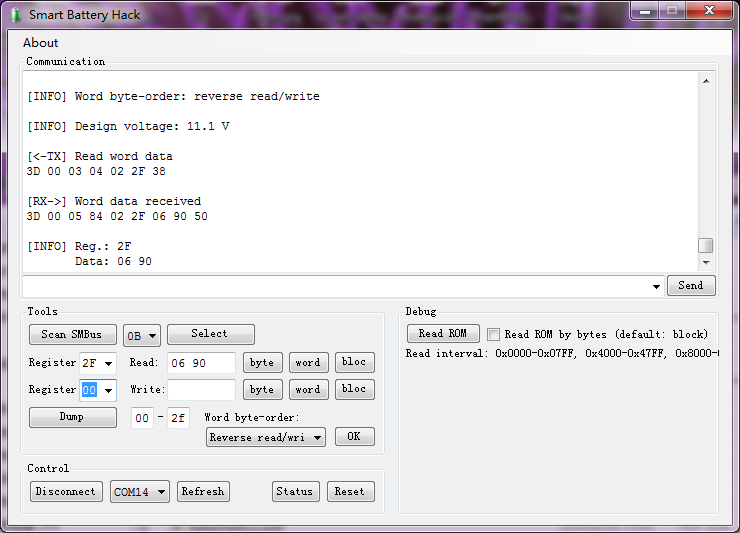
Next we unseal it by unseal key. The default unseal key is 2084 7a43, stored in data flash 0xf8 0xf9 0xf6 0xf7,and another default unseal key collected from internet is 0414 3672. This key can be modified by manufacture, so just for reference.

To using unseal key, we need writing this key to register 0x00 (ManufactureAccess).

Open GUI program in computer, find

Tools->register: select 00, write: entre 2084, click word button,

Tools->register: select 00, write: entre 7a43, click word button,

Verification: 

Read register 2f:

Tools->register: select 2f, read:(readonly, the value read shown here) , click word button, the read value is:0x0690=0b11010010000 the **least significant byte of the word** = 1001 0000, bit 5 is 0 unsealed.

1. Related register to rest batteries

When changing 18650, the following register should be set to reset batteries.

#define FullChargeCapacity 0x10

#define CycleCount 0x17

#define DesignCapacity 0x18

To set these capacity pay attention to unit in mAH.

1. Try to entre Full Access Mode

From somewhere I see under Full Access Mode, we can write modified value to the register directly, but I try FFFF FFFF, failed. Because I don’t know the exactly Full Access Mode key and don’t know how to show which mode I am in, I don’t know whether I am not under the Full Access Mode nor the register is always read only.In a word write to register is not feasible.

I am sure the default Full Access key is a two byte word in the table of Memory Map, pay attention to continuous two lines, named “reserve”, the data maybe the default full access key. If you try out, please tell me.

1. Modifying Data Flash

Because the reg’s value stored in Data Flash, I try to find their data flash address and modified them. In datasheet:

* Cycle Count Initialization

Cycle Count DF 0x0c-0x0d stores the initial value for the CycleCount() function. It should be programmed to 0x0000.

* Battery Pack Capacity and Voltage

Pack capacity in mAh units is stored in Design Capacity, DF 0x32-0x33. In mAh mode, the bq2084-V143 copies Design Capacity to DesignCapacity(). In mWh mode, the bq2084-V143 multiplies Design Capacity by Design Voltage DF 0x04-0x05 to calculate DesignCapacity() scaled to 10 mWh. Design Voltage is stored in mV.

The initial value for Full Charge Capacity, in mAh, is stored in DF 0x36-0x37. Full Charge Capacity is modified over the course of pack usage to reflect cell aging under the particular use conditions. The bq2084-V143 updates Full Charge Capacity in mAh after a capacity learning cycle.

Memory Map

The following table shows the memory map for the data flash. It shows the default programming for the bq2084-V133. The default programming reflects example data for a 3s2p Li-ion battery pack with a 0.02-Ω sense resistor. The data flash must be reprogrammed to meet the requirements of individual applications.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DATA FLASH ADDRESS | | NAME | LI-ION EXAMPLE | DATA | |
| HIGH BYTE | LOW BYTE | MSB | LSB |
| 0x32 | 0x33 | Design Capacity | 7200 mAh | 1c | 20 |
| 0x36 | 0x37 | Full Charge Capacity | 7200 mAh | 1c | 20 |
| 0x0c | 0x0d | Cycle Count | 0 | 00 | 00 |
| 0x38 | 0x39 | Cycle Count Threshold | 5200 mAh | 14 | 50 |

We can read or write a byte of data flash by 0x50 0x51 0x52 command in UNSEAL mode or FULL ACCESS mode.

The command definition:

* Writing to Data Flash With Command 0x50

The user may use Command 0x50 to write individual bytes up through 0xff only. The first parameter is the data byte, the second is the address.

For example, writing a 16-bit integer requires two writes:

lError = WriteSMBusWord(&H50, yDataMS, yAddress)

lError = WriteSMBusWord(&H50, yDataLS, yAddress + 1)

* Reading from data flash with commands 0x51 and 0x52

The user may use Command 0x51 to set the address (up to 0xff only) of the flash byte to read. Then use Command 0x52 to read the byte. When setting the address, the first byte is a dummy. For example, to read an integer:

//set flash address for ms data byte

lError = WriteSMBusWord(&H51, yDummy, yAddress)

//read ms byte

lError = ReadSMBusWord(&H52, yDummy, yDataMS)

//set flash address for ls data byte

lError = WriteSMBusWord(&H51, yDummy, yAddress + 1)

//read ls byte

lError = ReadSMBusWord(&H52, yDummy, yDataLS)

lDataWord = (256 \* CLng(yDataMS)) + yDataLS

**The data value should change to yourselves.**

* Reading from data flash In GUI:

Design Capacity:

Write: 51 0032

Read: 52 -> DC\_HB

Write: 51 0033

Read: 52 -> DC\_LB

Tools->register: select 51, write: entre 0032, click word button,

Tools->register: select 52, read:(readonly, the value read shown here) , click word button, the read value is High Byte of DC word

Tools->register: select 51, write: entre 0033, click word button,

Tools->register: select 52, read:(readonly, the value read shown here) , click word button, the read value is LowByte of DC word

Then you can convert DC\_HB,DC\_LB to decimal.

Here 0x00 is Dummy; 0x32 is Address of data flash. The same to following.

Full Charge Capacity:

Write: 51 0036

Read: 52 -> FCC\_HB

Write: 51 0037

Read: 52 -> FCC\_LB

Cycle Count:

Write: 51 000C

Read: 52 -> CC\_HB

Write: 51 000D

Read: 52 -> CC\_LB

* Writing to data flash In GUI:

Design Capacity:

Write: 50 1C32  
Write: 50 2033

Tools->register: select 50, write: entre 1c32, click word button,

Tools->register: select 50, write: entre 2033, click word button,

Here 0x1c is DataMS, high byte of value; 0x32 is Address of data flash.

Here 0x20 is DataLS, low byte of value; 0x33 is Address of data flash.The same to following.

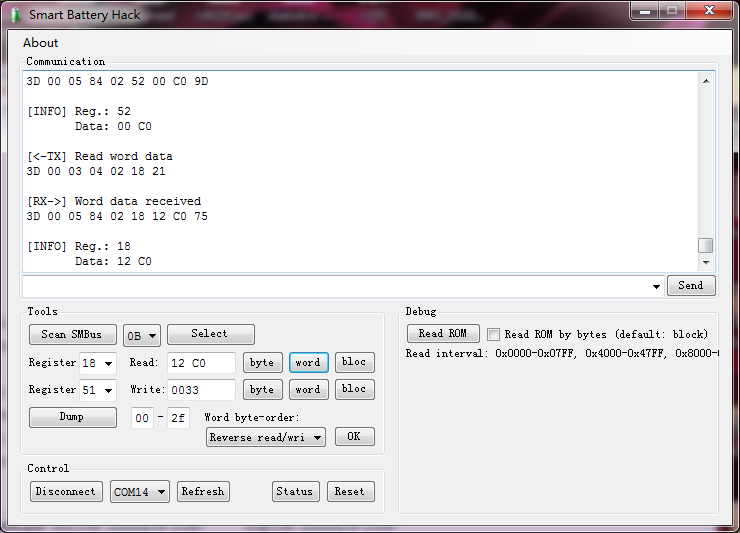
Full Charge Capacity:

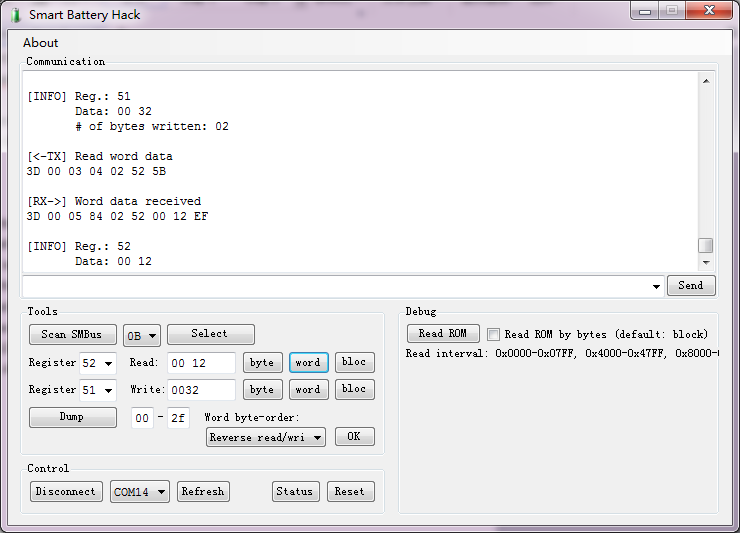
Write: 50 1C36  
Write: 50 2037

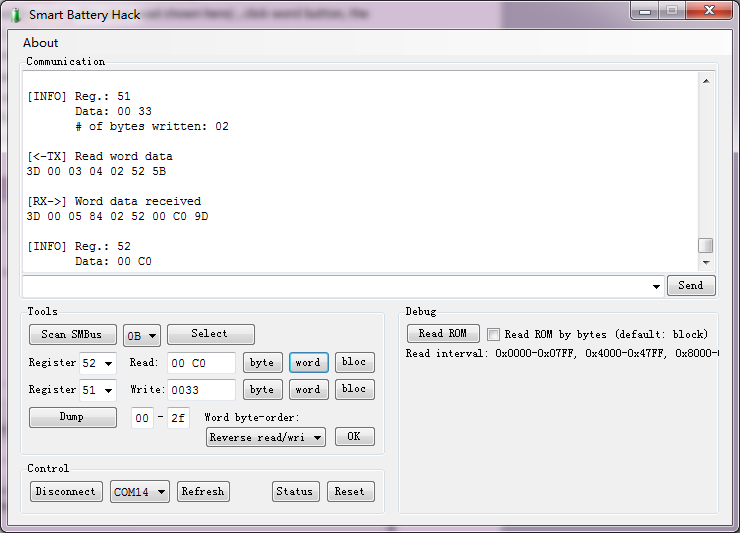
Cycle Count:

Write: 50 000C  
Write: 50 000D

After writing to data flash, we should verify again using 0x51 0x52 command, and then directly read equivalent register, if two methods got the same value, we succeed!

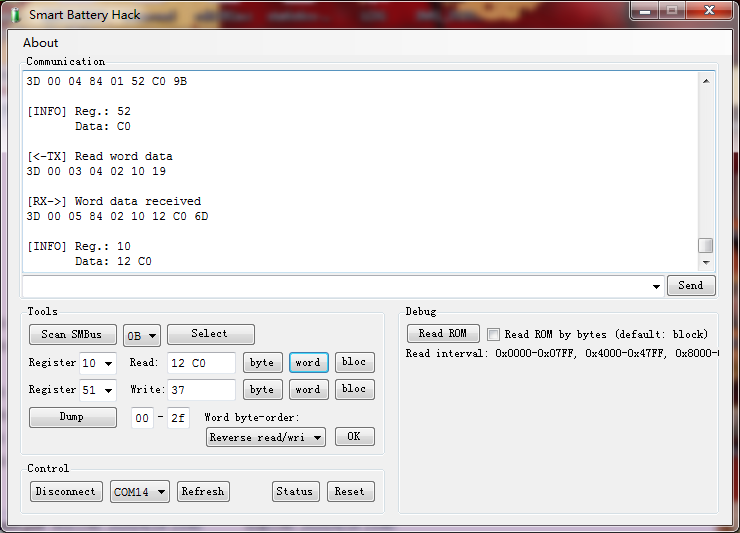


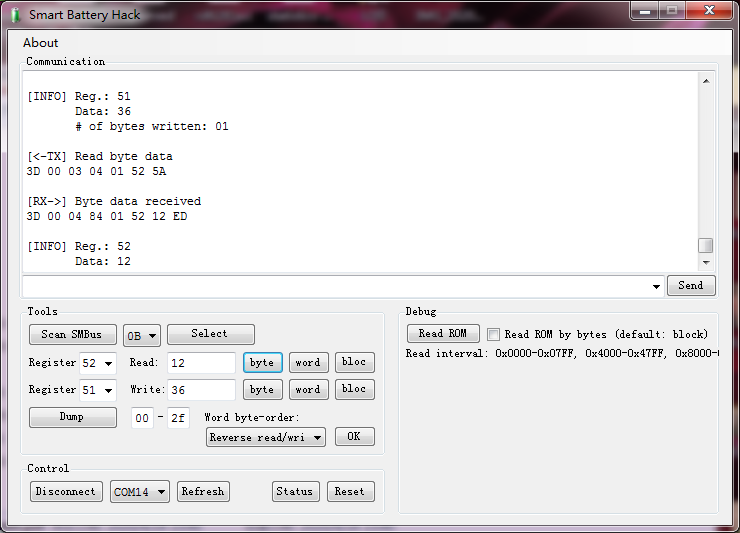


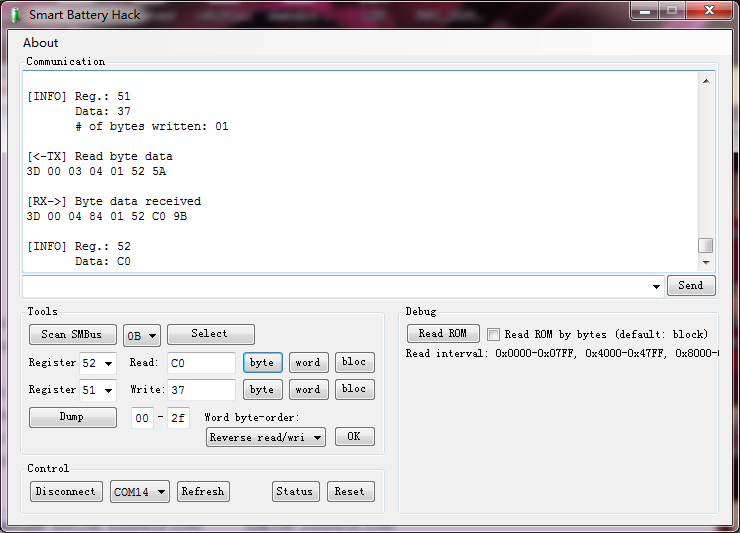


Pic 1: Read reg 0x18: 0x12C0(DC=4800mAH),

Pic 2,3 Value 0x12(DC\_High Byte),from data flash 0x32, 0xC0(DC\_Low Byte) from data flash 0x33

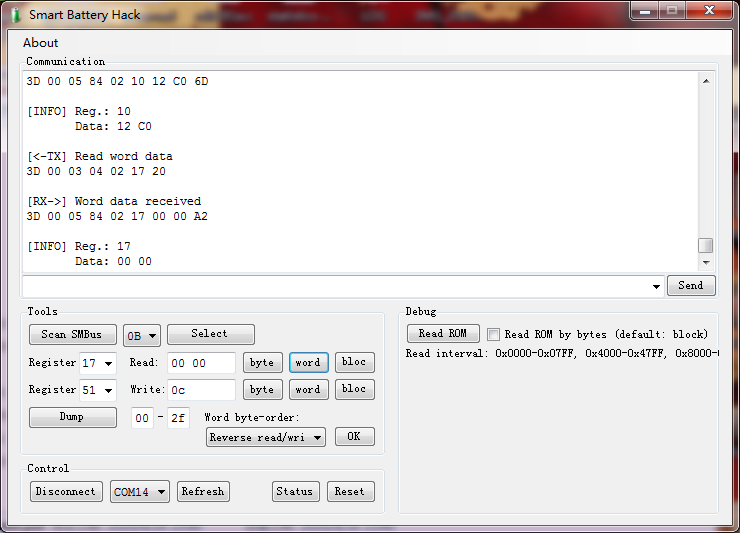


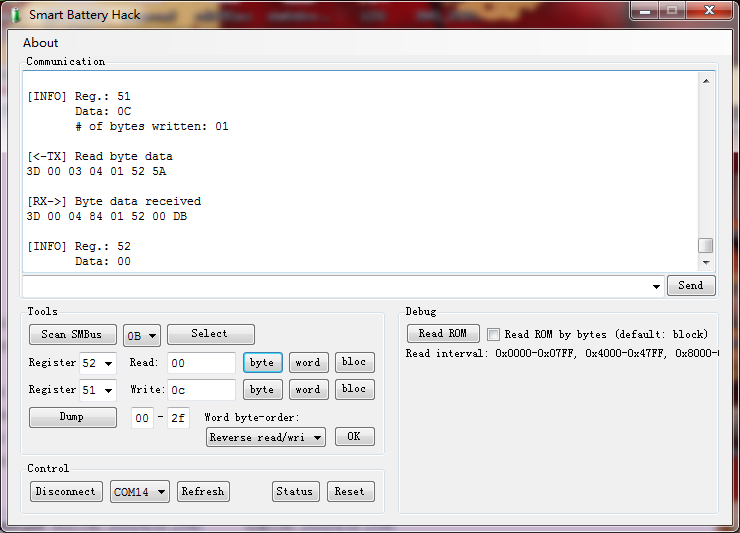


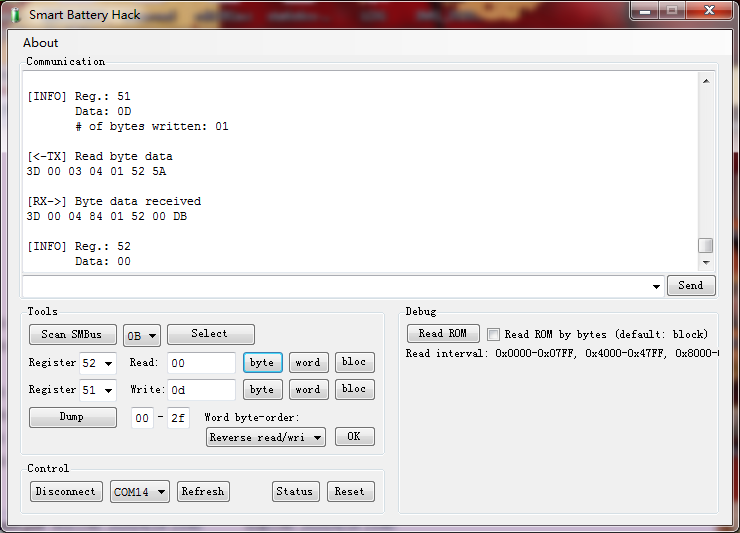


Pic 4: Read reg 0x10: 0x12C0(FCC=4800mAH),

Pic 5,6: Value 0x12(FCC\_High Byte),from data flash 0x36, 0xC0(FCC\_Low Byte) from data flash 0x37







Pic 7: Read reg 0x17: 0x0000(CC=0),

Pic 8,9: Value 0x00(CC\_High Byte),from data flash 0x0c, 0x00(CC\_Low Byte) from data flash 0x0d

1. Something to note

* About melted-fuse and PF.

The fuse controlled by both bq2084 and bq29401.

Table 12. SAFE Activation Conditions (from bq2084 datasheet)

|  |  |
| --- | --- |
| CONDITION | ACTIONS |
| Voltage() > Safety Over Voltage (DF 0x6b and 0x6c), | SOV and PF set |
| Temperature() > Safety Over Temperature in Charge (DF 0x75 and 0x76) | SOTC and PF set |
| Temperature() > Safety Over Temperature in Discharge (DF 0x77 and 0x78) | SOTD and PF set |
| When Miscellaneous Configuration (0x2a) bit 13 AC is set, and the number of Analogue Front End (bq29312) failures has reached the AFE Fail Limit (DF 0xe4) | AFE and PF set |
| When VCELL (MAX) – VCELL (MIN) > Cell Imbalance Threshold (DF 0xee, 0xef) during charging | CIM and PF set |
| If charge FET is off, and charge current greater than FET Fail charge current stored in DF 0x125−0x126 for the time determined by FET Fail time in DF 0x129, or if discharge FET is off and discharge current greater than FET Fail discharge current stored in DF 0x127−0x128 for the time determined by FET Fail time in DF 0x129. | FETF and PF set |
| The PFIN input has detected a low state for PFIN Time in consecutive seconds, as determined by DF 0x120 PFIN, | PF set |

NOTE: SAFE output activated and PF Flag set only if enabled by PF Config (DF 0x0x11f)

In general, bq2084 will melt fuse when over-voltage, over-temperature (over-current), broken-bq29312, Imbalance-cell, Fail-FET.

Bq29401, a cell voltage monitor, will melt the fuse when the cell voltage is abnormal. Its datasheet describe the correct connexion sequences.

To prevent incorrect output activation the following connection sequences must be used. 4-Series Cell Configuration

• VC1(=VDD) → VC2 → VC3 → VC4 → GND or

• GND → VC4 → VC3 → VC2 → VC1(=VDD)

3-Series Cell Configuration

• VC1(=VC2=VDD) → VC3 → VC4 → GND or

• GND → VC4 → VC3 → VC1(=VC2=VDD)

2-Series Cell Configuration

• VC1(=VC2=VC3=VDD) → VC4 → GND or

• GND → VC4 → VC1(=VC2=VC3=VDD)

So when changing 18650 cell, the correct way of dis-assembly is to cut the wires one-by-one going from + to - and when inserting new cells solder the wires - first and go up one-by-one to +. **RULES: ONE BY ONE**

If the fuse is melted and set Permanent Failure, result in without voltage output. The key to clear PF is 2673 1712 stored in data flash 0x123 0x124 0x121 0x122. Try to write them to 0x00 under unseal mode or full access mode.Open GUI program in computer, find

Tools->register: select 00, write: entre 2673, click word button,

Tools->register: select 00, write: entre 1712, click word button,

PS: I did not meet PF situation, so this step I did not test by myself.

* Other commands not shown in 2084’s datasheet.

There are two methods to write and read flash. The simplest method uses commands 0x50, 0x51 and 0x52 as described above. This covers the complete range of the data flash for the bq2083 and bq2085. However, the bq2084 contains many data flash locations outside of this range (above address 0xff). The preferred technique, for both types of devices is to read and write entire blocks. This is generally faster if there are several locations to read and/or write. This technique also allows full programming of the bq2084 data flash constants.

**Reading/Writing Data Flash With the SMB Block Protocol**

The flash data can be read or written with the following read/write page commands which use the SMB Block protocol. In order to write, however, the entire block must first be read, then edited and written back.

* For the bq2083 and bq2085

0x58 64 bytes Reads/Writes data flash locations 0x0000-0x003f

0x59 64 bytes Reads/Writes data flash locations 0x0040-0x007f

0x5a 64 bytes Reads/Writes data flash locations 0x0080-0x00bf

0x5b 42 bytes Reads/Writes data flash locations 0x00c0-0x00e9

Or use:

0x60 16 bytes Reads/Writes data flash locations 0x0000-0x000f

0x61 16 bytes Reads/Writes data flash locations 0x0010-0x001f

0x62 16 bytes Reads/Writes data flash locations 0x0020-0x002f

0x63 16 bytes Reads/Writes data flash locations 0x0030-0x003f

0x64 16 bytes Reads/Writes data flash locations 0x0040-0x004f

0x65 16 bytes Reads/Writes data flash locations 0x0050-0x005f

0x66 16 bytes Reads/Writes data flash locations 0x0060-0x006f

0x67 16 bytes Reads/Writes data flash locations 0x0070-0x007f

0x68 16 bytes Reads/Writes data flash locations 0x0080-0x008f

0x69 16 bytes Reads/Writes data flash locations 0x0090-0x009f

0x6a 16 bytes Reads/Writes data flash locations 0x00a0-0x00af

0x6b 16 bytes Reads/Writes data flash locations 0x00b0-0x00bf

0x6c 16 bytes Reads/Writes data flash locations 0x00c0-0x00cf

0x6d 16 bytes Reads/Writes data flash locations 0x00d0-0x00df

0x6e 10 bytes Reads/Writes data flash locations 0x00e0-0x00e9

* For the bq2084

0x58 64 bytes Reads/Writes data flash locations 0x0000-0x003f

0x59 64 bytes Reads/Writes data flash locations 0x0040-0x007f

0x5a 64 bytes Reads/Writes data flash locations 0x0080-0x00bf

0x5b 64 bytes Reads/Writes data flash locations 0x00c0-0x00ff

0x5c 52 bytes Reads/Writes data flash locations 0x0100-0x133

Or use:

0x60 16 bytes Reads/Writes data flash locations 0x0000-0x000f

0x61 16 bytes Reads/Writes data flash locations 0x0010-0x001f

0x62 16 bytes Reads/Writes data flash locations 0x0020-0x002f

0x63 16 bytes Reads/Writes data flash locations 0x0030-0x003f

0x64 16 bytes Reads/Writes data flash locations 0x0040-0x004f

0x65 16 bytes Reads/Writes data flash locations 0x0050-0x005f

0x66 16 bytes Reads/Writes data flash locations 0x0060-0x006f

0x67 16 bytes Reads/Writes data flash locations 0x0070-0x007f

0x68 16 bytes Reads/Writes data flash locations 0x0080-0x008f

0x69 16 bytes Reads/Writes data flash locations 0x0090-0x009f

0x6a 16 bytes Reads/Writes data flash locations 0x00a0-0x00af

0x6b 16 bytes Reads/Writes data flash locations 0x00b0-0x00bf

0x6c 16 bytes Reads/Writes data flash locations 0x00c0-0x00cf

0x6d 16 bytes Reads/Writes data flash locations 0x00d0-0x00df

0x6e 16 bytes Reads/Writes data flash locations 0x00e0-0x00ef

0x6f 16 bytes Reads/Writes data flash locations 0x00f0-0x00ff

0x70 16 bytes Reads/Writes data flash locations 0x0100-0x10f

0x71 16 bytes Reads/Writes data flash locations 0x0110-0x011f

0x72 16 bytes Reads/Writes data flash locations 0x0120-0x012f

0x73 4 bytes Reads/Writes data flash locations 0x0130-0x133

PS: I did not test such command by myself.

* Changing of firmware of arduino

Because of different data flash reading command, the reading rom function should be adjust too. Changing such two lines: 40 to 51; 42 to 52 for Byte-by-Bye dump. The last line is for block dump, maybe using block read/write command above, and changing block size in \*.ino files, still in testing, don’t use.

#define SetROMAddress 0x40 -> 0x51

#define PeekROMByte 0x42 -> 0x52

#define PeekROMBlock 0x43 -> no register for block read

* About other chips:

The same register maybe has different function, especially since 0x23, which are extended by manufacture. The 0x00 - 0x22 are standard function in SBS 1.1. This article mainly take bq2084 as example, using for other chips, take a look at datasheet first. Especially 0x50 and 0x58,0x60 data flash read/write command. Furthermore the data flash address has been updated, so be careful.

* Reference:

<http://application-notes.digchip.com/001/1-1474.pdf>

<http://powergood.com.ua/forum/viewtopic.php?t=679>

<http://bbs.mydigit.cn/read.php?tid=2316282>

<https://pzxxw.com/thread-78500-1-1.html>

<http://www.karosium.com/2016/08/smbusb-hacking-smart-batteries.html>

<https://github.com/laszlodaniel/SmartBatteryHack>