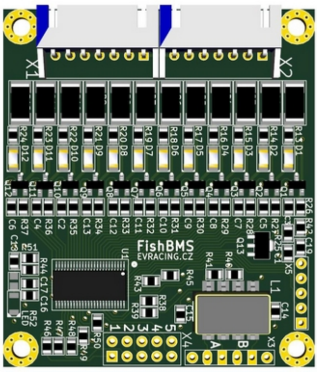
FishBMS – battery management system

documentation



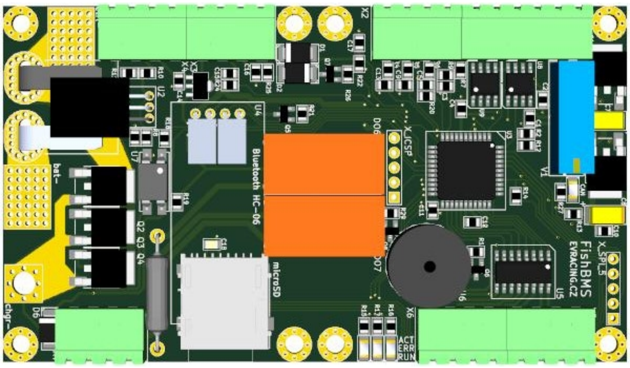


Table of Contents

Last document update: May 11, 2023

Introduction

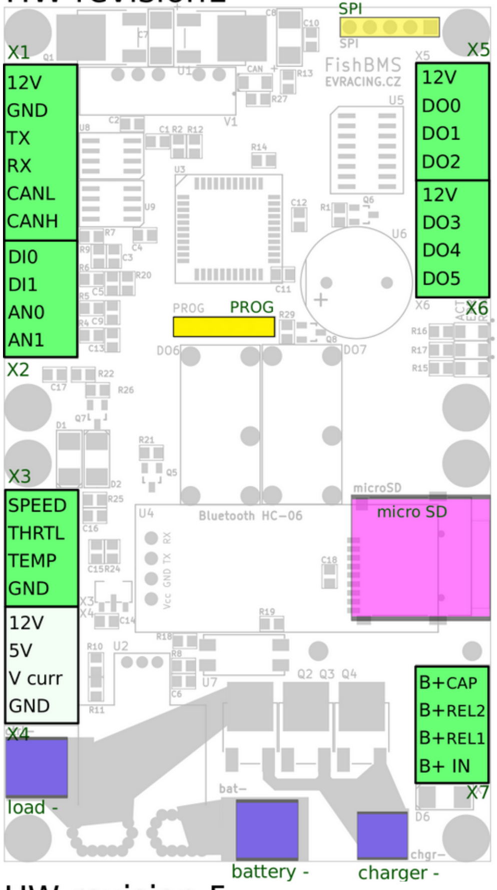
A battery management system (BMS) is any electronic system that manages a rechargeable battery (cell or battery pack), such as by protecting the battery from operating outside its Safe Operating Area, monitoring its state, calculating secondary data, reporting that data, controlling its environment, and balancing it.

BMS Master - parameters		
Hardware	rev1	rev4
Dimensions	107 x 61 x 10 [mm]	107 x 61 x 10 [mm]
Power supply (no galvanic isolation) for battery pack with voltage higher than 95V we recommend external DC/DC to 12V	10 - 14V	30 - 95V (X1 V_in) 10 - 14V (X1 12V) other option: 10 - 100V (with mod)
Hardware sleep feature	no	yes
KTY temperature sensor (motor)	1k	1k
UART speed (no galvanic isolation)	115200 bps	115200 bps
CAN bus speed (isolated), optionable	500 kbps	500 kbps
Consumption	1W	0.3W
Consumption with BT and CAN module	1.5W	0.5W
Consumption in suspend mode	-	< 0.03W

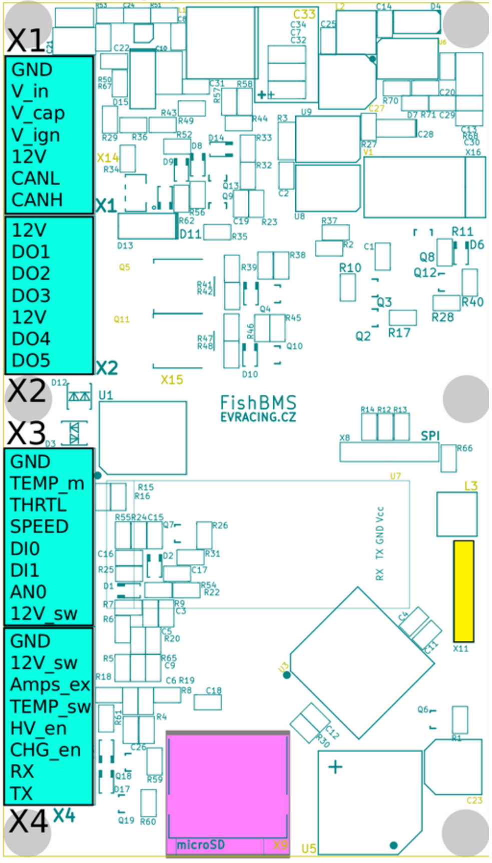
Cell module - parameters	
Dimensions	53 x 61 x 10 [mm]
NTC thermistor	10k, beta 3900K

BMS Master board description

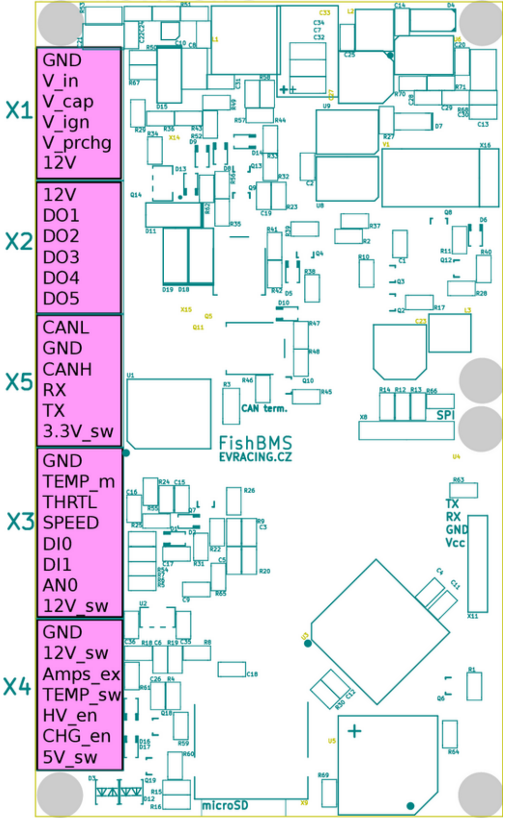
HW revision1



HW revision 4



HW revision 5



X1 connector (Power, UART, optional CANbus)		
pin	name	description
12V	Power +	
GND	Power GND	
TX*	Data transceive	
RX*	Data receive	
CANL	CAN low	
CANH	CAN high	

* to use external USB to serial adapter it will be necessary to take out the internal HC-06 bluetooth module!

X2 connector (2x digital in, 2x analog in)		
pin	name	description
DI0		
DI1		
AN0		
AN1		

X3 connector (speed, power derate, temperature)		
pin	name	description
SPEED	Speed input	Connect motor hall sensor signal or small relay switch
THRTL	PWM output	
TEMP	Temperature input	By default KTY sensor
GND	Ground	Ground reference (same as battery minus !)

X4 connector (optional external current sensor)		
pin	name	description
12V	Power output	
5V	Power output	
V curr	Signal output	
GND	Ground ref.	

X5 + X6 connector (6x digital output – relay driver)		
pin	name	description
12V	Power output	
DO0		
DO1		
DO2		
12V		
DO3		
DO4		
DO5		

What to do after after first power on (configuration)

Each time any configuration is changed (writing 4000+ registers), BMS will check settings for possible conflicts and values out of range. If a conflict / out of range values is detected, error flag “config fail” will be set and value will be changed to fit within range.

If you do not have any configuration prepared (copied from other system) and this is your first setup we recommend to begin with default settings:

- write 5001 = 28730

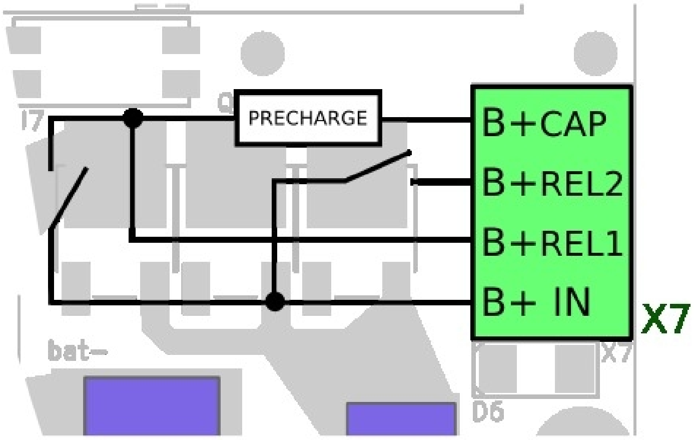
and then start to make changes. Factory values are:

```
   CONFIGLAYER.maxCellVoltage = 3700;     //3700mV
   CONFIGLAYER.warningMargin  = 300;      //300mV
   CONFIGLAYER.minCellVoltage = 2900;     //2900mV
   CONFIGLAYER.tempHighLimit = 125;       //85 degrees
   CONFIGLAYER.tempLowLimit  = 40;        //0 degrees
   CONFIGLAYER.balancingDelta = 50;       //mV
   CONFIGLAYER.currentSensorSlope = 660;  //current sensor range
   CONFIGLAYER.currentSensorOffset = 1650;//1.65V, offset
   CONFIGLAYER.wheelCircumference = 2000; //200mm
   CONFIGLAYER.capacity = 1000;           //100Ah
   CONFIGLAYER.numPoles = 100;            //number of poles
   CONFIGLAYER.ktyType = 0;               //disable motor temperature check
   CONFIGLAYER.ntcBeta = 3800;            //
   CONFIGLAYER.outputMode.w = 0;          //
   CONFIGLAYER.outputFlags.w = 0;         //
   CONFIGLAYER.cfg1.w = 0;                //
   CONFIGLAYER.prechargeResistorDivider = 3125;
```

Cell configuration + temps configuration = all off

Other values will be set to recommended values (power off timeout, CPU reference voltage, CAN id disable, etc).

X7 connector (precharge, direct relay output)		
pin	name	description
B+ cap	precharge	
B+ rel2	2nd relay NO	
B+ rel1	1st relay NO	
B+ in	common	Common contact for both relays



Internal connection of REL1 and REL2 with precharge resistor

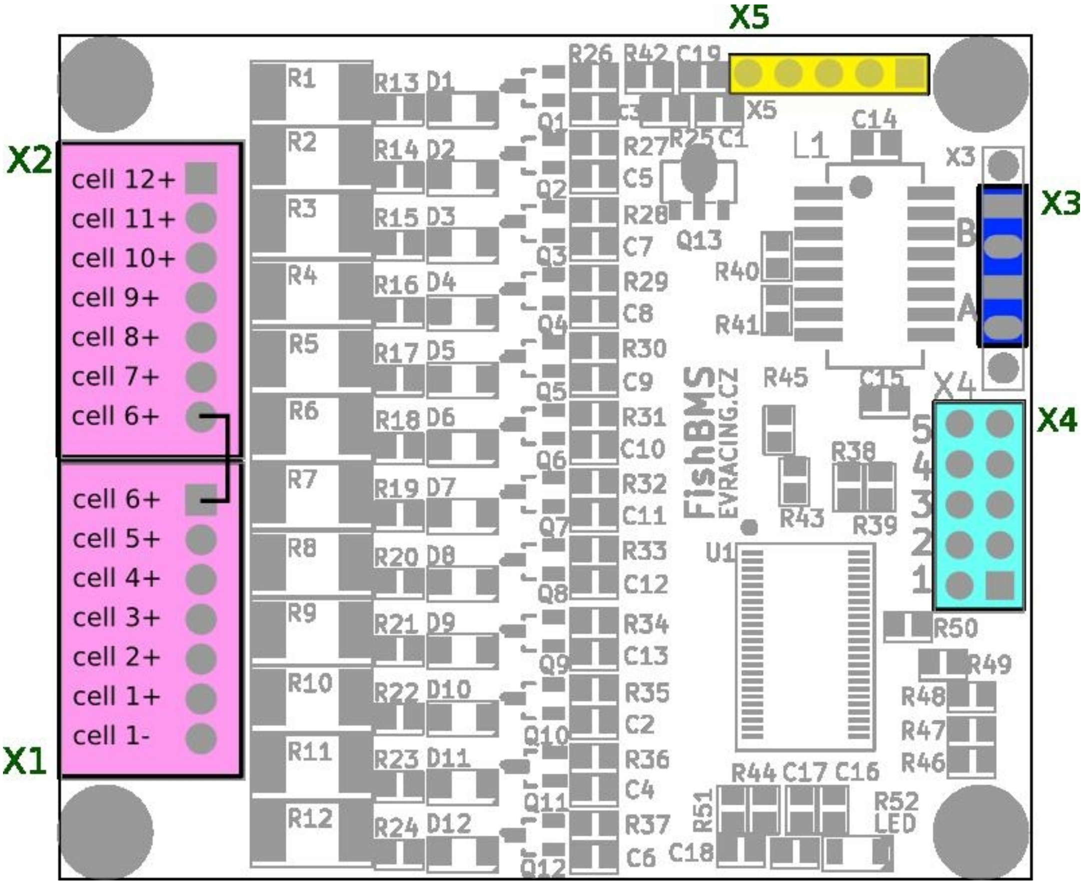
X7 connector is used for precharge and ignition switch by default. It means that REL1 will be switched on before switching the main contactor in order to precharge controller's capacitors and will be switched off after the main contactor closes.

Precharge event is controlled with voltage feedback. It means that pin B+ CAP must rise at least to 90% of battery voltage (measured by cell sum). For configuration of this feature please check registers 4035 and 4043 (added in FW rev 10).

REL2 output is used as ignition switch. It will turn on after 1 second after successful precharge.

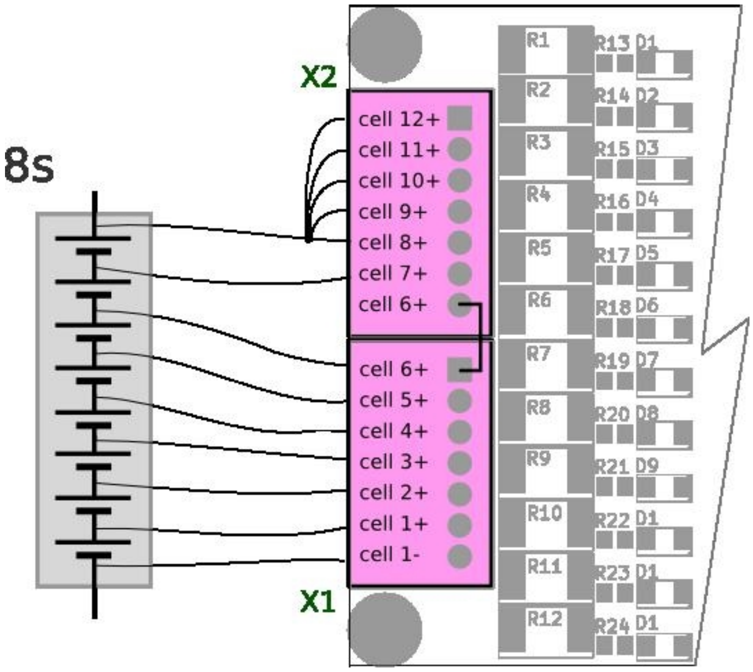
PROG connector (easily accessible from the bottom of the board)		
pin	name	description
1		PIN has rectangular shape
2		
3		
4		
5		

Measurement board description

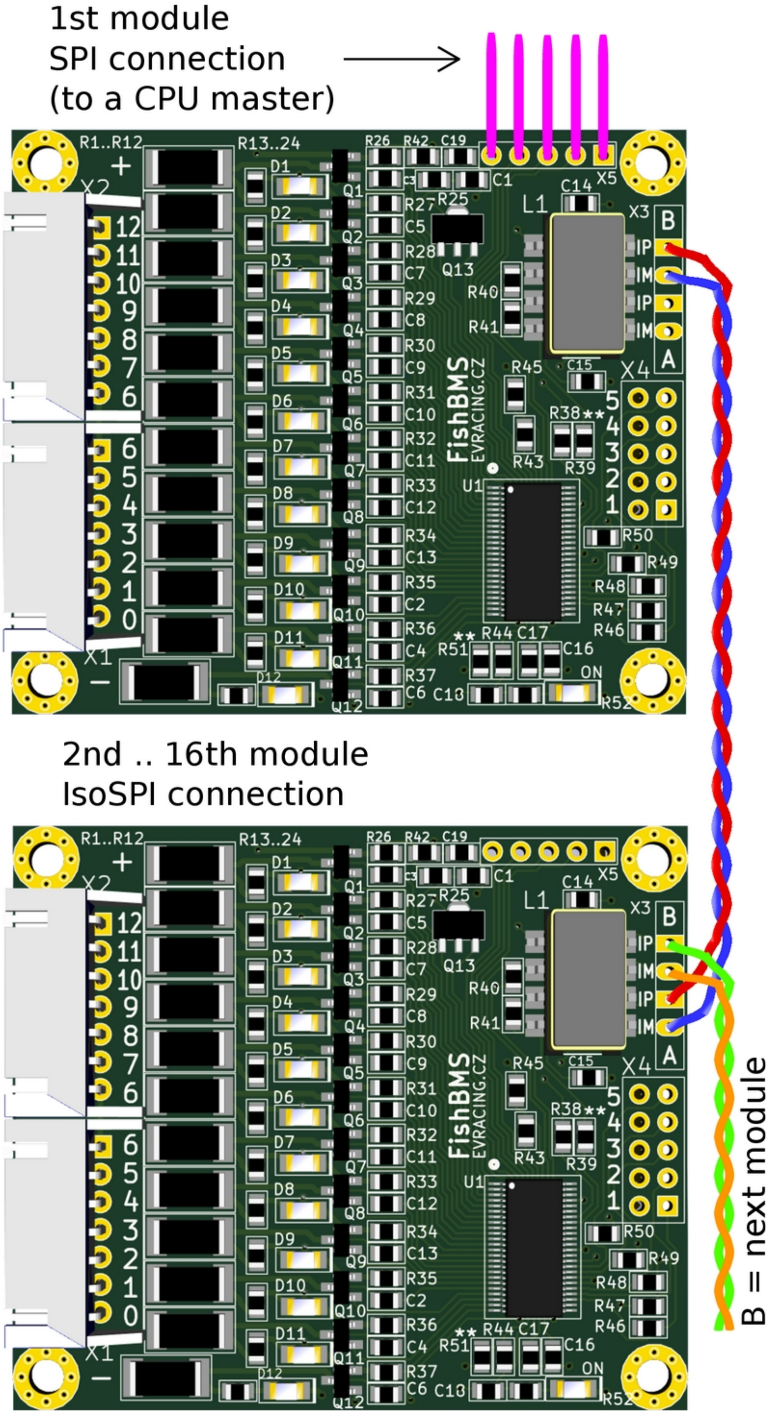


Cell connection example

Unused cells need to be shorted to the highest cell in the module! (in order to power the measurement module correctly).



Module interconnection



1st module
SPI connection
(to a CPU master)

2nd .. 16th module
IsoSPI connection

B = next module

FishBMS 24s

Please keep in mind, that the first module is not galvanically isolated from the CPU board and therefore battery minus has to start at module zero!

Module order (hw revision 1)

Order: C, D, A, B

Module order (hw revision 4+)

A, B, C, D

<<add picture>>

Digital outputs description

BMS master board has in total of 10 (6+2+1+1) usable digital outputs + onboard beeper and 3 status LED outputs.

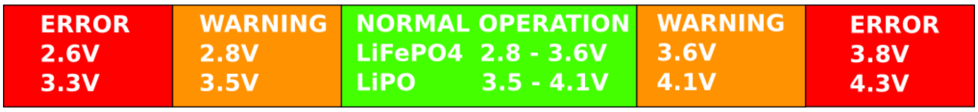
Digital outputs summary		
pin	name	description
DO0	Driver output	bistable relay1, aux output
DO1	Driver output	bistable relay1, aux output
DO2	Driver output	bistable relay2, aux output
DO3	Driver output	bistable relay2, aux output
DO4	Driver output	aux output
DO5	Driver output	aux output
REL1	Relay output	NO relay output with B+ IN common
REL2	Relay output	NO relay output with B+ IN common
CAP	Relay output	NO relay output with B+ IN common with precharge resistor onboard (is using REL2 as an output)
CHARGER-	MOSFET output	Able to switch up to ~20A in one direction (recommended 10 – 15A maximum)
THRTL	PWM output	not implemented yet - throttle override functions (limit power) - analog gauge function (SOC)

BMS safety operation

Software is equipped with basic safety functions including adjustable limits and timeouts. Different outputs can be driven according to these states based on the configuration.

4029	R/W	[mV] maximum cell voltage
4030	R/W	[mV] minimum cell voltage
...		
4033	R/W	[delta mV] warning margin
4034	R/W	[ms] shutdown timeout after error occurs (min 1000 ms)
4035	R/W	Output mode

Table 1: BMS safety configuration



ERROR	WARNING	NORMAL OPERATION	WARNING	ERROR
2.6V	2.8V	LiFePO4 2.8 - 3.6V	3.6V	3.8V
3.3V	3.5V	LiPO 3.5 - 4.1V	4.1V	4.3V

Illustration 1: Example of operation limits

Normal = system runs without any error or warning
Warning = system runs and reports warning (digital output, CAN signal...)
Error = system will turn off soon and reports error (digital output, CAN signal...)

Example:

- set 4030 to 3000 mV, set 4033 to 200, set 4034 to 5000 ms
- when one cell drops below 3200 mV warning bit will be set and you may hear beeper as warning signal every 8 seconds
- appropriate relay driver may be turned on regarding the configuration (mode reg 4035)
- when one cell drops below 3000 mV error bit will be set, and also timeout 5000ms will start to count down
- if cell rises above 3000 mV during the 5000 ms timeout, error bit will be cleared and timeout will reset
- when cell again drops below 3000 mV and stays there for the given timeout (5000 ms), after timeout expires BMS will turn off keyswitch in software and starts trying to turn the high voltage off during next 5000 ms interval. It will turn high voltage off during that timeout only if current drops below 10A. If current does not drop below this value during the timeout, BMS will then turn off the contactor anyway regardless of the current

Temperature warning will be received 5C before the limit (error, check settings in 4017).

How SOC meter works

Essential is current measurement - make sure you set correctly registers 4020 (current sensor slope) and 4019 (zero offset error). Good practice is to compare measured values with DC clamp multimeter (e.g. 0 amps and 100 amps). Then BMS calculates SOC percentage based on current value measured using battery capacity (register 4021) from initial value. Initial value can be set (register 3006) but it is not necessary. If your battery does not perform full cycles this principle of measuring SOC will get inaccurate and you will probably want to use drifting corrections. For this enter SOC open cell voltage table - based on your chemistry and working temperature.

SOC corrections (drifting coefficients, register 4024)

BMS will calculate estimated SOC based on highest cell and lowest cell using an SOC OCV table (registers 4051 to 4061). Lowest cell estimation will be used if SOC < 50% or estimated SOC_low < 15%, otherwise SOC_high estimation is used for corrections. If you do not want to use these correction, just enter 0.

Battery resistance correction 0-15:
0: no correction
1: 0.1mOhm (10A => 0.1mV, 100A => 1mV, e.g. 1000Ah+ battery pack)
2: 0.4mOhm (10A => 0.4mV, 100A => 4mV)
3: 0.9mOhm (10A => 0.9mV, 100A => 9mV)
4: 1.6mOhm (10A => 1.6mV, 100A => 16mV)
5: 2.5mOhm (10A => 2.5mV, 100A => 25mV)
6: 3.6 mOhm (10A => 3.6mV, 100A => 36mV)
7: 4.9 mOhm (10A => 4.9mV, 100A => 49mV)
...
15: 225 mOhm (10A => 22.5mV, 100A => 0.225V)

Drifting speed correction 0-15 (fw17 = slow down 2x vs fw16):
0: no correction
1: 1 < difference < 30% = 0.6% / hour; difference > 30% = 1.2% / hour
...
15: 1 < difference < 30% = 9% / hour; difference > 30% = 18% / hour

Drifting speed acceleration 0-15:
If estimated SOC is higher than 85% or lower than 15%, drifting speed will be multiplied with this coefficient to reach estimated SOC faster. This is nice option for LiFEPO4 cells where you cannot adjust SOC in the middle of the curve (because it leads to very inaccurate estimation) and you correct SOC at the very bottom or very top of the curve.

Reserved: 0-63
Future...

Error and warning description

Error value can be read from the register 3001. Before an error bit is set appropriate warning bit is set (if applicable - e.g. cell undervoltage or overvoltage) in register 3000.

Battery error and warnings:

bit	name	description
0x01	cellVoltHigh	Highest cell voltage exceeded the limit
0x02	cellVoltLow	Lowest cell voltage fell below the limit
0x04	cellTempHigh	Highest cell temperature exceeded the limit (fw 17+)
0x08	cellTempLow	Lowest cell temperature fell under the limit (fw 17+)
0x10	currentOver	Discharge limit exceeded
0x20	currentUnder	Charge limit exceeded
0x40	voltDiff	Cell voltage reaches limits but SOC is too low / high (gets cleared when SOC reaches 40 / 60%
0x80	isolation	
0x0100	communication	Error in communication with slave modules
0x0200	socLow	
0x0400	motorTempOver	Motor temperature exceeded the limit
0x0800	reserved	Switch temperature exceeded the limit
0x1000	reserved	
0x2000	reserved	
0x4000	tempCalc	
0x8000	contactor	

Table 2: Error & warning bits

BMS errors:

bit	name	description
0x01	contactor	
0x02	precharge	precharge procecedure was not successful
0x04	systemFail	
0x08	eepromFail	
0x10	bootFail	
0x20	sdCardFail	SD card is not present or does not work
0x40	configFail	config data were provided in wrong form
0x80	tempCalcMotorFail	not possible to calculate motor temperature correctly
0x0100	tempCalcNTCFail	not possible to calculate NTC temperature correctly
0x0200		
0x0400		
0x0800		
0x1000		
0x2000		
0x4000		
0x8000		

Table 3: BMS errors

Output modes FW rev < 10 (do not use, for reference only)

FishBMS has several outputs and it is possible to configure their behaviour according to desired function (different application - electric bike, solar storage etc.).

Default mode (register 4035 = 0)

Please note that outputs DO2 - DO5 will be updated only if keyswitch is active (high voltage on state). During idle state they will all be turned off.

- DO0 + DO1 = bistable A (as a main battery switch for discharge), **DO0 on**, **DO1 off**
- DO2 = voltage error high (NC), with hysteresis of warning margin
- DO3 = voltage error low (NC), with hysteresis of warning margin
- DO4 = voltage warning high (NO, can be used for “almost full battery light”)
- DO5 = voltage warning low (NO, can be used for “almost empty battery light”)
- REL2 precharge
- REL1 keyswitch feedback with delay 1-2 s (keyswitch B+ for the controller)

Double bistable mode (register 4035 = 1)

Bistable relay A will behave the same way like in the default mode - including precharge REL2 and delayed keyswitch feedback REL1. The only difference is that it will not turn off during cell overvoltage error.

Bistable relay B will behave the same way like onboard MOSFET charger input. It will turn of the relay when cell overvoltage error appears and turn the relay on again when maximum cell drops below warning voltage level.

- DO0 + DO1 = bistable A (discharge switch only - undervoltage)
- DO2 = voltage warning low (NO, can be used for “almost empty battery light”)
- DO3 + DO4 = bistable B (charge switch only - overvoltage)
- DO5 = voltage warning high (NO, can be used for “almost full battery light”)
- REL2 precharge
- REL1 keyswitch feedback with delay 1-2 s (keyswitch B+ for the controller)

Default mode main switch bistable only (register 4036 = 2)

In this mode the bistable relay A is also kept for its default function - battery disconnect. DO2, DO3, DO4, DO5 can be set by user.

Custom mode (register 4036 = 65534)

DO0 - DO5 can be used by user.

Output modes FW rev >= 10 or higher

register address: 4035

Bit#	**sum**	**description**
bit0	1	DO0 main contactor
bit1	2	DO0 + DO1 discharge bistable relay (pulse output), undervoltage protection
bit2	4	DO3 + DO4 charge bistable relay (pulse output), overvoltage protection
bit3	8	DO2 voltage warning low (“almost empty battery signal”)
bit4	16	DO5 voltage warning high (“almost full battery signal”)
bit5	32	DO3 voltage error low (“empty battery signal”)
bit6	64	DO4 voltage error high (“full battery signal”)
bit7	128	0: normally open behavior (bit3 to bit6) 1: normally closed behavior (bit3 to bit6)
bit8	256	DO2 on when high voltage warning, goes off minus margin hyst.
bit9	512	DO0 + DO1 discharge and charge bistable relay, undervoltage + overvoltage protection (pulse outpout), warning margin not necessary for HV relay activation
bit10	1024	DO5 - ON/OFF with hysteresis (e.g. inverter control) **off condition** (remains off minimum 25 seconds = filtration): 1) when low voltage warning occurs immediately **OR** 2) SOC < low SOC (see 4049) **on condition**: 1) lowest cell reaches min cell + 2*margin **AND** 2) SOC > high SOC (see 4049)
bit11	2048	DO3 off when SOC < 10%, DO3 on SOC > 10%
bit12	4096	FW17+ DO1 pulse width SOC signalization (2x blink = battery is off, 1x blink = battery is on, duty cycle is SOC %, period about 3.2s)
bit13	8192	DO4 on when SOC > 90%, DO4 off SOC < 90%
bit14	16384	DO1 + DO2 charge bistable relay (pulse output), overvoltage protection
bit15	32768	FW21+ (DO3+DO4) two color LED 100% / 50% / 30% / 15% (green, green+red, red, blinking red)

Output flags FW rev >= 15

register address: 4037

bit#	**sum**	**description**
bit0	1	0: precharge feedback enabled (wait until capacitor voltage > 80% battery voltage, timeout 5s), percentage can be tuned by changing register 4043 1: precharge feedback disabled (constant precharge time 3s)
bit1	2	IGNITION output 0: turns off together with HV enable bit (or 1s prior to shutdown timeout) 1: turns off when low voltage warning appears (and back on when min cell voltage > 2*margin + configured min cell voltage)
bit2	4	Dynamic limit low voltage 0: off 1: error and warning limits for low voltage will be shifted 0.5V up when no current is flowing. These limits will be dynamically shifted based on current 0-(4048 max discharge current)
bit3	8	
bit4	16	
bit5	32	
bit6	64	
bit7	128	
bit8	256	
bit9	512	
bit10		
bit11		
bit12		
bit13		
bit14		
bit15	32768	power saving mode: 0: normal operation 1: BMS has been suspended due to low cell voltage (charge cells or change this config using SD card)

Input modes FW rev >= 11

register address: 4036

bit#	sum	description
bit0-1	0-3	High voltage control (keyswitch) 0: always off (modbus/CAN control only) 1: autostart (see details “autostart” below) 2: keyswitch on when DI0 enabled 3: push long DI0 → on, push long DI0 → off
bit2-3	0-3	0: reserved 1: reserved 2: reserved 3: reserved
bit4	16	limit power (reg 4042) when DI1 activated (pulled down)
bit5	32	
bit6	64	
bit7	128	
bit8		
bit9		
bit10		
bit11		
bit12		
bit13		
bit14		
bit15	32768	

Autostart

When enabled, high voltage state is switched automatically on after BMS starts. High voltage state (including precharge) will be activated also after 25s timeout, if high voltage state was deactivated by an error (e.g. cell undervoltage). Do not use this option if the battery can stay longer without surveillance or there is nothing which will charge the battery automatically (e.g. solar charger), because there may be a risk of deep discharge.

Power saving mode

- feature to be explained

SD Card operation

FishBMS is equipped with a push-push micro SD card holder to support FAT32 cards up to 8 GB of size (possibly bigger).

Downloading and uploading configuration

SD card allows additional possibility (besides serial / uart / bluetooth interface) to download and upload BMS configuration. This is usefull when you need to configure another BMS with exactly the same configuration like the other one.

To download a new configuration, copy the file “__FISH.TXT” to the root of SD card. Insert the SD card and power on BMS. You should notice that boot phase takes longer (all lights are on). After new config is successfully saved to the internal EEPROM, you will find a file “OKFISH.TXT”, which is the original config file, only renamed so it will not be loaded every time you turn the power on.

Each time BMS starts and SD card is plugged in the system will save current configuration to the file “SAVEFISH.TXT”. This is useful when you make some changes e.g. with bluetooth application and want to copy setting to another device.

In case on any error during SD card operation you will find a file “ERROR.TXT” with a problem description.

Logging BMS data

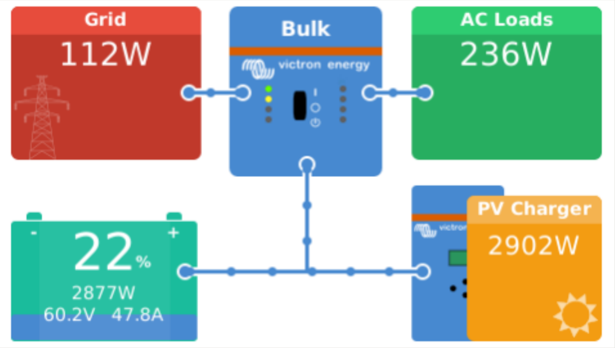
The ability to log various data automatically to a csv files on a SD card can be enabled in configuration – a few basic templates and timing are available.

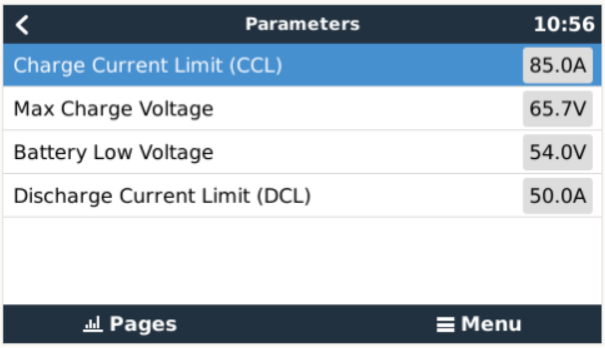
In order to save more complex or customer specific data we can prepare custom firmware based on requirements.

Victron GX interface (CANbus)

Connect BMS CAN to Venus on VE.CAN port (pin 7 CANH = whitebrown, pin 8 CANL = brown).

FishBMS can be used as a battery monitor for Victron systems. The only thing which needs to be done is proper wiring of CAN bus lines to a CCGX / Venus GX / Octo GX device. In software compile option “MODULE_VICTRON_STORAGE” needs to be added.




< Parameters 10:56

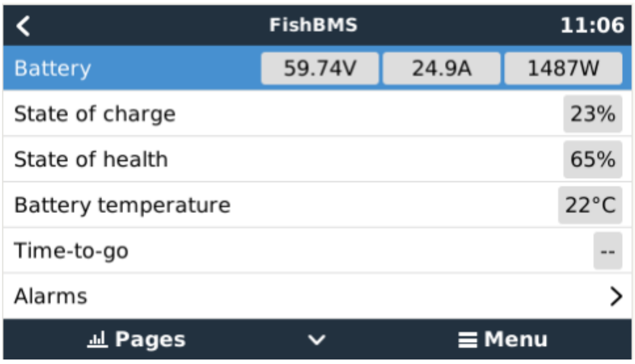
Charge Current Limit (CCL)	85.0A
Max Charge Voltage	65.7V
Battery Low Voltage	54.0V
Discharge Current Limit (DCL)	50.0A

Register values in 4045 – 4048 must be set according to the battery, also SOC parameters (OCV table, drifting params) since it is used in CCGX / Venus GX / Octo GX.
- MPPT controllers will listen to charge current limit CCL (Venus firmware at least 2.20)
- if discharge current limit (DCL) is 0, Victron will stop inverting! (in island/off grid mode it will shut down the power)
- if discharge current limit is small, the rest of power will be taken from the grid
- charge and discharge voltage are not used yet (Feb 2019)
- alarm and warnings will be shown on the display according to BMS errors/warnings
- if no temperature sensors are wired, BMS will send maximum balancer temperature
- Victron now supports not only ESS mode, but others too (Feb 2019)
- make sure DVCC and SVS are enabled

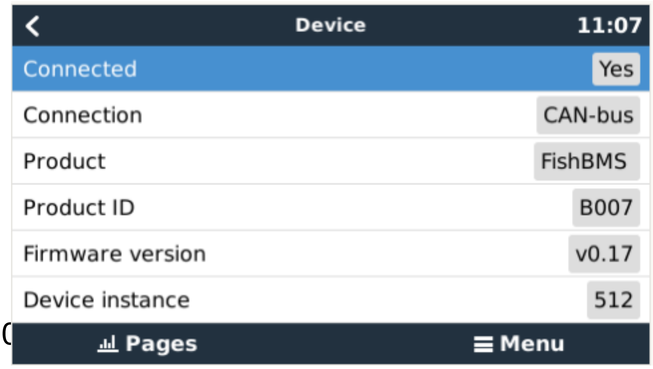
How to check min/max cell from Venus screen? (deprecated, see below)

- make sure you are using BMS firmware 17 or higher
- if state of charge is **higher** than 50% then V_{cell_max} = State of health / 20
- if state of charge is **lower** than 50% then V_{cell_min} = State of health / 20

Example: State of health 65%, state of charge is 23%, minimum cell voltage is 65/20=3.25V


< FishBMS 11:06

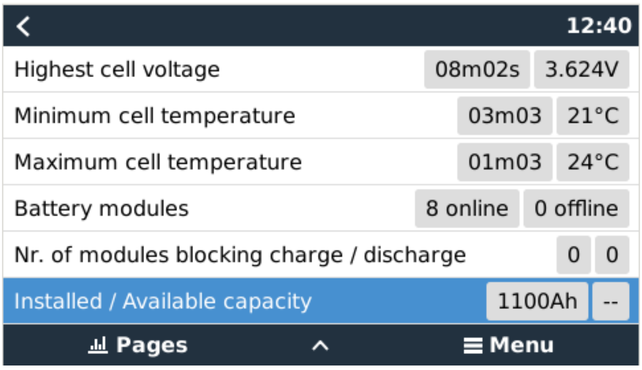
Battery	59.74V	24.9A	1487W
State of charge			23%
State of health			65%
Battery temperature			22°C
Time-to-go			--
Alarms			>


< Device 11:07

Connected	Yes
Connection	CAN-bus
Product	FishBMS
Product ID	B007
Firmware version	v0.17
Device instance	512

BMS firmware 20+ features

- shows cell min and cell max voltages
- shows cell min and cell max temperatures
- shows total capacity
- shows firmware version in “device” screen
- shows number of battery modules
- fixed battery average temperature
- fixed zero battery charge current when low voltage error
- battery identification “FishBMS – Xxs” where xx means how many cells is configured
- identification for min/max values where 08m02s means 8th module 2nd cell
- GX firmware 2.50 or higher



Highest cell voltage	08m02s	3.624V
Minimum cell temperature	03m03	21°C
Maximum cell temperature	01m03	24°C
Battery modules	8 online	0 offline
Nr. of modules blocking charge / discharge	0	0
Installed / Available capacity	1100Ah	--

Android application – FishBMS

Latest development version of app is available online: http://evracing.cz/fishbms

Currently it is possible to monitor following BMS values.

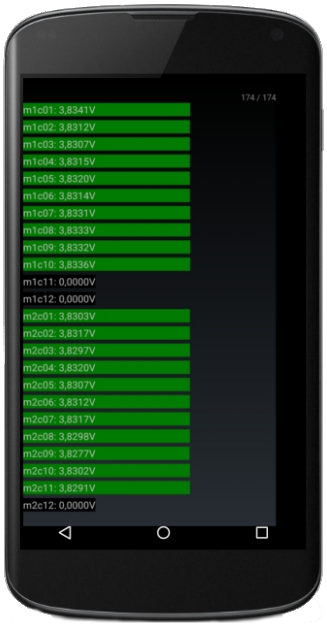
- battery pack voltage
- current (drive & regen)
- minimum and maximum cell voltage
- all cells values (0.1mV resolution)
- cells temperatures (5 channels per measurement module)
- motor temperature
- speed

FishBMS android application can be also used for BMS configuration over MODBUS / Bluetooth, and offers some more user interface for:

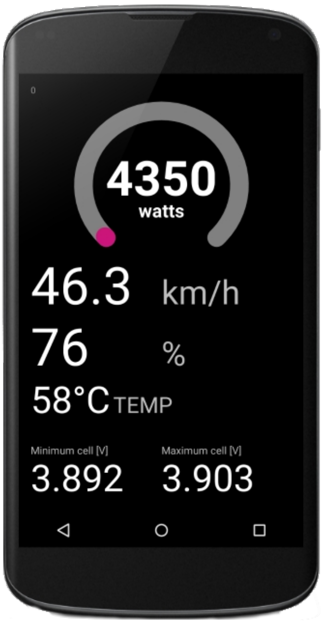
- current sensor calibration (currently done manually via registers)
- speed sensor calibration based on GPS (to be implemented)

In future it may be possible to support custom created views available quickly under buttons from app main menu. Following views can be used by users now:

- monitor view (power or amps, speed, SOC, motor temp, min max cell)
- cells view (showing all the cells' voltages and temperatures)
- cell & module configuration view (to be implemented)



```
174 / 174
m1c01: 3,8341V
m1c02: 3,8312V
m1c03: 3,8307V
m1c04: 3,8315V
m1c05: 3,8320V
m1c06: 3,8314V
m1c07: 3,8331V
m1c08: 3,8333V
m1c09: 3,8332V
m1c10: 3,8336V
m1c11: 0,0000V
m1c12: 0,0000V
m2c01: 3,8303V
m2c02: 3,8317V
m2c03: 3,8297V
m2c04: 3,8320V
m2c05: 3,8307V
m2c06: 3,8312V
m2c07: 3,8317V
m2c08: 3,8298V
m2c09: 3,8277V
m2c10: 3,8302V
m2c11: 3,8291V
m2c12: 0,0000V
```



0

4350 watts

46.3 km/h

76 %

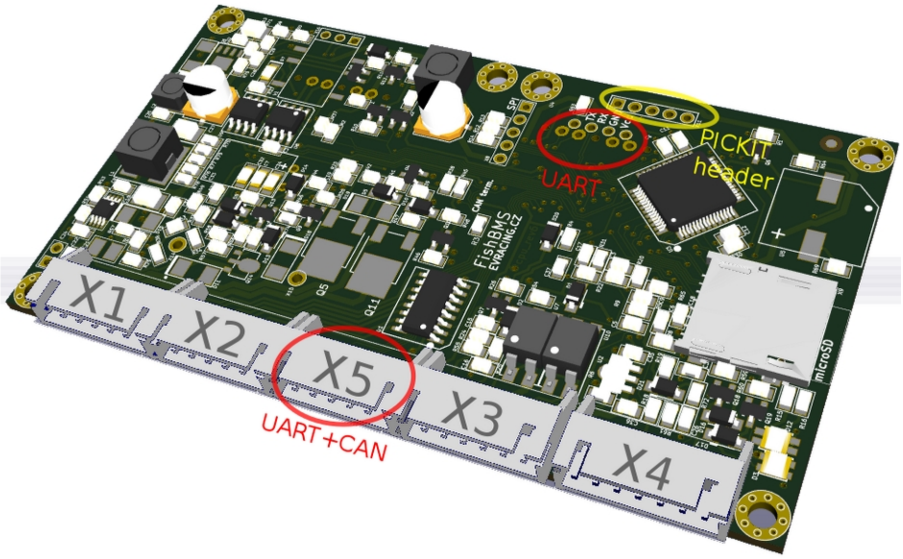
58°C TEMP

Minimum cell [V] 3.892

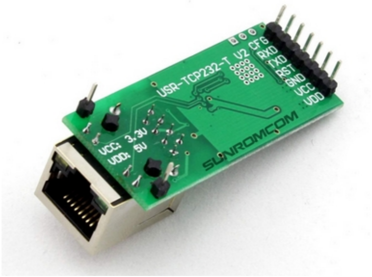
Maximum cell [V] 3.903

Communication interfaces

X5 connector serves for communication purposes (UART = GND, RX, TX, 3.3V or/and CAN = CANL, CANH). In addition there is an UART header located directly on board. If BMS is equipped with bluetooth module (usually HC-06) it is then located in this pin header and in order to use UART on X5 then the onboard module has to be removed for correct operation.



UART communication modules

HC-06 bluetooth - 3.3V power supply 	**USR TCP232 module** - 3.3V Vcc power supply 
PL2303 UART-USB converter - do not connect red 5V wire 	**USR GSM module** - requires 12V power (cannot be supplied from BMS!)

Protocol details

BMS uses MODBUS protocol over serial/UART for communication and configuration. We recommend to use supplied Bluetooth module HC-06 and FishBMS Android application to readout the data and to configure the BMS (latest unstable release is available online http://evracing.cz/fishbms). It is also possible to connect USB to serial adapter and read data out directly with a computer (PC, Raspberry etc.).

For reliable battery protection one should understand the basic BMS configuration (registers 4000 – 4080). The most important is setting cell minimum voltage and cell maximum voltage. After exceeding these levels the BMS will switch off the main contactor(s) (meaning change state from **hvon** to the state **idle** or **sleep**). This is an emergency action to protect the cells and it should not normally happen. Thus BMS will not automatically turn on (it will require new keyswitch event).

Simplified MODBUS implementation

Please note that only functions 0x03 (read holding registers) and 0x10 (write multiple registers) are implemented. You can read or write up to 20 registers maximum. To read or write more registers you need to repeat the command. Default slave address of the BMS master is 1. In MODBUS all bytes are sent in big endian format except CRC, which has the opposite (little endian) format.

Read holding registers example

01 03 03 E8 00 02 AA 2D (request)

bytes	description
01	The slave address (always 01)
03	Function code (03 is for read holding registers function)
03 E8	The data address of the first register to read
00 02	The total number of registers requested.
AA 2D	CRC (cyclic redundancy check)

01 03 04 8F B5 8F BA B4 0F (answer)

bytes	description
01	The slave address (always 01)
03	Function code (03 is for read holding registers function)
04	The number of data bytes to follow (2 registers x 2 bytes each = 4)
8F B5	The contents of register 1000 (36789 → 3678.9 mV)
8F BA	The contents of register 1001 (36794 → 3679.4 mV)
B4 0F	CRC (cyclic redundancy check)

<table>
<tr><td colspan="3"><b>MODBUS Registers – cell data</b></td></tr>
<tr><td colspan="3">Cell data start at register 1000 + given offset. For each additional module just add offset 100*module number starting from zero. If you enter the thermistor Beta value (register 4028 then Analog inputs are recalculated to a temperature in cK units)

Register number = 1000 + module*100 + offset

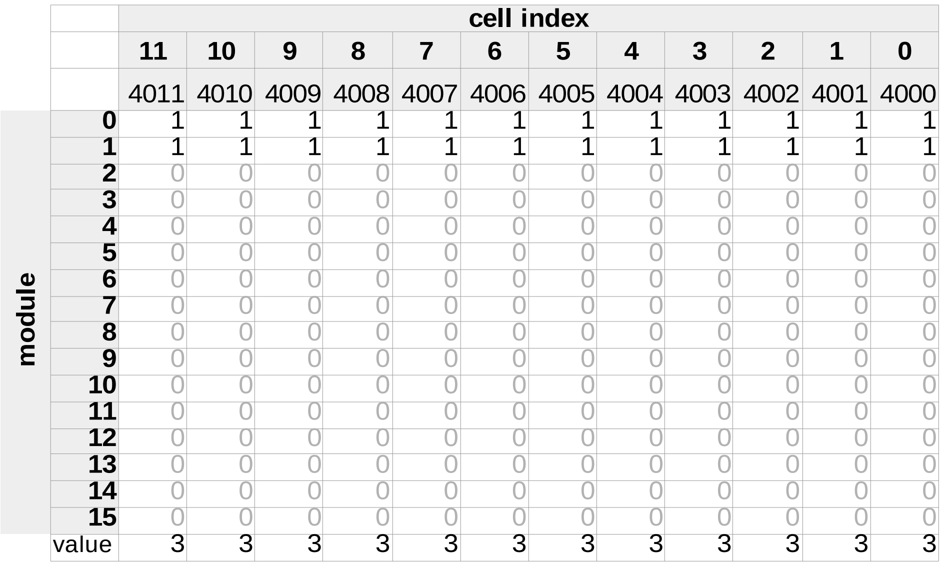
E.g. register for 7th cell voltage in 5th module is 1506 (= 1000+5*100+6).</td></tr>
<tr><td>#reg offset</td><td>R/W</td><td>Description</td></tr>
<tr><td>0</td><td>R</td><td>Cell 1</td></tr>
<tr><td>1</td><td>R</td><td>Cell 2</td></tr>
<tr><td>2</td><td>R</td><td>Cell 3</td></tr>
<tr><td>3</td><td>R</td><td>Cell 4</td></tr>
<tr><td>4</td><td>R</td><td>Cell 5</td></tr>
<tr><td>5</td><td>R</td><td>Cell 6</td></tr>
<tr><td>6</td><td>R</td><td>Cell 7</td></tr>
<tr><td>7</td><td>R</td><td>Cell 8</td></tr>
<tr><td>8</td><td>R</td><td>Cell 9</td></tr>
<tr><td>9</td><td>R</td><td>Cell 10</td></tr>
<tr><td>10</td><td>R</td><td>Cell 11</td></tr>
<tr><td>11</td><td>R</td><td>Cell 12</td></tr>
<tr><td>12</td><td>R</td><td>Analog input[0]</td></tr>
<tr><td>13</td><td>R</td><td>Analog input[1]</td></tr>
<tr><td>14</td><td>R</td><td>Analog input[2]</td></tr>
<tr><td>15</td><td>R</td><td>Analog input[3]</td></tr>
<tr><td>16</td><td>R</td><td>Analog input[4]</td></tr>
<tr><td>17, 18</td><td>R</td><td>Reserved for module configuration</td></tr>
<tr><td>19</td><td>R</td><td>Balancing ON/OFF (bit0 → 1st cell, bit1 → 2nd cell etc.)</td></tr>
</table>

MODBUS Registers – global data		
#reg	R/W	Description
3000	R	Warning
3001	R	Error
3002	R	Error CPU
3003	R	Current [0.1 A] (signed)
3004	R	Voltage [0.01 V]
3005	R	Speed [0.01 km/h]
3006	R/W	SOC [0.01 %]
3007	R	Cell minimum voltage [0.1 mV]
3008	R	Cell maximum voltage [0.1 mV]
3009	R	Motor temperature [cK]
3010	R/W	Amperhour counter [0.01 Ah]
3011	R/W	Trip [0.01 km]
3012	R/W	Throttle override value (255 → throttle fully closed)
3013	R	Temperature – minimum cell
3014	R	Temperature – maximum cell
3015	R	Temperature – average cell
3016	R	Temperature – balancer max
3017	R	Analog input 1 raw value (motor sensor)
3018	R	Analog input 2 raw value (current sensor)
3019	R	Analog input 3 raw value (analog input 0)
3020	R	Analog input 4 raw value (analog input 1, HW >= rev4 switch NTC)
3021	R	Analog input 5 raw value (capacitor voltage)
3022	R	Analog input 1 voltage [0.1mV]
3023	R	Analog input 2 voltage [0.1mV]
3024	R	Analog input 3 voltage [0.1mV]
3025	R	Analog input 4 voltage [0.1mV]
3026	R	Analog input 5 voltage [0.01V]
3027	R	Total number of measurement modules (from config)
3028	R	Total number of cells (from config)
3029	R	PEC error communication counter (isoSPI)
3030	R	PEC percentage (isoSPI communication error rate)

3031	R/W	Ah counter positive [0.01 As] (discharge)
3032	R/W	
3033	R/W	Ah counter negative [0.01 As] (regen or charge)
3034	R/W	
3035	R/W	Trip counter [1 mm]
3036	R/W	
3037	R/W	Trip mm counter last
3038	R/W	
3039	R/W	outputs
3040	R/W	Last As counter
3041	R/W	
3042	R/W	DistMm
3043	R/W	debug
3044	R/W	debug
3045	R/W	debug
3046	R/W	debug
3047	R/W	debug
3048	R/W	debug
3049	R/W	debug
3050	R/W	debug
3051	R/W	Current limit charge
3052	R/W	Current limit discharge
3053	R/W	Charge end voltage
3054	R/W	Discharge end voltage
3055	R	Switch temperature NTC
3056	R	Min cell dynamic calculated limit

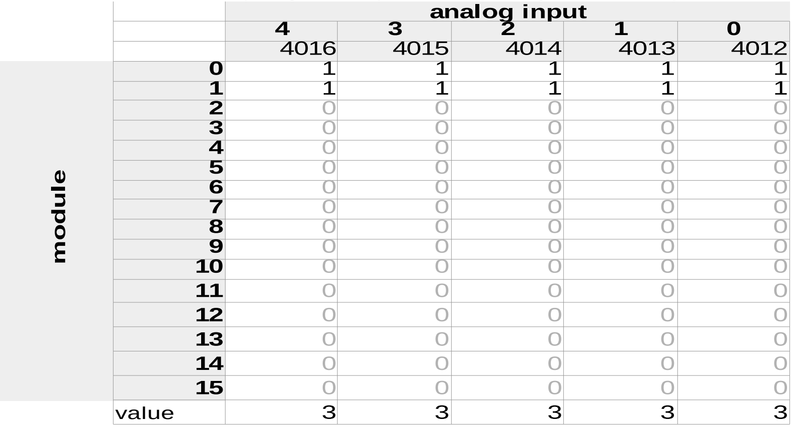
MODBUS Registers – commands		
#reg	R/W	Description
3500	R/W	bit0: keyswitch bit1-15: reserved
3501	R/W	**BMS testing** bit0: balancing (all on, all off) - write 1 bit1: balancing (knight rider) - write 2 bit2: cycle outputs (DO0-5, rel1, rel2) - write 4 bit3: set outputs off (DO0-5, rel1, rel2) - write 8 bit4: beeper - write 16 bit5: activate charger input - write 32 bit6: deactivate charger input - write 64 bit7: run test mode overdischarge - write 128 bit8: run test mode overcharge - write 256 bit9: run over temperature test - write 512 bit10: run under temperature test - write 1024 bit11-15: reserved
3502	W	**Turn outputs ON** bit0: DO0 bit1: DO1 bit2: DO2 bit3: DO3 bit4: DO4 bit5: DO5 bit6: rel1 (precharge) bit7: rel2 (keyswitch) bit8: chgEn bit9-15: reserved
3503	W	**Turn outputs OFF** bit0: DO0 bit1: DO1 bit2: DO2 bit3: DO3 bit4: DO4 bit5: DO5 bit6: rel1 (precharge) bit7: rel2 (keyswitch) bit8: chgEn bit9-15: reserved
3504	R/W	set beeper timeout in (6 --> 60ms etc)
3505	R	flags
3506	R	byte0: inp0, inp1, out0=HV_en, out1, out2, out3, out4, out5 byte1: out6=precharge, out7=ignition

MODBUS Registers – configuration (non volatile memory)		
#reg	R/W	Description
4000	R/W	Cell configuration registers:
4001	R/W	
4002	R/W	
4003	R/W	
4004	R/W	
4005	R/W	
4006	R/W	
4007	R/W	
4008	R/W	
4009	R/W	
4010	R/W	
4011	R/W	



		cell index											
		11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
		4011	4010	4009	4008	4007	4006	4005	4004	4003	4002	4001	4000
module	**0**	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	0	0	0
	11	0	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	0
	13	0	0	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0
	value	3	3	3	3	3	3	3	3	3	3	3	3

4012	R/W	Cell module analog input registers (temperatures):
4013	R/W	
4014	R/W	
4015	R/W	
4016	R/W	



		analog input				
		4	**3**	**2**	**1**	**0**
		4016	4015	4014	4013	4012
module	**0**	1	1	1	1	1
	1	1	1	1	1	1
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	0
	5	0	0	0	0	0
	6	0	0	0	0	0
	7	0	0	0	0	0
	8	0	0	0	0	0
	9	0	0	0	0	0
	10	0	0	0	0	0
	11	0	0	0	0	0
	12	0	0	0	0	0
	13	0	0	0	0	0
	14	0	0	0	0	0
	15	0	0	0	0	0
	value	3	3	3	3	3

4017	R/W	~~CellsType (used for SOC calculation)~~ ~~0 → Li-ion~~ ~~1 → LiFePO4~~ ~~2 → LiPO~~ low byte = temperature max limit (range -40 to 215) high byte = charge temperature limit (range -40 to 215) to disable temperature check set charge temp limit to 0 and/or max limit to 255, write 255 to disable both limits write 10330 (=0x285A) 0C charge minimum, 50C maximum
4018	R/W	currentSensor 0 → no current sensor used 1 → integrated current +-200A sensor used 2 → external current sensor used
4019	R/W	currentSensorOffset (offset in A/D steps, default is 1650) - zero calibration

4020	R/W	CurrentSensorSlope (100 * conversion value 10*0.66 mV / A) e.g. 6.6 mV / A → 660, beginning SW rev8 signed value to allow direction change (-660 ==> 64875)
4021	R/W	capacity (0.1Ah (0-6553.5 Ah, default 100Ah → 1000)
4022	R/W	WheelCircumference (units mm, default 2000)
4023	R/W	numPoles (number of poles times ratio * 100, default 1 → 100) e.g. 13 pulses per rev → 1300 e.g. 4 pulses per rev times 6.35 chain transfer ratio → 2540
4024	R/W	FW16+: 1byte = battery resistance [0.1mOhm] - for battery pack > 100Ah use value < 10 2byte = drift speed (0 = no drifting / 5s 1 soc difference 10% ==> 0.01%, 100% ==> 0.1% 10 soc difference 10% ==> 0.1%, 100% ==> 1% max 25 ~~[s] after this timeout SOC will reset based on OCV~~ ~~(default 10min → 600, off → 65535)~~ ~~Note: if the chemistry is set to LiFePO4 the reset will happen only if the highest cell is above 3.3V or lowest cell under 3.1V~~
4025	R/W	[mV] reference voltage for the CPU (defaul 3.3V → 3300)
4026	R/W	[ohms] reference resistor size (default 2000)
4027	R/W	if KTY sensor is used then > 0 (default 1)
4028	R/W	Beta value for thermistor NTC (cell temp sensor, default 3800)
4029	R/W	**[mV] maximum cell voltage *** (error, max 5000mV)
4030	R/W	**[mV] minimum cell voltage *** (error, min 1500mV)
4031	R/W	**[mV] balancing treshold** (default: 3600mV)
4032	R/W	**[delta mV] balancing difference**, highest cell difference allowed (default 1000mV, min 2mV max 1000mV)
4033	R/W	[delta mV] warning margin (is also used as charger connect hysteresis value)
4034	R/W	[ms] shutdown timeout after error occurs (min 1000 ms)
4035	R/W	**Please refer to the chapter Output modes + output flags 4037**

4036	R/W	Keyswitch combination input (default 3) 1: automatically on after BMS starts 2: keyswitch on when DI0 enabled 3: push long DI0 → on, push long DI0 → off FW rev >= 10 Input modes NOTE: if set 0 then it is possible to turn keyswitch on only via MODBUS reg. 350X (e.g. from app)
4037	R/W	Output flags – see above
4038	R/W	Beeper config 0: beeper off >0: beeper on
4039	R/W	CAN id - non zero to enable CAN module, speed 500kbps if CAN X id is set to 2048-4096 then resulting CAN id will be X – 2048 and speed will be set to 250kbps firmware options: MODULE_VICTRON_STORAGE fixed can speed 250kbps only: CAN_SEND_TCCHARGER = protocol id (1000,1017,1018,1030)
4040	R/W	Max motor temperature [cK] (default no limit, recommended 37315 --> 100°C)
4041	R/W	Watchdog - reboot BMS after no MODBUS communication [s] 65535 = disabled, minimum 30s
4042	R/W	Throttl override mode (0 - 255)
4043	R/W	Precharge resistor ratio (default: based on HW revision)
4044	R/W	turn off high voltage after timeout when no current (default 65535 = off)
4045	R/W	**End of charge limit voltage per cell** (must be lower or same as max cell voltage) [1mV]
4046	R/W	**Max charge current** [0.1A] is used to calculate register 3050 (will start to decrease its value starting 4045 - 200mV)
4047	R/W	**Minimum discharge voltage per cell** (must be higher or same as min cell voltage) [1mV]
4048	R/W	**Max discharge current** [0.1A] is used to calculate register 3051 (will start to decrease its value starting 4046 + 200mV)
4049	R/W	set DO5 on when soc > high byte (fw 17+) clear DO5 on when soc < low byte - invert when high byte < low byte

4050	R/W	Max MOSFET switch temperature [cK] (default no limit, recommended 37315 --> 100°C)
4051-4061	R/W	SOC OCV table (0%, 10%, ... 100%)
4062	R/W	Proportional (charge PID)
4063	R/W	low byte = bottom SOC limit (range 0 to 100) high byte = top SOC limit (range 0 to 100) default bottom=10% top=90% (write 0x0A5A=**2650**)
4064	R/W	1-4bit: paralel pack (total voltage divided by 1-16) 5-12bit: 0.1mV offset each cell (0-25.5mV) 13bit: positive (1) negative (0) 14-16: reserved Default = 0 (no voltage calibration)
4065 - 4075	-/-	reserved
4076	R/W	SOC (will be loaded after power off)
4077	R/W	Total distance counter [1 m] (saved each 2 hours)
4078	R/W	
4079	R	Runtime counter (incremented each 2 hours)
4080	R	Serial number

* maximum cell voltage cannot be lower than minimum cell voltage, maximum cell voltage cannot be higher than 5000mV, minimum cell voltage cannot be lower than 1500mV (if such a condition occures then default limits will be set per chemistry, or default LiFePO4 limits if chemistry is not set)

MODBUS special registers		
#reg	R/W	Description
5000	R	Firmware version
5001	W	Write 43690 to reset BMS Write 28730 to set default settings (fw 17+) Write 7658 to change BT name (write string to debug regs), fw18+
5002	R	Hardware revision

CANbus description

Byte order is motorola. CAN ID can be defined in the register 4039.

RX messages

BMS can receive certain data from another devices so it is possible to change parameters, activate outputs and so on.

BMS modbus RX (message ID +0x00), length 3 - 5			
byte 0	Mode	**0 read modbus address** 1 read response OK 2 read response ERROR **3 write modbus address** 4 write response OK 5 write response ERROR 6-255 reserved, custom	uint8
byte 1, 2	Modbus address	register address	uint16
byte 3, 4	Modbus data	value	uint16
Other modes (not part of generic firmware)			
mode = 10: byte 1+2 current (0.1A, signed), 3+4 speed (pulse counter), 5 motortemp (C, offset -40) compile option = CAN_RECEIVE_MSG10			

e.g.: keyswitch on: 03 0d ac 00 01

TX messages

(ID +0x01) reserved					

BMS State of charge TX (ID +0x02)			
byte 0	SOC	State of charge	uint8, 0 - 100 %, div 2
byte 1	Temp min	Minimum battery temperature	uint8, offset -40, -40 - 215 C
byte 2	Temp max	Maximum battery temperature	uint8, offset -40, -40 - 215 C
byte 3, 4	Cell min	Minimal cell voltage	uint16, 0-6553.6 mV
byte 5, 6	Cell max	Maximal cell voltage	uint16, 0-6553.6 mV
byte 7	Watchdog	Watchdog counter	uint8, 0-255

BMS Power TX (ID +0x03)			
byte 0,1	Voltage	Sum of all cells	uint16, 0-655.36V
byte 2,3	Current	Battery current	int16, -3276.7 to 3276.8 A
byte 4,5	Error bits	See Table 2: Error & warning bits	0 - 65535
byte 6, 7	Bit status	Inputs and outputs feedback	uint16

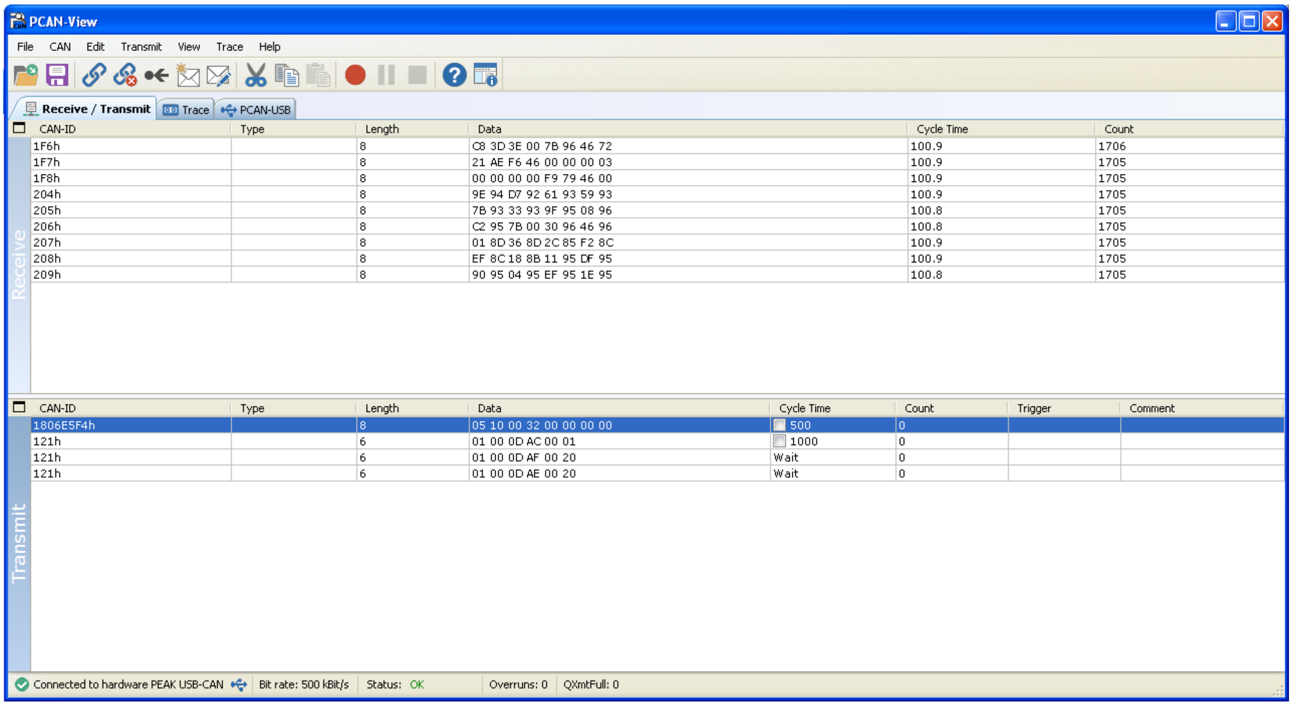
BMS Speed TX (ID +0x04)			
byte 0,1	Speed	Calculated speed	uint16, 0-655.36 km/h
byte 2,3	Trip	Calculated distance	uint16, 0-655.35 km
byte 4,5	Capacity	Calculated capacity	-327.67 to 327.68 Ah

BMS voltages TX (starting ID +0x10 up to number of cells / 4)			
byte 0,1	Cell i	Cell voltage	uint16, 0-6553.6 mV
byte 2,3	Cell i+1	Cell voltage	uint16, 0-6553.6 mV
byte 4,5	Cell i+2	Cell voltage	uint16, 0-6553.6 mV
byte 6,7	Cell i+3	Cell voltage	uint16, 0-6553.6 mV
Not part of generic firmware (compile option = CAN_SEND_VOLTAGES)			

BMS temperatures TX (starting ID +0x40 up to number of temperatures / 4)			
byte 0,1	Temp j	Cell temperature	uint16, 0-655.36 cK
byte 2,3	Temp j+1	Cell temperature	uint16, 0-655.36 cK
byte 4,5	Temp j+2	Cell temperature	uint16, 0-655.36 cK
byte 6,7	Temp j+3	Cell temperature	uint16, 0-655.36 cK
Not part of generic firmware (compile option = CAN_SEND_VOLTAGES)			

Example CAN output

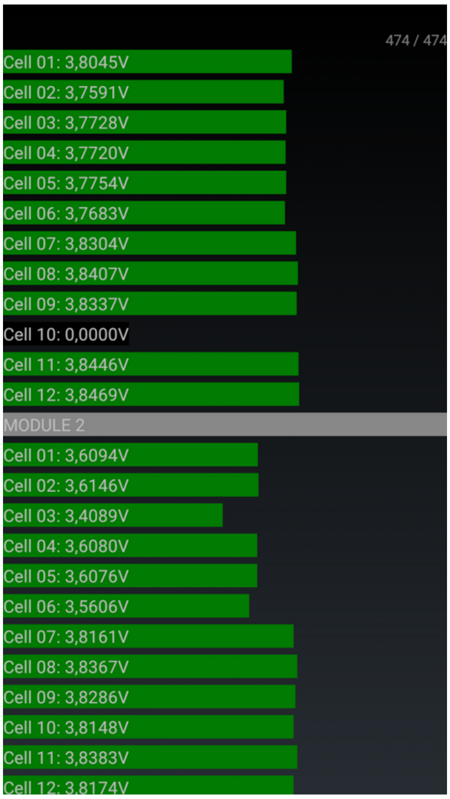
CAN ID was set to 500 (0x1F4) while all cells connected to both first and second measurement module are enabled.



CAN-ID	Type	Length	Data	Cycle Time	Count
1F6h		8	C8 3D 3E 00 7B 96 46 72	100.9	1706
1F7h		8	21 AE F6 46 00 00 00 03	100.9	1705
1F8h		8	00 00 00 00 F9 79 46 00	100.9	1705
204h		8	9E 94 D7 92 61 93 59 93	100.9	1705
205h		8	7B 93 33 93 9F 95 08 96	100.8	1705
206h		8	C2 95 7B 00 30 96 46 96	100.8	1705
207h		8	01 8D 36 8D 2C 85 F2 8C	100.9	1705
208h		8	EF 8C 18 8B 11 95 DF 95	100.9	1705
209h		8	90 95 04 95 EF 95 1E 95	100.8	1705

CAN-ID	Type	Length	Data	Cycle Time	Count	Trigger	Comment
1806E5F4h		8	05 10 00 32 00 00 00 00	□ 500	0		
121h		6	01 00 0D AC 00 01	□ 1000	0		
121h		6	01 00 0D AF 00 20	Wait	0		
121h		6	01 00 0D AE 00 20	Wait	0		

Connected to hardware PEAK USB-CAN | Bit rate: 500 kBit/s | Status: OK | Overruns: 0 | QXmtFull: 0

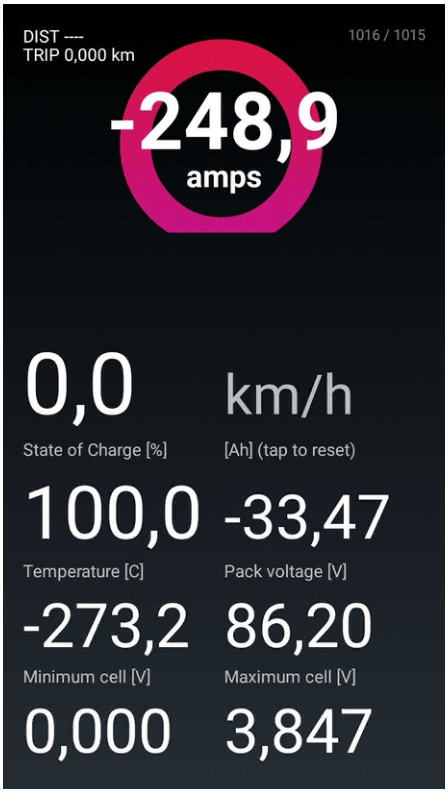


474 / 474

Cell 01: 3,8045V
Cell 02: 3,7591V
Cell 03: 3,7728V
Cell 04: 3,7720V
Cell 05: 3,7754V
Cell 06: 3,7683V
Cell 07: 3,8304V
Cell 08: 3,8407V
Cell 09: 3,8337V
Cell 10: 0,0000V
Cell 11: 3,8446V
Cell 12: 3,8469V

MODULE 2

Cell 01: 3,6094V
Cell 02: 3,6146V
Cell 03: 3,4089V
Cell 04: 3,6080V
Cell 05: 3,6076V
Cell 06: 3,5606V
Cell 07: 3,8161V
Cell 08: 3,8367V
Cell 09: 3,8286V
Cell 10: 3,8148V
Cell 11: 3,8383V
Cell 12: 3,8174V



DIST ----
TRIP 0,000 km

1016 / 1015

-248,9 amps

0,0 km/h

State of Charge [%]: 100,0

[Ah] (tap to reset): -33,47

Temperature [C]: -273,2

Pack voltage [V]: 86,20

Minimum cell [V]: 0,000

Maximum cell [V]: 3,847

Flashing new firmware

New firmware can be flashed to support new features or fix bugs.

Firmware options

MODULE_VICTRON_STORAGE

- CAN bus communication data for VenusGX / ColorControlGX / OctoGX (set min/max values in 4045 – 4048 for this purpose)
- ~~DOUT1~~ (left out from revision 16):
    - activates when battery status changes to HV (contactor closed, high voltage enabled)
    - deactivates when warning low (change to error low?)

POWER_OFF_ENABLE

- self power off function when when lowest cell voltage is lower than minimum cell voltage for at least 2 hours

CAN_SEND_BASICDATA

- populate battery basic data on CANbus (includes messages: “BMS State of charge TX”, “BMS Power TX”, “BMS Speed TX”)

CAN_SEND_VOLTAGES

- populate all cell voltages and temperatures on CANbus (includes messages “BMS voltages TX” and “BMS temperatures TX”)

Upgrade via SD card

This step can be done easily by copying the new firmware to a SD card and rebooting BMS (power cycle). The firmware file must be named “firmware.hex”.

Upgrade via Android app

The other possibility is to use a FishBMS Android app to download the firmware file into the BMS. SD card is also required for this approach.

Changelog Software

FW revision 22 (2023)

- charge current limit 10% of nominal if cell temperature is below 3C
- fix recalculate charge voltage limit once when reach 50% SOC
- SOC jump to 1.5% if under low voltage error for more than 10s

FW revision 21 (2021)

- TC charger timeout (connect and disconnect charger based on CAN RX from TCC)
- 4035 bit 14 - enable bistable relay for charger
- disabled runtime counter to prevent SN overflow
- 4039 two color SOC output DO3 & 4

FW revision 20 (2020-05)

- added 4063 bottom and top SOC limits (currently used for switching DOs)
- added 4064 voltage calibration (paralel packs)
- changed cell imbalance error to warning
- CAN speed configurable for Victron (250kbps and 500kbps)
- fixed zero charging current constraint when battery fully depleted (in error)
- added dynamic low cell error limit (weak battery or high current drain needed) 500mV up
- Victron protocol update (min/max cell voltage + other values)
- fix min/max temperature output (will use onboard balancing temp sensor if no external)

NOTE before update: check 4039 (must be > 0), 4063 and 4064!

FW revision 19 (2019-07)

- added cell imbalance error, changed discharge current estimation (Victron protocol, reaching discharge voltage per cell = 0.1A limit, reaching absolute min cell voltage = 0 A limit = inverter off)

FW revision 18 (2019-04)

- increased interval for testing outputs (+ sound signal)
- added function to change bluetooth dongle name

FW revision 17 (2019-03-24)

- SOC drifting slowed down
- charging control (CHG_EN output) uses “Victron” registers now
- suspend mode enable also if no communication with slaves, removed “__reset.txt” and added INP1 pull to zero in order to release suspend mode
- Victron protocol update (see “Victron ESS modes” chapter for details)
- CONFIGLAYER.canID added limited CAN speed configration
- added possibility to reset default settings (register 5001)
- use balancer temp if no additional temp sensors are defined
- SOC pulse output signal (can be used for LED)
- added temperature limits for charge and discharge (register 4019)
- use PCB temperature if no sensors attached
- adjusted SOC drifting parameters
- BMS protocol update (shows cell min/max in Venus – using SOH)
- RX buffer overflow fix

FW revision 16 (2018)

- TC charger support over CANbus (must be compiled)
- proportional charge parameter to prevent oscillations is configurable
- balancing temp limit (80C)

- limiting PCB temperature (80C) when balancing (activate 1st and/or 2nd channel on MM)
- SOC OCV table and drift configuration (register 4024)
- user defined SOC OCV map (registers 4051 to 4061)
- undervoltage suspend mode (use __reset.txt for suspend mode release)
- remote flashing feature (BMS ready)
- modbus max regs highered to 64

FW revision 15 (2017)

- fixed temperature reference measurement bug (wrong temp reading from MM)
- introduced register 4050 (charge switch temperature limit)
- fixed min voltage and min/max current for Victron interface
- readable HW revision in 5002

FW revision 14 (2017-09-21)

- combined bistable relay output (charge & discharge, details in register 4035)
- bistable switch on and switch off routine changed
- autostart (details in register 4036)
- added CAN matrix to support Victron inverters via CCGX/Venus, not in default FW
- added register 3506 (input and output feedback)
- LTC communication error calculation changed
- unlock sequence to write SN
- turn off charger when switch temp is higher than “maxMotorTemperature”

FW revision 13 (2017-07-16)

- SW pull-up enabled for TX (TX stays high all the time)
- 3.3V turn on order fixed (getting stuck on SPI communication rarely while booting)
- test balancing lights shift timeout fixed

FW revision 12 (2017-06-20)

- added CAN matrix to support Studer inverters via XCOM-CAN, not in default FW

FW revision 11 (2017-05-31)

- HV off timeout (automatic turn off when no current)
- INP1 power limit (throttle override function)

FW revision 10 (2017-04-08)

- changed current sampling frequency (faster), added moving average / kalman filter
- measure NTC temperatures (each measurement module has 5 channels)
- added voltage measurement filtering (200ms acquire time for all voltages)
- fix SOC reset by low voltage
- added support for CPU master revision 3
- precharge feedback voltage condition enabled
- upgraded Android app (temperature support)

FW revision 8 (2017-02-20)

- modbus communication fix (BT getting stuck)

Changelog Hardware

HW revision 5 (2018-01), revision 6

- added diode for precharge switch (bugfix)
- I/O connectors reorganized (prevent installation error, more pins added: Vprecharge, 5V, 3.3V, new connector X5)
- RX/TX bugfix for ethernet module (piggy back connection)

HW revision 4 (2017-06), released

- added power off feature (total disconnect)
- step down 3.3V for better efficiency
- separate board for current measurement (+ bistable contact) and charge switch (6x FET)
- precharge + keyswitch relay exchanged for FETs

HW revision 3 (2017-05)

- output drivers made from discrete components and isolated (not continued)
- integrated DC/DC power supply <95V to 12V
- JST-XH connectors from one side

HW revision 2 (2016)

- step down power supply 12-36V to 3.3V
- change the order of measurements modules

HW revision 1 (2015), released

- initial revision

HW revision 0 (2014)

- first prototype revision

Application and programming examples

Python3 examples

These examples uses Python 3. You may also need to install python3-serial package and python3-pymodbus.

Reading the data out with Python (USB – serial adapter)

In following example we read out 12 cell voltages from first module (starting with register 1000).

```
#!/usr/bin/python3
from pymodbus.client.sync import ModbusSerialClient
client = ModbusSerialClient(method = "rtu", port=”/dev/ttyUSB0, baudrate=115200,
stopbits=1, bytesize=8, timeout=1)
rq = client.read_holding_registers(1000,12,unit=1)
print(rq.registers)
```

Configuring new Bluetooth PIN or name (USB – serial adapter)

Following snippet can be used to reconfigure HC-06 bluetooth module. You only need to connect the module to an USB – serial adapter using GND, RX, TX and power 3.3V (or 5V if the BT module can handle it).

```
#!/usr/bin/python3
import sys
import serial
import time

def writeReadSerial(strtowrite):
    global connection
    print ("Sending: "+strtowrite)
    for char in strtowrite:
        connection.write(char.encode())
        connection.flush()
    output = ""
    print("received: ")
    for i in range(20):
        char = connection.read()
        output += char.decode()
    return (output)

if (len(sys.argv) != 6):
    print("Help:\n\nBAUD1---------1200 \nBAUD2---------2400\nBAUD3---------4800\
nBAUD4---------9600\nBAUD5---------19200\nBAUD6---------38400\nBAUD7---------
57600\nBAUD8---------115200\n")
    sys.exit("Wrong number of arguments. Expecting arguments: PIN NAME USBTTY
BaudOLD BaudNewNum")

PIN=str(sys.argv[1])
NAME=str(sys.argv[2])
DEVICE=str(sys.argv[3])
BAUDOLD=str(sys.argv[4])
BAUD=str(sys.argv[5])
```

```
connection = serial.Serial(port=DEVICE, baudrate=BAUDOLD,timeout=0.1)
print("connected to "+DEVICE)
print("sending: AT")
print(writeReadSerial("AT+VERSION?"))
print(writeReadSerial("AT+NAME"+NAME))
print(writeReadSerial("AT+PIN"+PIN))
print(writeReadSerial("AT+BAUD"+BAUD))
connection.close()
```

Using QModBus utility

QModBus utility is available for Windows / Linux / Mac and can be easily use to read out or configure FishBMS.

Testing BMS function

Following code will test most of the BMS features:

- outputs DO0, DO1, DO2, DO3, DO4, DO5
- outputs rel1, rel2
-

```
#!/usr/bin/python3

#TODO – put code
```