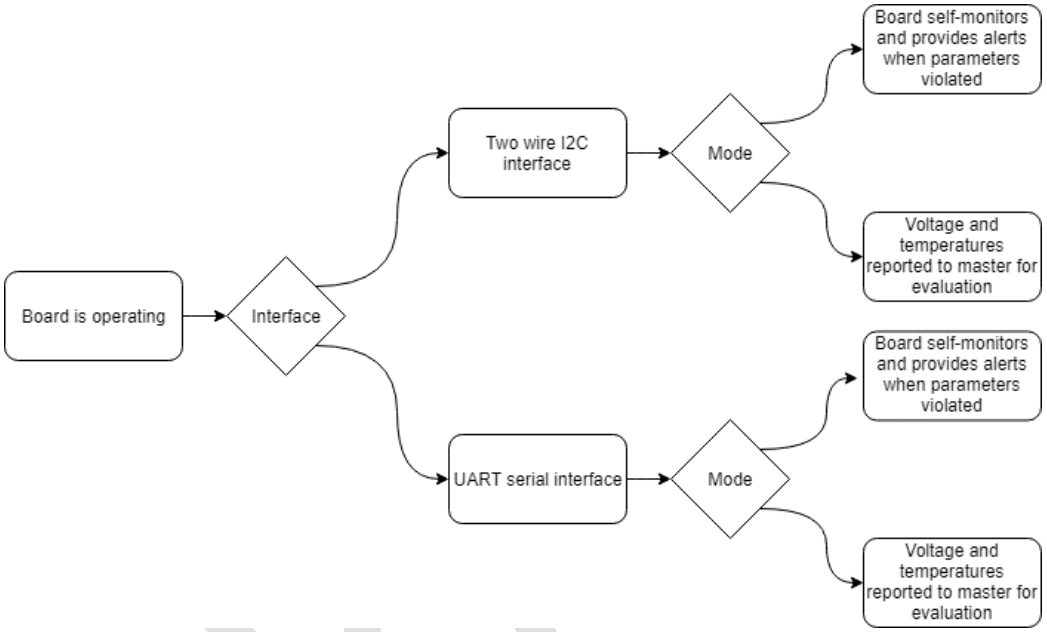
12S BMB Interface Guide

Introduction

This guide will provide information on the process for interfacing with your battery monitoring board. While four-forty BMBs can be used with higher S-count packs through expansion boards, all communication occurs through the primary board. There are two protocols for talking with your BMB, outlined below:



The two communication bus options are I2C and UART serial. I2C is a two-wire master-slave protocol, allowing you to query the BMB "slave" using a "master" microcontroller of your choosing. UART (Universal Asynchronous Receive Transmit) is a protocol that permits direct asynchronous communication between two devices, including PCs.

For initial BMB configuration and computer monitored BMB systems, UART is the recommended option. For microcontroller managed or embedded BMB systems (such as in vehicles), I2C is recommended.

Once the communication bus type has been selected, there are two interface strategies that may be employed to check battery status: **alert monitoring** and **data monitoring**.

Data monitoring is part of the core functionality of the monitor: it allows you to request the voltage of any cell or temperature of any thermistor, and lets you control when to balance cells or perform other actions. It places the responsibility for evaluating the status of the battery on the requestor. This provides extensive information, but at the cost of more time-consuming communications.

Alert monitoring allows the user to leverage the intelligence built into the BMB itself to evaluate the status of the battery. By setting desired parameters for battery operation (max and min voltages and temperatures as well as balancing parameters), the BMB can monitor for unsafe

conditions and throw an alert if a problem is detected. This reduces the burden on the requesting system. The BMB can also toggle a physical I/O line when an alert is thrown.

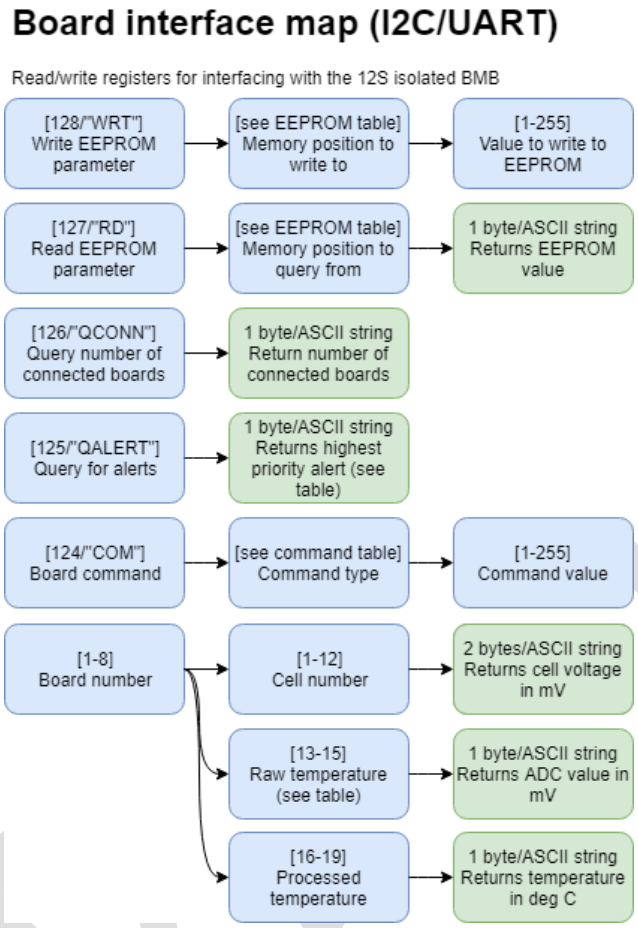
In many cases, a mix of these monitoring strategies will prove most practical. For instance, a user may primarily use the BMB in alert monitoring mode, except for occasionally requesting voltages to determine the approximate state of charge of the battery.

A note on Alert Monitoring

To make use of alert monitoring, you need to provide your unique battery specifications to the BMB. These values are written to EEPROM: non-volatile memory that will be retained even after removing power to the board. **It is essential that you provide your own values for this functionality, as the defaults may not be safe for your battery or implementation**. To discourage relying on alerts without writing new values, new boards will always **default to an alert state** until changes are made. Battery parameters and the parameter read/write process is outlined in following sections.

Interface Procedure

Both I2C and UART interact with the BMB with the same pattern. Serial responses will be returned automatically as soon as processing has completed. Serial commands can be written in upper or lower case, and a string of serial commands must be terminated by a "/n" or a "/r" before the request will be processed (many terminal programs will append these automatically). I2C responses will occur after sending a "read" command to the board with the proper number of bytes requested. Each of these procedures is documented in more detail below.



EEPROM memory positions are listed below:

Table 1 - EEPROM memory positions

Name	Position	Description	Default value
MaxCellVoltage1	1	The maximum allowed cell voltage *before* decimal point [Volts]	4
MaxCellVoltage2	2	The maximum allowed cell voltage *after* the decimal point [centiVolts (1/100 Volt)]	20
MinCellVoltage1	3	The minimum allowed cell voltage *before* decimal point [Volts]	2
MinCellVoltage2	4	The minimum allowed cell voltage *after* the decimal point [centiVolts (1/100 Volt)]	75
Board1SCount	5	S-count of board 1	12
Board2SCount	6	S-count of board 2 (unused if no board connected)	0
Board3SCount	7	S-count of board 3 (unused if no board connected)	0

Board4SCount	8	S-count of board 4 (unused if no board connected)	0
Board5SCount	9	S-count of board 5 (unused if no board connected)	0
Board6SCount	10	S-count of board 6 (unused if no board connected)	0
Board7SCount	11	S-count of board 7 (unused if no board connected)	0
Board8SCount	12	S-count of board 8 (unused if no board connected)	0
MaxExtTemp	13	Maximum temp of external thermistor [degrees Celcius+128]	178
MinExtTemp	14	Minimum temp of external thermistor [degrees Celcius+128]	130
ThermistorLookupTable	15	See thermistor chart	0
MaxDieTemp	16	Maximum temperature of the LTC6802-2 chip before alert is thrown [degrees Celcius+128]	210
WarningDieTemp	17	Maximum temperature of the LTC6802-2 chip before balance is disabled to cool chip [degrees Celcius+128]	188
TargetBalanceDifference	18	Minimum voltage delta-v between bricks that will trigger balancing [mV]	20
MaxBalanceDifference	19	Maximum voltage delta-v allowed between bricks before an alert is thrown [mV]	50
AutoBalance	20	Should system automatically start balancing if TargetBalanceDifference is exceeded? [1=Y,0=N]	0
SerialEnabled	21	Serial communications on/off (WARNING: serial cannot be turned back on over serial connection) [1=Y,0=N]	1
BalanceLED	22	Turns on the iso-LED whenever a board is balancing [1=Y,0=N]	0
EnableExtTemp	23	Enables monitoring of external thermistor/s (see thermistor enable table)	1
EnableAlertLine	24	Enables alert line (5V when alert raised, 0V otherwise) on digital I/O 2 [1=Y,0=N]	0

Board commands are shown below:

Table 2 - Board commands

Name	Position	Description	Default value
EnableBalancing	1	Commands the board to begin balancing if needed [1=Y,0=N]. Ignored if auto-balance is on.	0
ChargeMode [future feature]	2	Changes monitoring parameters to different ranges for charging	0
ChipLED [future feature]	3	Commands the iso-side LED to turn on or off [1=Y,0=N]. Ignored if balance LED is on.	0
EnableDig1	4	Sets voltage on digital I/O 1 to 5V (0V when disabled)	0
EnableDig2	5	Sets voltage on digital I/O 2 to 5V (0V when disabled, ignored if EnableAlertLine is enabled)	0

Alert error codes are provided below:

Table 3 - Alert codes

Name	Code	Priority	Description
No alert	0	-	No alerts raised
CellOvervoltage	1	1	One or more cells exceed the maximum allowed voltage
ExtOvertemp	2	2	One or more external thermistors exceed the maximum allowed voltage
CellUndervoltage	3	3	One or more cells are below the minimum allowed voltage
ExtUndertemp	4	4	One or more external thermistors are below the minimum allowed temperature
UnacceptableImbalance	5	5	Cells are unacceptably unbalanced
DieOvertemp	6	6	Die temperature exceeds upper limit

Temperature addresses are provided below:

Table 4 - Temperature addresses

Address	Description
13	Die temperature [C]
14	Voltage present at thermistor 1 ADC [mV] (currently unvalidated)
15	Voltage present at thermistor 2 ADC [mV] (currently unvalidated)
16	Currently unused
17	Processed thermistor 1 temperature [C]
18	Processed thermistor 2 temperature [C]
19	Currently unused

Thermistor monitor enable commands are provided below

Command	Result
0	No thermistor monitoring enabled
1	Thermistor 1 enabled Thermistor 2 disabled
2	Thermistor 1 disabled Thermistor 2 enabled
3	Both thermistors enabled

I2C interface

The BMB I2C interface is designed to operate with the following settings:

Parameter	Setting
Clock frequency	100kHz
Default board address	0x8
Address length	7 bits (addresses 0-7 unused)
Bus high voltage	3.3-5.0V
Recommended pullup resistance	1.8kOhm (can be jumpered on/off on board)

Below is example Arduino code demonstrating a typical communication between the master and BMB slave. This exchange can be adapted to any I2C compatible microcontroller.

```
#include <Wire.h>
byte numberOfBoards = 0; //Hold the number of boards detected
byte msb;
byte lsb;
void setup() {
    Wire.begin(); // join i2c bus
    Serial.begin(9600);
    numberOfBoards = checkForBoards(); //Read how many boards are detected
}
void loop() {
    for (byte board = 1; board <= numberOfBoards; board++) { //Cycle through each board
        Serial.print('Board ');
        Serial.print(board);
        Serial.println(':'); //Print out which board is being read from
        for (byte cell = 1; cell < 13; cell++) { //Cycle through each cell
            Wire.beginTransmission(8); // transmit to device #8 (BMB default I2C address)
            Wire.write(board); // talk to the first board
            Wire.write(cell); // sends one byte
            Wire.endTransmission(); // stop transmitting
            Wire.requestFrom(8, 2); // Request 2 bytes from slave 8
            while (Wire.available()) {
                msb = Wire.read(); // first byte is most significant
                lsb = Wire.read(); // second byte is least significant
```

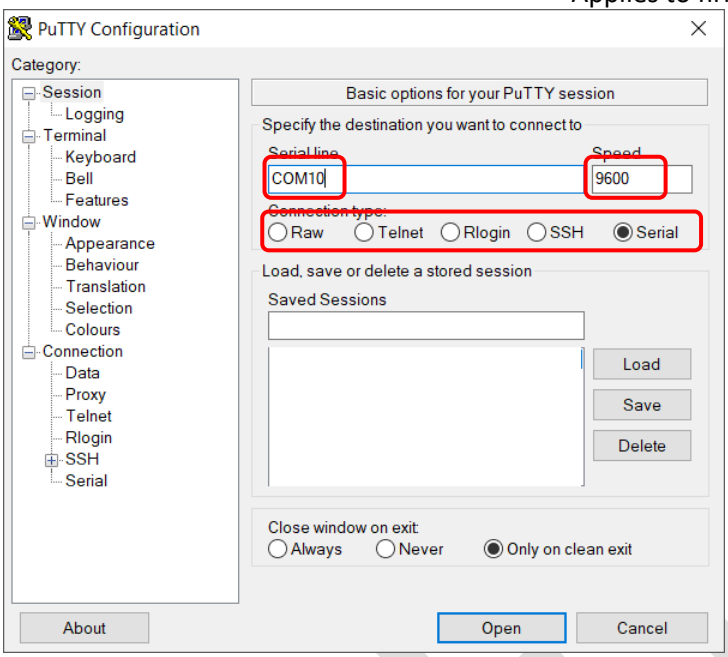
```
            }
            int voltage = (msb << 8) | (lsb); //Voltage in mV, 12bit resolution
            Serial.print('Cell ');
            Serial.print(cell);
            Serial.print(': ');
            Serial.println(voltage); //Print the voltages of each cell on each board
        }
    }
}
byte checkForBoards() {
    byte boardNumber;
    Wire.beginTransmission(8); // transmit to device #8 (BMB default I2C address)
    Wire.write(0x7E); // request number of boards connected
    Wire.endTransmission(); // send out write Command
    Wire.requestFrom(8, 1); // request 1 byte from device #8
    while (Wire.available()) { // continue reading until buffer is empty
        boardNumber = Wire.read(); // read the information
    }
    return boardNumber;
}
```

Serial interface

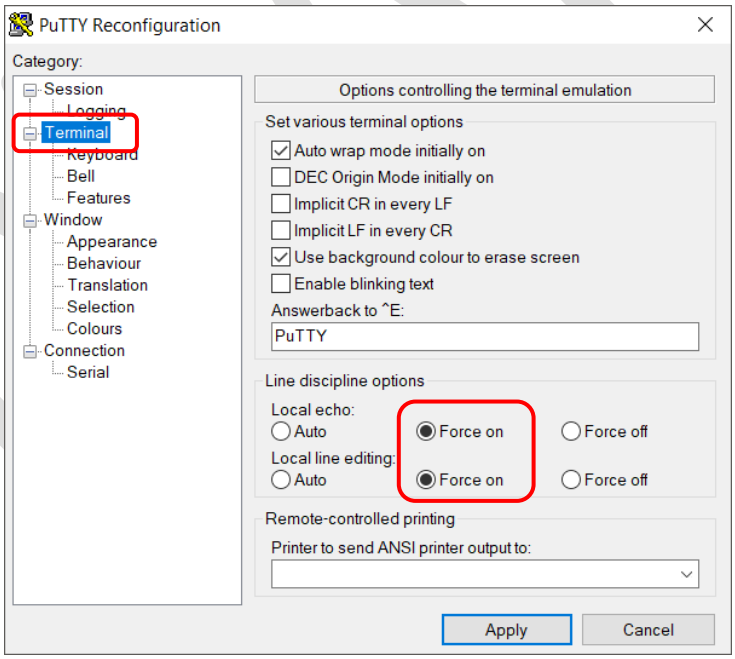
By default, the serial interface operates with the following settings:

Parameter	Setting
Baud rate	9600
Data bits	8
Stop bits	1
Parity	None
Flow control	None or XON/XOFF

Steps for connecting to the BMB using the popular windows terminal program "PuTTY" are presented below. Similar steps could be applied to a different terminal program of your choice:



The program is opened, and the Serial connection type is selected. Serial speed is kept at the default of 9600. The serial line is set to whichever COM port the FTDI programmer has been recognized on (in Windows, you can check this under the "Ports (COM & LPT)" header in Device Manager).



After clicking on the Terminal heading in the Category box on the left-hand side, both "Local discipline options" are changed from the default of "Auto" to "Force on". The "Apply"/"Open" button at the bottom of the window can now be pressed to connect to the board.

Upon successful connection to the BMB, the following lines will be sent on bootup:

```
--four-forty BMB--

support@four-forty.com

Firmware version: [major version].[minor version] [Beta status]

I2C [initialization status]
```

Shortly after this, the BMB will begin operation. It is now ready to accept queries. All valid serial messages will generate a response from the BMB, either with requested information, error message, or a "done" message. This can be used as an indicator of when the BMB has finished processing the last query and is ready for another.