**LyteFyba BMS protocol**

**Introduction**

Any lithium ion battery having more than about four cells, needs a BMS (Battery Management System) which will monitor the cells individually, and keep them within their safe operating area of voltage, temperature and current. The BMS also needs to periodically equalise the state-of-charge of all the cells, also referred to as balancing. The LyteFyba system is designed for both electric vehicles and household electricity supply systems. It has a small circuit board that bolts to the top of each cell (or each group of cells in parallel), called a CMU (Cell Management Unit). The CMU monitors the voltage and temperature of the cell, and can bypass approximately 0.6 amps for the purpose of balancing, whenever the cell's voltage exceeds a threshold. That threshold can be set to correspond to 100% state of charge (SoC) at low charge current, or to a slightly lower state of charge.

The multiple CMUs communicate with a single master unit called the BMU (Battery Management Unit or BMS Master Unit). The BMU also monitors a current shunt that measures the battery current, which is common to all cells, and it controls a contactor that can disconnect the battery to protect the cells if all else fails. It can also control up to four other contactors which can be used for precharge of capacitors in inverters or chargers, and separate control of charge sources, discretionary loads and normal loads (as opposed to essential loads). The BMU can measure the overall battery voltage and the voltage of the PV array or another DC source having a common connection with the battery (up to 400 V).

**System controller**

The BMU has a serial port to communicate with a system controller, such as a Raspberry Pi or Beagle Bone Black. It has another serial port to communicate with an inverter/charger. The BMU can be made to operate in a variety of modes by changing its firmware. New firmware can be loaded into the BMU and CMUs via the system controller serial port on the BMU, by preceding it with a password that is a sequence of control characters that do not occur in normal operation.

In one such mode, communication with the inverter/charger can be delegated to the system controller. In another, the BMU can *be* the system controller and the two serial ports can be used to communicate with a separate inverter and charger if necessary.

**Cell stress**

The simplest battery monitoring systems use a single alarm-wire which is daisy-chained through all of the CMUs so that if any cell goes outside its safe operating area the circuit is broken. This single-bit all-or-nothing approach does not give the rest of the system any warning of impending disconnection and so the LyteFyba system instead uses a four-bit number (0 to 15) to represent the "stress level" of each cell. Stress may be due to under-voltage, over-voltage, under-temperature or over-temperature. Stress levels 7 down to 0 are increasingly safe and levels 8 up to 15 are increasingly unsafe.

At regular intervals, the first CMU sends its stress level to the second CMU which, like all the remaining CMUs, either passes it on or substitutes its own stress level if this is greater. So the stress level that arrives at the BMU a few milliseconds later is the stress level of the most stressed cell. The BMU then passes this on to the system controller, or substitutes a stress level based on the current if this is greater. Stress updates can be configured to be as fast as 16 times per second or as slow as one every 2 seconds.

The CMU to CMU communication is via opto-isolated twisted-pair. Low cost industrial optic fibre can replace the twisted-pair (for safety and noise immunity) when connecting between CMUs on different shelves or in different enclosures. Optic fibre can also be used to communicate from the BMU to the first CMU, and from the last CMU back to the BMU. To communicate with the system controller and the inverter/charger, the BMU uses two RS232 ports with male D9 connectors configured as DTE, with fixed levels on the hardware handshaking pins. Optic fibre can also be used for these two comm ports. The default communications speed is 9600 baud, with no parity and 1 stop bit.

**Status bytes**

The stress level is communicated in binary, in a single status byte that always has its high bit set (bit 7). The stress level is given in binary in the lowest 4 bits of this status byte. Of the remaining 3 bits, bit 6 tells if there has been a communication error and therefore the stress information may not be representative of all cells, and bits 4 and 5 encode partial information about the reason for the stress, namely whether it can be ignored during charge, ignored during discharge, or not ignored at all. A fourth value indicates when all cells are full. This is used by the BMU to reset its coulomb counter to 100%. It can also be used by the system controller to terminate charging. The stress level corresponding to “full” may or may not be the same as the stress level at which bypassing occurs, depending on the specific CMU firmware, or its settings.

Status byte format:

7 6 5 4 3 2 1 0

+---------+---------+---------+---------+---------+---------+---------+---------+

| Always | Comms | Stress type | Stress level (0 to 15 in binary) |

| 1 | error | | |

+---------+---------+---------+---------+---------+---------+---------+---------+

0 is for Treat 00 don't ignore 0 no stress

ASCII as 01 ignore on discharge 7 operating set-point

commands stress 10 ignore on charge 12 alarm stress

and 8 11 all full, and 15 extreme stress

responses ignore on discharge (BMU disconnects battery)

In response to a stress level above 7, the system controller should command the inverter/charger to reduce the magnitude of the battery current (or AC power). It is recommended that the system controller implement a proportional/integral(PI)-control algorithm with the stress as input and the inverter/charger's battery current limit (or AC power limit) as output, and a stress set-point of 7, The system controller should also provide the ability to tune the PI-controller's gain and time-constant for stability, during system commissioning.

If the system controller does not receive any status byte for 10 status times, it should treat this the same as receiving status bytes with the comms error bit set, and feed a stress of 8 to its PI-controller, causing the current to ramp down, ultimately to zero if no stress bytes are received.

During charge, any status byte whose stress type bits are 10 can be treated as having a stress level of 6 or less. During discharge, any status byte whose stress type bits are 01 or 11 can be treated as having a stress level of 6 or less.

The above control algorithm is implemented in the BMU when the BMU is acting as the system controller.

**ASCII Commands and responses**

While status bytes (bytes with the high bit set) are generated regularly and automatically, there is effectively a second independent channel of communication with the BMU and the CMUs, using ASCII commands and responses. Being ASCII, their bytes always have the high bit clear. Each ASCII command, and each response, must be contained in a variable length packet which is terminated by a carriage return (0x0D), symbolised as ⏎ below.

The carriage return is preceded by a two-character error-checking code or CRC (cyclic redundancy check), which is the 12-bit CRC described in <https://github.com/dkeenan7/LyteFyba/blob/master/common/Crc12.s43>.

For convenience when learning or debugging, you can toggle error-checking off (and back on again) with the character sequence “kCB⏎”. This is a “k” command followed by its CRC.

Most commands are single ASCII characters, usually lowercase letters. For example, the v (for voltage) command can be sent as

vWP⏎ or simply v⏎ if error-checking is turned off.

The “WP” is the CRC, and ⏎ is intended to represent the carriage-return character. Assuming there are 16 cells, the response will have the form:

\016:v 3311..⏎

\015:v 3315..⏎

\014:v 3322..⏎

\013:v 3318..⏎

\012:v 3321..⏎

\011:v 3313..⏎

\010:v 3318..⏎

\009:v 3311..⏎

\008:v 3319..⏎

\007:v 3329..⏎

\006:v 3312..⏎

\005:v 3318..⏎

\004:v 3326..⏎

\003:v 3317..⏎

\002:v 3313..⏎

\001:v 3321..⏎

The backslash is the comment character and prevents following CMUs from attempting to interpret the packet as a command. The 3 digits between the backslash and the colon give the ID of the responding CMU in decimal. The lowercase v gives the command that this is a response to, and the 4 digit decimal number gives the voltage of the cell in millivolts. The two periods are standing in for what will, in reality, be an apparently random pair of error-checking characters.

Please note that status bytes (high bit set) may appear anywhere, in the middle of, or between, packets. They do not affect the checksum.

The t (for temperature) command is similar. Its CRC is jA. The resulting temperatures appear as 2 digit decimal numbers in degrees Celsius, possibly preceded by a minus sign (hyphen). As with the CMU ID numbers, leading zeros in results are not suppressed. e.g.

\016:t 53..⏎

\015:t 25..⏎

\014:t 01..⏎

\013:t -05..⏎

...

The g (for gauge) command (CRC `F) is ignored by the CMUs, but the BMU will respond with the depth of discharge in tenths of a percent, based on its coulomb counter. e.g.

\255:g 0247..⏎

The ID of the BMU is always 255. This response represents 24.7% DoD. In other words, 75.3% SoC.

There are many other ASCII commands, but those above are the most important. A brief summary of the others can be found here:

<https://github.com/dkeenan7/LyteFyba/blob/master/common/definitions.txt>

You can learn more about the LyteFyba BMS here:

<http://dkeenan7.github.io/LyteFyba/>

-- Dave Keenan, 2015-08-13, updated 2019-08-17

<http://dkeenan.com>