UPDATE: 13.10.2022

We have Growatt three phase inverter model SPH 10000TL3 BH with a software version:
YA1.0/yaaa030402/ZDAA-0002/A0B1D0T3PFU8MAS3

Battery BMS controller is PACE P75S100A-0792 with a software version:
P75S100A-0792-1.00. (see http://www.pacebms.com/en/)

Devices communicate via Canbus protocol based on Pylontech de facto standard version 2019/05/30 1.18.
(Search with "CANBUS protocol of high voltage system" in google)

The interface unit between devices is Raspberry Pi 4 with 2-Channel CAN Waveshare shield
https://www.waveshare.com/2-ch-can-hat.htm

Raspberry program is Python3 code. Python3 execution code is fast enough to follow Canbus traffic.
All Canbus traffic flows through Raspberry. We can control all CAN ID data packages send by Growatt and/or PACE BMS and replace their contents.

The first problem:

Growatt Canbus speed is 500 kbps and PACE speed is 250 kbps. We had no chance to change these speeds. So Raspberry handles this task.

The second problem:

Growatt Cloud server is viewing **SOC-value**, but Growatt inverter is not using it for stopping the charge or the discharge. Raspberry as an interface unit will take care of SOC-value control and stop charging/discharging based on pre-defined values in Python3 program. Fox example upper limit 90% and lower limit as 20%.

The third problem:

Battery BMS sends CAN ID: 0x4220 Bytes#0-1 as “Charge Cutoff Voltage”, but inverter is not using it to stop the charge.
Battery BMS sends CAN ID: 0x4220 Bytes#2-3 as “DisCharge Cutoff Voltage”, but inverter is not using it to stop the discharge.
Charging and discharging could be controlled with CAN ID:0x4210 Byte#0-1 “Battery Pile Control Total Voltage” values. But this needs calibrations to verify the voltage based on the SOC-value, so we rely on SOC-value provide by Battery BMS, and available in Canbus traffic.

The fourth problem:

Growatt has a parameter AC CHG, which can have values On/Off. These values have no meaning. Growatt is charging from GRID whenever the charging is allowed and PV-panels are not producing current enough. In charging, PV-panels are primary.

How to charge the battery?
This is a story.

When the charging is prepared, first force Raspberry to send information to stop charging and discharging as below replacing CAN ID: 0x4280 Byte#0 and Byte#1 as below.

Battery BMS sends CAN ID: 0x4280 Byte#0= 0xAA as “Charge forbidden mark”. Battery BMS sends CAN ID: 0x4280 Byte#1= 0xAA as “Discharge forbidden mark”.

(Remember that Canbus traffic is continuos between Growatt<->BMS)

Then define charging “MAX Charge Current” like this:

Set CAN ID: 0x4220 Bytes#4-5=30000 means 0 Amp. Set CAN ID: 0x4220 Bytes#6-7 charge current limit. For example 30010 means 1 Amp, 30020 means 2 Amp. etc.

(In Pylontech manual “**CANBUS Protocol of High Voltage system**”,
Bytes#4-5 is commented as “MAX Charge Current”, and
Bytes#6-7 is commented as “MAX Discharge Current”.
The comments are incorrect. They should be swapped)

After these reparations replace CAN ID: 0x4280 Byte#0= 0xAB in Canbus data flow. This activates the charging.

How to discharge the battery?

When discharging is prepared, first force Raspberry to send information to stop charging and discharging as below replacing as CAN ID: 0x4280 Byte#0 and Byte#1 as below.

Battery BMS sends CAN ID: 0x4280 Byte#0= 0xAA as “Charge forbidden mark”. Battery BMS sends CAN ID: 0x4280 Byte#1= 0xAA as “Discharge forbidden mark”.

(Remember that Canbus traffic is continuos between Growatt<->BMS)

Then define charging “MAX Charge Current” like this:

Set CAN ID: 0x4220 Bytes#4-5 discharge current limit. For example 29990 means -1 Amp, 29980 means -2 Amp. etc. Set CAN ID: 0x4220 Bytes#6-7 30000 means 0 Amp.

After these reparations replace CAN ID: 0x4280 Byte#1= 0xAB in Canbus data flow. This activates discharging.

The charging and discharging are stopped also when current limits are defined as zero or 30000. Even in the case when “forbidden mark” is defined to allow charge/discharge.

As you see the logic is quite strange as it looks like the Bytes#4-5/Bytes#6-7 comments were swapped in page 5/11. But observe that the discharging current MUST be set as negative value for example 29990 means -1 Amp . This is not commented at all!

We observed that we can control charging/discharging with these parameters:

Charge

Battery BMS sends CAN ID: 0x4280 Byte#0= 0xAA as “Charge forbidden mark”.
Battery BMS sends CAN ID: 0x4280 Byte#1= 0xAB as “Discharge forbidden mark”.
Set CAN ID: 0x4220 Bytes#4-5=30000 means 0 Amp.
Set CAN ID: 0x4220 Bytes#6-7 For example 30010 means 1 Amp, 30020 means 2 Amp. etc.

Discharge

Battery BMS sends CAN ID: 0x4280 Byte#0= 0xAA as “Charge forbidden mark”.
Battery BMS sends CAN ID: 0x4280 Byte#1= 0xAB as “Discharge forbidden mark”.
Set CAN ID: 0x4220 Bytes#4-5=30000 means 0 Amp.
Set CAN ID: 0x4220 Bytes#6-7 For example 29990 means -1 Amp, 29980 means -2 Amp. etc.

So “Charge forbidden mark”, “Discharge forbidden mark” and “MAX Charge Current”can be defined as constants. Only fine tuning “MAX Discharge Current” from positive value to negative value will define charging/discharging states!!!

How about AC CHG attribute?

Growatt inverter is NOT controlled by AC CHG On/Off state. The inverted charges continuously, if the charge parameters are defined as above is described. (Remember that PV-panels are primary source to charge batteries, and extra current is taken from the Grid to fulfill charge current definition.)

This will be solved by using SDM630 MODBUS V3. This measuring unit is connected between the GRID and LOAD. We can measure the current and current directions from LOAD to GRID and vice versa. With this information we can fine tune the charging from the GRID, when PV-Panels are not enough. So AC CHG parameter will be ignored.

During daytime PV-panels provide power to the LOAD and the GRID. If the current to the GRID is positive, we can charge batteries at a same time. Fine tuning is done by adjusting battery charging current properly, and checking the response in the SMD630 current meter. When the GRID current is negative, the charging is forbidden. In that situation PV-panels produce power too little.

UPDATE: 13.10.2022 END END END END END END END END END END END END END END END END END END

OTHER things:

```
Our Growatt  sends continuosly these commands

3010              this is called "Heatbeat"       (Broadcast mode)
4200 data[0]=0x02 "Ensemble information"          (Broadcast mode)
4200 data[0]=0x00 "System equipment information"  (Broadcast mode)
8200 data[0]=0xAA "Control device quit sleep status"
8210 data[0]=0xAA "Charge command"
```

GROWATT IS a master and Battery system is a slave.

And Growatt's CAN ID is "1" and our slave CAN ID is "0" or zero.

So Growatt is sending requests and waiting for answers,
which are described in Pylontech manual-pdf, which i already sent to.

When you read my attach file, you can follow our Enershare battery controller
responses to Growatt's enquirements.

If you use canbus, you must really know what is your growatt Canbus speed.
I assume that it is the very same as our Growatt 500,000 BPS.

(Our Enershare battery controller speed is 250,000 BPS,
so we created an interface to manage this speed problem,
The reason was that we cannot change Growatt Canbus speed (This was checked with Growatt),
also also it was impossible to change Enershare Battery controller speed!!!, also checked with Enershare company)

In Pylontech manual it is told (page 3/11) that Pylontech battery controller speed is 500kbps.

If the speed is diffrent in your Battery canbus,

then you surely will have "BMS COM Fault" error.

Check that:
RJ-45 in Growatt side is as follows:

Pin#4 is CAN HIGH
PIN#5 is CAN LOW

Enershare

Report Sending is LSB or lower byte comes first

```
can1  00004200   [8]  02 00 00 00 00 00 00 00 Response below
can1  00007310   [8]  01 00 02 01 01 02 00 00
can1  00007320   [8]  4B 00 05 0F 2D 00 56 00

can1  00004200   [8]  00 00 00 00 00 00 00 00 Response below (10 lines)

can1  00004210   [8]  A5 09 30 75 9D 04 2E 64
```

Byte0-1	09A5 = 2469	Battery Pile Total Voltage	Resolution: 0.1V Offset: 0
Byte2-3	7530 = 30000	Battery Pile Current	Resolution: 0.1A Offset: -3000A
Byte4-5	049D = 1181	second level BMS Temperature	Resolution: 0.1 ºC Offset: -100 ºC
Byte6	2E = 46	SOC	Resolution: 1%, Offset: 0
Byte7	64 = 100	SOH	Resolution: 1%, Offset: 0

```
can1  00004220   [8]  8C 0A E9 07 4A 79 4A 79
```

Byte0-1	0A8C = 2700	Charge Cutoff Voltage	Resolution: 0.1V Offset: 0
Byte2-3	07E0 = 2016	Discharge Cutoff Voltage	Resolution: 0.1V Offset: 0
Byte4-5	794A = 31050	MAX Charge Current	Resolution: 0.1A Offset: -3000A
Byte6-7	794A = 31050	MAX Discharge Current	Resolution: 0.1A Offset: -3000A

```
can1  00004230   [8]  DF 0C DA 0C 03 00 06 00
```

Byte0-1	0CDF = 3295	MAX Single Battery Cell Voltage	Resolution: 0.001V Offset: 0
Byte2-3	0CDA = 3290	MIN Single Battery Cell Voltage	Resolution: 0.1V Offset: 0
Byte4-5	0003 = 3	MAX Single Battery Cell Voltage Number	Resolution: 1 Offset: 0
Byte6-7	0006 = 6	MIN Single Battery Cell Voltage Number	Resolution: 1 Offset: 0

```
can1  00004240   [8]  7E 04 62 04 11 00 03 00
```

Byte0-1	047E = 1150	MAX Single Battery Cell Temperature	Resolution: 0.1ºC Offset: -100
Byte2-3	0462 = 1122	MIN Single Battery Cell Temperature	Resolution: 0.1ºC Offset: -100
Byte4-5	0011 = 17	MAX Single Battery Cell Temperature Number	Resolution: 1 Offset: 0
Byte6-7	0003 = 3	MIN Single Battery	Resolution: 1 Offset: 0

		Cell Temperature Number	

can1 00004250 [8] 03 02 00 00 00 00 00 00

Byte0	**03** = 3	Basic Status	See Table 1 for details. **3==Idle**
Byte1-2	**0002** = 2	Cycle Period	
Byte3	**00** = 0	Error	See Table 2 for details.
Byte4-5	**0000** = 0	Alarm	See Table 3 for details.
Byte6	**00 = 0**	Protection	See Table 4 for details.
Byte7			

can1 00004260 [8] AC C7 74 27 03 00 02 00

Byte0-1	**C7AC = 51116**	Module Max. Voltage	Resolution: 0.001V Offset: 0
Byte2-3	**2774** = 10100	Module Min. Voltage	Resolution: 0.001V Offset: 0
Byte4-5	**0003** = 3	Module Max. Voltage Number	Resolution: 1 Offset: 0
Byte6-7	**0002** = 2	Module Min. Voltage Number	Resolution: 1 Offset: 0

can1 00004270 [8] 7E 04 62 04 05 00 01 00

Byte0-1	**047E = 1150**	Module Max. Temperature	Resolution: 0.1°C Offset: -100
Byte2-3	**0462** = 1122	Module Min. Temperature	Resolution: 0.1°C Offset: -100
Byte4-5	**0005 = 5**	Module Max. Temperature Number	Resolution: 1 Offset: 0
Byte6-7	**0001** = 1	Module Min. Temperature Number	Resolution: 1 Offset: 0

can1 00004280 [8] 00 00 00 00 00 00 00 00

Byte0	**00 = 0**	Charge forbidden mark	0xAA : effect; other value: NULL
Byte1	**00** = 0	Discharge forbidden mark	0xAA : effect ; other value: NULL
Byte2	**00 = 0**	Reserve	
Byte3	**00** = 0	Reserve	
Byte4	**00 = 0**	Reserve	
Byte5	**00 = 0**	Reserve	
Byte6	**00 = 0**	Reserve	
Byte7	**00 = 0**	Reserve	

can1 00004290 [8] 00 00 00 00 00 00 00 00

Byte0	**00 = 0**	1	See Table 5 for details.
Byte1	**00** = 0	Reserve	
Byte2	**00 = 0**	Reserve	
Byte3	**00** = 0	Reserve	
Byte4	**00 = 0**	Reserve	
Byte5	**00 = 0**	Reserve	
Byte6	**00 = 0**	Reserve	
Byte7	**00 = 0**	Reserve	

command ID: 0x4200 data[0]=2 (is broadcast frame)

response is two data packets below

CAN ID : 0x7310+Addr eli 0x7310

CAN ID：0x7320+Addr eli 0x7320

Get data from ID: 0x7310 **This CAN ID ： 0x7310+Addr**

1 0 2 1 1 2 0 0
Byte#0 = 0x01 ver. A
Byte#1 = 0x00 reserve
Byte#2 = 0x02 Hardware Version-V 0x02
Byte#3 = 0x01 Hardwware Versio –R 0x01
Byte#4 = 0x01 Software Version-V Major 0x01
Byte#5 = 0x02 Software Version-V Minor 0x02
Byte#6 = 0x00 Software Version
Byte#7 = 0x00 Software Version

Get data from ID: 0x7320

4B 0 5 F 2D 0 56 0

Byte#0 = 0x4B Battery Module Qty eli Total number of batteries 0x4B = **75** eli 5 packs of **15 cells**
Byte#1 = 0x00 reserve
Byte#2 = 0x05 Battery Module in series Qty. eli Number of battery modules in series in one cabinet **5**
Byte#3 = 0x0F 15 Cell Qty. in battery module eli Number of batteries in the module **15**
Byte#4 = 0x2D **45** Voltage Level eli Voltage platform yhdessä 15 kennon paketissa resolution: 1V
Byte#5 = 0x00 Offset: 0
Byte#6 = 0x56 **86** AH number Resolution 1AH Varauskyky
Byte#7 = 0x00 Offset: 0

___

Enershare is send errornous information (above) in the bytes 4-5 as **0x2D** which is Voltage level. It should be 240V or (hexa) 0xF0, because we have 5 battery module
48*5= 240V. This must a mistake. This is corrected with RPI when data flows between
Gro-Ener. But I think Growatt is not using this info.

We discovered this by comparing your information you provide in
https://www.eevblog.com/forum/programming/pylontech-sc0500-protocol-hacking/msg3742672/#msg3742672

where you have byte 4-5 a figure 144.

```
  can1  00004200   [8]  02 00 00 00 00 00 00 00 Response Growatt send this
command
  can1  00007310   [8]  01 00 02 01 01 02 00 00 Our battery system  respons
this
  can1  00007320   [8]  4B 00 05 0F 2D 00 56 00 Our battery system  respons
this

  can1  00004200   [8]  00 00 00 00 00 00 00 00 Response Growatt send this
command
  can1  00004210   [8]  A5 09 30 75 9D 04 2E 64 Our battery system  respons
this
  can1  00004220   [8]  8C 0A E9 07 4A 79 4A 79 Our battery system  respons
this
  can1  00004230   [8]  DF 0C DA 0C 03 00 06 00 Our battery system  respons
this
  can1  00004240   [8]  7E 04 62 04 11 00 03 00 Our battery system  respons
this
  can1  00004250   [8]  03 02 00 00 00 00 00 00 Our battery system  respons
this
  can1  00004260   [8]  AC C7 74 27 03 00 02 00 Our battery system  respons
this
  can1  00004270   [8]  7E 04 62 04 05 00 01 00 Our battery system  respons
this
  can1  00004280   [8]  00 00 00 00 00 00 00 00 Our battery system  respons
this
  can1  00004290   [8]  00 00 00 00 00 00 00 00 Our battery system  respons
this

  can1  00003010   [8]  01 00 00 00 00 00 00 00 No Response Growatt send this
command

  can1  00008200   [8]  AA 00 00 00 00 00 00 00 No Response Growatt send this
command

  can1  00008210   [8]  AA 00 00 00 00 00 00 00 No Response Growatt send this
command
```

Example candump can0

```
  can0  00003010   [8]  0F 7D 00 00 00 00 00 00 Growatt send this command
  can0  00004200   [8]  02 00 00 00 00 00 00 00 Growatt send this command
  can0  00007310   [8]  01 00 02 01 01 02 00 00 Our battery system  respons
this
  can0  00007320   [8]  4B 00 05 0F 2D 00 56 00 Our battery system  respons
this
  can0  00003010   [8]  0F 7E 00 00 00 00 00 00 Growatt send this command
  can0  00003010   [8]  0F 7F 00 00 00 00 00 00 Growatt send this command
  can0  00008210   [8]  AA 00 00 00 00 00 00 00 Growatt send this command
  can0  00004200   [8]  00 00 00 00 00 00 00 00 Growatt send this command
  can0  00004210   [8]  A2 09 30 75 BD 04 2E 64           SOC
  can0  00004220   [8]  8C 0A E9 07 4A 79 4A 79 Our battery system  respons
this
  can0  00004230   [8]  DC 0C D5 0C 07 00 06 00 Our battery system  respons
this
  can0  00004240   [8]  70 04 56 04 13 00 03 00 Our battery system  respons
this
  can0  00003010   [8]  0F 80 00 00 00 00 00 00 Growatt send this command
```

```
  can0  00004250   [8]  03 02 00 00 00 00 00 00 Our battery system  respons
this
  can0  00004260   [8]  43 F2 79 30 03 00 01 00 Our battery system  respons
this
  can0  00004270   [8]  70 04 56 04 05 00 01 00 Our battery system  respons
this
  can0  00004280   [8]  00 00 00 00 00 00 00 00 Our battery system  respons
this
  can0  00004290   [8]  00 00 00 00 00 00 00 00 Our battery system  respons
this
  can0  00003010   [8]  0F 81 00 00 00 00 00 00 Growatt send this command
```

Real things:

1.Growatt sends a data request about the battery status with a CAN ID command:

CAN ID : 0x4200 x 8 **00** 00 00 00 00 00 00 00 (Ensemble Information)
and gets an answer

00004210 [8] A5 09 30 75 9D 04 **2F** 64,

where SOC = 0x2F=47%.
Example: Enershare Calculation 100*RemainCapacity/FullCapacity=39710/85000=46,71%

Growatt reads this SOC status calculated by Enershare and reflects it through its own cloud server in its tracked statistics. Growatt is instructed to stop charging when the SOC exceeds a predefined upper limit, but Growatt does not comply with the specified limit.

=> Something is really wrong

2. Growatt does not stop charging. Growatt sends a data request on battery status:

CAN ID : 0x4200 x 8 **00** 00 00 00 00 00 00 00 (Ensemble Information)

and receives a response indicating that Enershare wants to block the charging
.
0x4280 x 8 **AA** 00 00 00 00 00 00 00 (Charging is not allowed).

But Growatt does not follow this information.

=> Something is really wrong

3.Growatt will not discharge the battery under any circumstances. Growatt receives this information from Enershare in the data package, see above. DisCharging is allowed.

0x4280 x 8 AA **00** 00 00 00 00 00 00 (DisCharging is allowed).

=> Something is really wrong

4. Growatt inverter charges the battery directly from the mains (GRID) regardless of the mode in which GRID First, LOAD First, BAT First, Growatt is programmed to operate, and even if AC CHG = OFF. Direct charging from the GRID is strictly prohibited. As a result, the battery is charged up to SOC = 100% until Enershare finally makes an emergency stop.

=> Something is really really wrong

Growatt firmware updates does not solve these problems.