Gmail [redacted]

RE: 【Technical Service】MEAN WELL Switching Power Supply

11 messages

Anke Ge <anke.ge@meanwellusa.com> Tue, May 25, 2021 at 12:55 PM
To: [redacted]

Hi Devin,

Could you let me know what the issue is and what I can help?

Best Regards,

Anke Ge
Technical Support Engineer | Eng. Dept.
T: 510-683-8886 ext.210 or 669-297-0797
MEAN WELL USA, INC. | **w:** www.meanwellusa.com | **E:** anke.ge@meanwellusa.com
SDG related companies : MEAN WELL | POWERNEX | SHARE WELL | FOUNDATION
Pioneering Global Standard Power Supply Industry

MEAN WELL SWITCHING POWER SUPPLY

Technical Service

You have a Technical Service

Contact information :

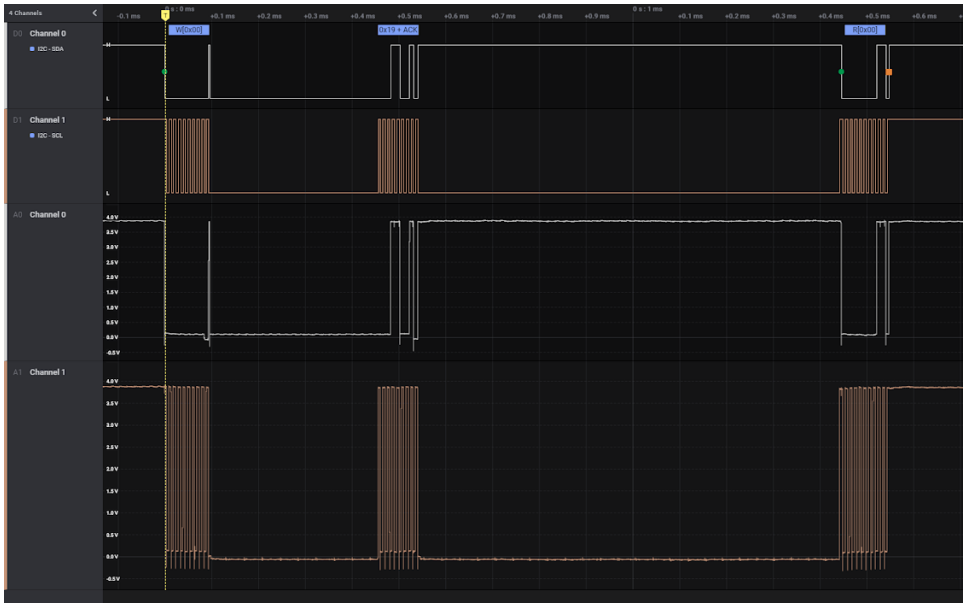
Company	Personal Use
Name	Devin
City/State	/
Country	UNITED STATES
E-mail	[redacted]
TEL	[redacted]
Website	
Subject	Having trouble communicating with RPB-1600 over PMBUS
Buy From	Bought from TRC Electronics Inc
Message	Hello, I'm attempting to communicate with my RPB-1600 over PMBUS and am having some troubles. I'd be happy to send over some screenshots and additional details if I could talk to someone who has technical knowledge of PMBUS. Thanks!

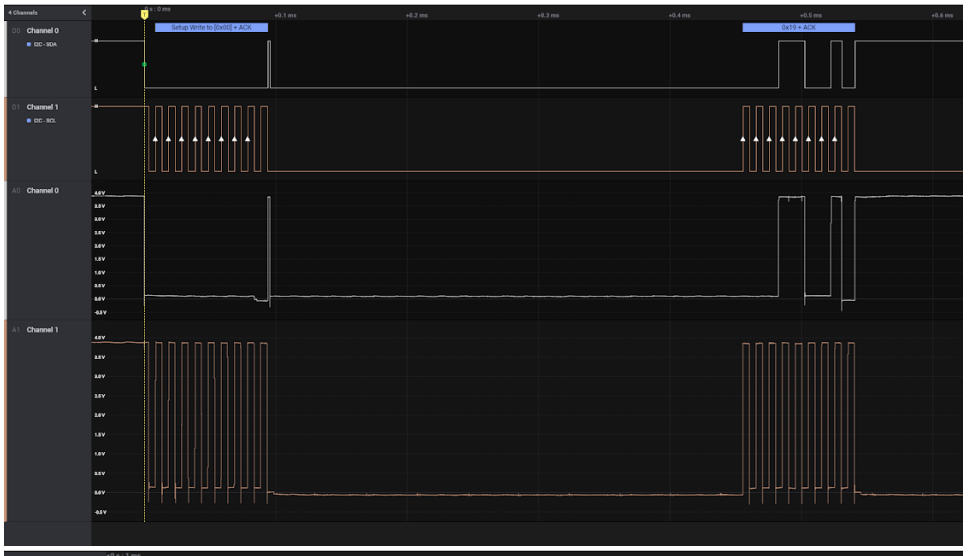
Devin Malanaphy <[redacted]> Tue, May 25, 2021 at 3:08 PM
To: Anke Ge <anke.ge@meanwellusa.com>

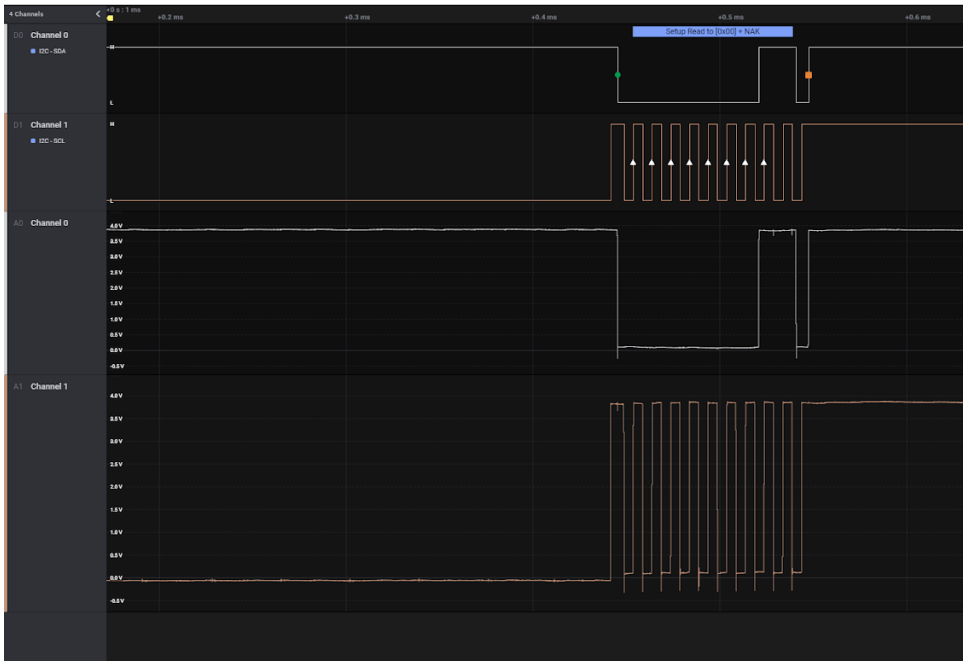
Yes! Thank you, hello. I'm attempting to communicate with the Meanwell RPB-1600 charger by replicating the PMBus protocol using a standard i2c library. I'm able to send the charger an address and command, which it acknowledges, but when I try to read data I get a NAK. See below some screenshots of a logic analyzer showing the i2c exchange. I'm testing with the "Capability" command, which is hex 0x19.

The first image below is a full attempt to read using the "capability" command. The top two signals are the digital SDA and SCL respectively, and the bottom two signals are the analog view of the same things. First I write the 7 bit address, which is 0. The address is set using the three address pins on the charger, which are all grounded to produce an address of 0. The charger responds to this with an ACK. Then, I write the command which in this case is 0x19. The charger acknowledges this message as well. The second image below shows a zoomed in view of these two messages, the annotation above the messages is helpful for figuring out what's what. Next I attempt to read from the charger by sending the address 0x00, with the read/write bit set to read. The charger rejects this read request by "NAKing" it. There's a zoomed in picture of this below as well (image 3). I'm wondering what part of this transaction is incorrect, and what I can do to correct it. I've confirmed the address is correct because when I attempt to write to an address other than 0x00 it NAKs immediately. I've also confirmed the clock speed is correct, at 100kHz. Any help would be greatly appreciated.

Thanks!
Devin Malanaphy







[Quoted text hidden]

Anke Ge <anke.ge@meanwellusa.com> Tue, May 25, 2021 at 9:13 PM
To: Devin Malanaphy [redacted]

Please note PMbus has a 0x00 address reserved. Please try using 0x40 as the address again, and let me know if that works for you

Best Regards,

Anke Ge
Technical Support Engineer | Eng. Dept.
T: 510-683-8886 ext.210 or 669-297-0797
MEAN WELL USA, INC. | w: www.meanwellusa.com | E: anke.ge@meanwellusa.com
SDG related companies : MEAN WELL | POWERNEX | SHARE WELL | FOUNDATION
Pioneering Global Standard Power Supply Industry

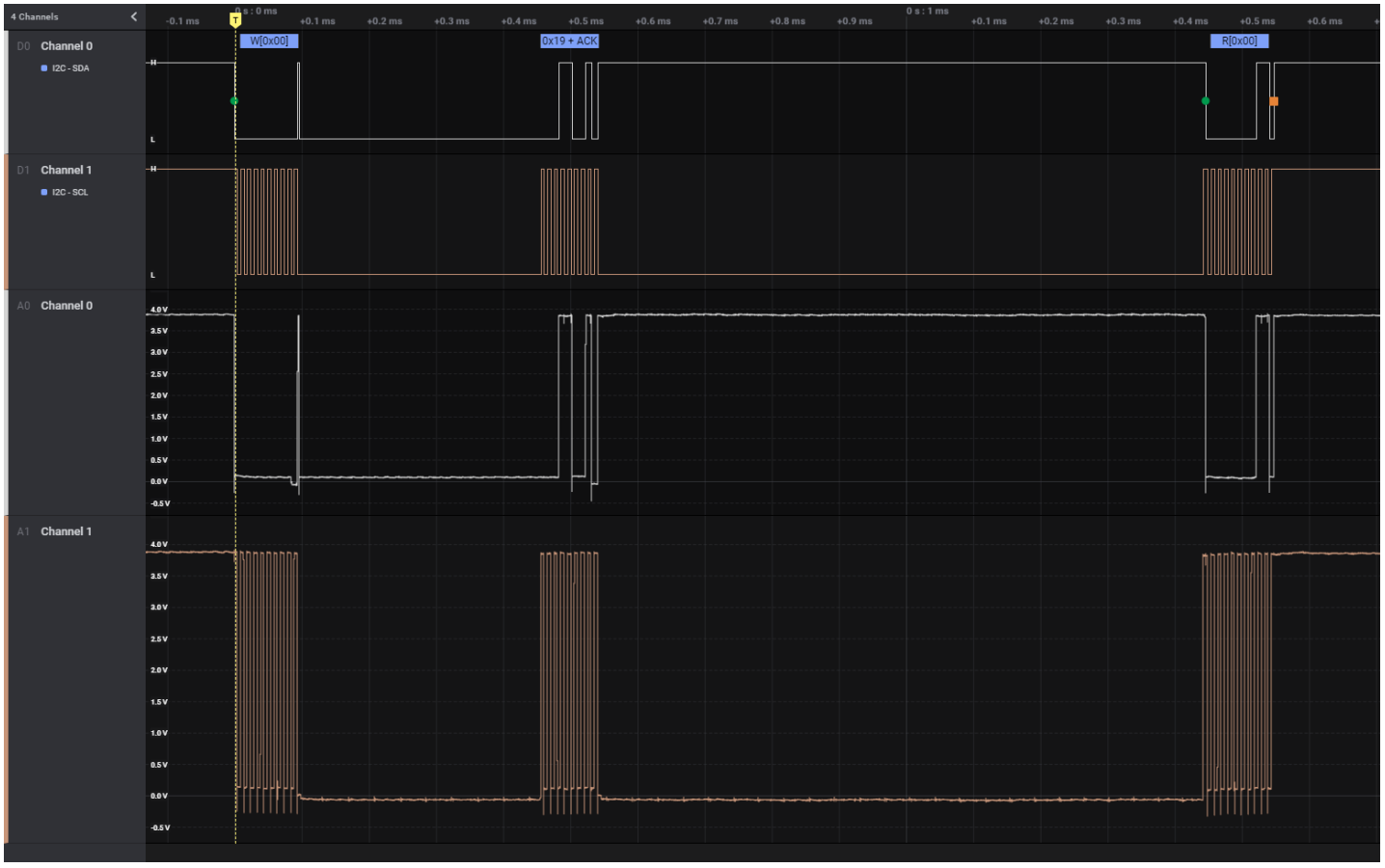
From: Devin Malanaphy [redacted]
Sent: Tuesday, May 25, 2021 12:09 PM
To: Anke Ge <anke.ge@meanwellusa.com>
Subject: Re: 【Technical Service】MEAN WELL Switching Power Supply

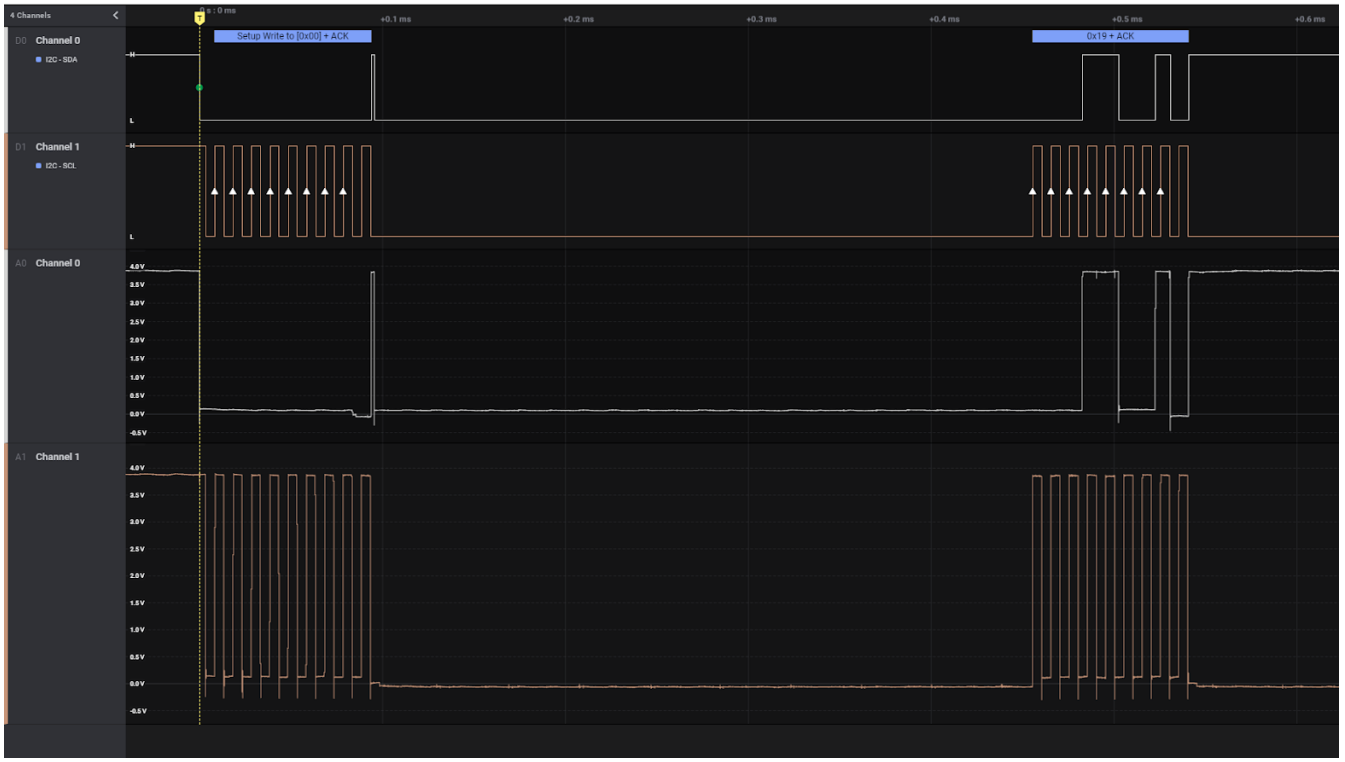
Yes! Thank you, hello. I'm attempting to communicate with the Meanwell RPB-1600 charger by replicating the PMBus protocol using a standard i2c library. I'm able to send the charger an address and command, which it acknowledges, but when I try to read data I get a NAK. See below some screenshots of a logic analyzer showing the i2c exchange. I'm testing with the "Capability" command, which is hex 0x19.

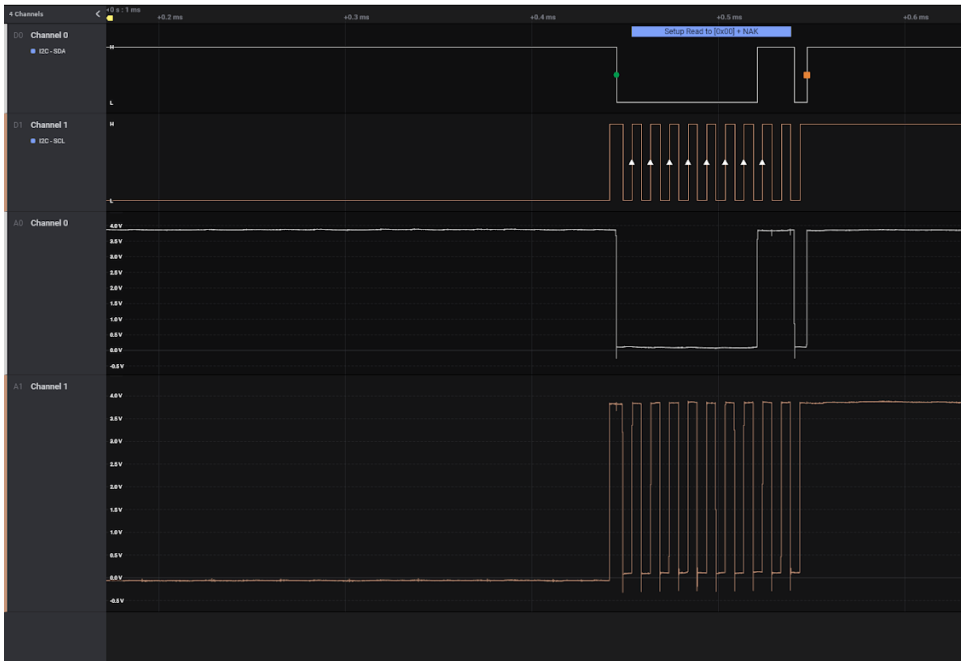
The first image below is a full attempt to read using the "capability" command. The top two signals are the digital SDA and SCL respectively, and the bottom two signals are the analog view of the same things. First I write the 7 bit address, which is 0. The address is set using the three address pins on the charger, which are all grounded to produce an address of 0. The charger responds to this with an ACK. Then, I write the command which in this case is 0x19. The charger acknowledges this message as well. The second image below shows a zoomed in view of these two messages, the annotation above the messages is helpful for figuring out what's what. Next I attempt to read from the charger by sending the address 0x00, with the read/write bit set to read. The charger rejects this read request by "NAKing" it. There's a zoomed in picture of this below as well (image 3). I'm wondering what part of this transaction is incorrect, and what I can do to correct it. I've confirmed the address is correct because when I attempt to write to an address other than 0x00 it NAKs immediately. I've also confirmed the clock speed is correct, at 100kHz. Any help would be greatly appreciated.

Thanks!

Devin Malanaphy


4 Channels
D0 Channel 0
I2C - SDA
D1 Channel 1
I2C - SCL
A0 Channel 0
A1 Channel 1
W[0x00]
0x19 + ACK
R[0x00]


4 Channels
D0 Channel 0
I2C - SDA
D1 Channel 1
I2C - SCL
A0 Channel 0
A1 Channel 1
Setup Write to [0x00] + ACK
0x19 + ACK



[Quoted text hidden]

Devin Malanaphy [redacted] Wed, May 26, 2021 at 9:04 AM
To: Anke Ge <anke.ge@meanwellusa.com>

That did it! Thanks so much for your help!

-Devin
[Quoted text hidden]

Devin Malanaphy [redacted] Fri, May 28, 2021 at 2:07 PM
To: Anke Ge <anke.ge@meanwellusa.com>

Anke,

I'm running into another weird issue as I try to read values from & program the charger. When I read the CURVE_CV and CURVE_FV commands I get 0V for both. The raw data that I receive for those commands is 0x00 for the low byte, and 0x48 for the high byte. Decoding this in the linear data format, I get an N of 9 (the datasheet indicates it should be N= -9) and a "Y" value (or mantissa) of 0. When I attempt to write to these values they don't change when I attempt to read them back. Additionally, when I attempt to modify the CURVE_CONFIG (command 0xB4) by writing 0x0044 (to change to a 2 stage charging cycle) the charger ACKs all of the bytes I write, but the values aren't changed when I read them back.

Any help or suggestions would be greatly appreciated.

Thanks so much!
Devin
[Quoted text hidden]

Devin Malanaphy [redacted] Fri, May 28, 2021 at 3:11 PM
To: Anke Ge <anke.ge@meanwellusa.com>

Anke,

Interestingly, I just discovered I can in fact set the CURVE_CC value and the change is reflected when I perform a read, and the value holds over a power cycle.

-Devin
[Quoted text hidden]

Anke Ge <anke.ge@meanwellusa.com> Fri, May 28, 2021 at 10:34 PM
To: Devin Malanaphy [redacted]

Hi Devin,

Please note all charging curve settings are saved in EEPROM, and the charger will recall them every time it starts. Also, 0x4800 should be 36 if my math is correct. It is DEC(0x4800)/512.

Could you let me know if the battery connected when you attempt to change the charging voltage?

Charging Curve

When it is opted for charging curve, D0 set to 0, charging curve function is enabled with additional PMBus commands. There are 4 built-in charging curves, "default" curve, one pre-defined curve for "gel battery", one pre-defined curve for "flooded battery" and one pre-defined curve for "AGM battery". Each curve can be selected by Command B4h CURVE_CONFIG.

In addition, users are able to customize their own charge curves, which will be stored to "default" after modification. CV can be set by Command B1h CURVE_CV ; FV can be set by Command B2h CURVE_FV ; Charge current level of stage1 can be set by Command B0h CURVE_CC; Taper current level from stage2 to stage3 can be set by Command B3h CURVE_TC. Please refer to the following PMBus Command List in table 8-2 for detailed information on commands and parameters.

NOTE: 1. The updated charging parameters is saved into EEPROM. The updated charging curve takes effect after RPB-1600 is restarted.

2. When charging curve is enabled, the following commands will be invalid while other PMBus commands are effective: Command 01h OPERATION (regarding Remote ON-OFF function), Command 22h VOUT_TRIM (regarding Output voltage programming function) and Command 46h IOUT_OC_FAULT_LIMIT (regarding Output current programming function).

[Quoted text hidden]

Devin Malanaphy [redacted] Sat, May 29, 2021 at 3:29 PM
To: Anke Ge <anke.ge@meanwellusa.com>

Anke,

Thank you for your prompt reply. I apologize but I'm not very familiar with the PMBus specification or how to calculate the bytes to send to the charger. I'm trying to follow the linear data format as it's outlined in this document.

PMBus Power System Mgt Protocol Specification – Part II – Revision 1.1

7.1. Linear Data Format

The Linear Data Format is typically used for commanding and reporting the parameters such as (but not only) the following:

- Output Current,
- Input Voltage,
- Input Current,
- Operating Temperatures,
- Time (durations), and
- Energy Storage Capacitor Voltage.

The Linear Data Format is a two byte value with:

- An 11 bit, two's complement mantissa and
- A 5 bit, two's complement exponent (scaling factor).

The format of the two data bytes is illustrated in Figure 4.

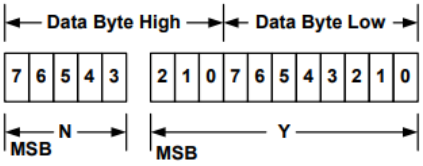


Figure 4. Linear Data Format Data Bytes

The relation between *Y*, *N* and the "real world" value is:

$X = Y \cdot 2^N$

Where, as described above:

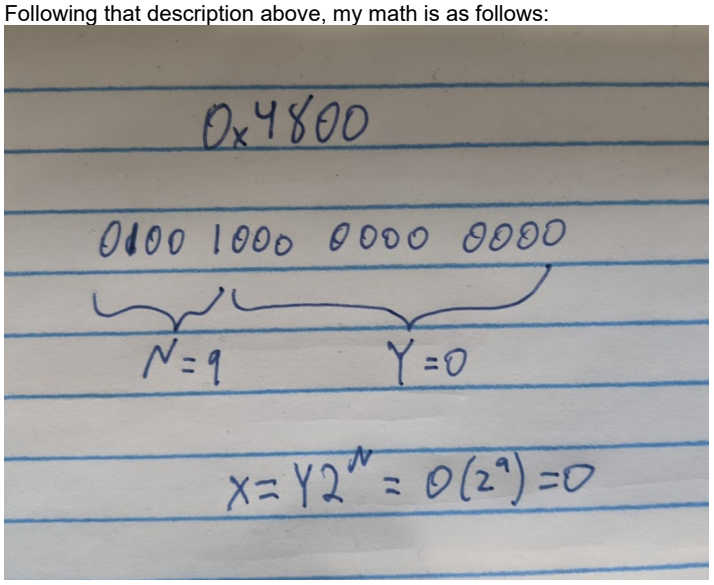
X is the "real world" value;

Y is an 11 bit, two's complement integer; and

N is a 5 bit, two's complement integer.

Devices that use the Linear format must accept and be able to process any value of *N*.

Following that description above, my math is as follows:



I have not connected the battery yet, I've just been adjusting the parameters with the battery disconnected. Another odd behavior is that I can't seem to adjust the parameters inside CURVE_CONFIG (Command 0xB4). When I write to them the charger ACKs the i2c bytes but the values haven't changed when I attempt to read the parameters back.

As always, any help is greatly appreciated.

Thanks,
Devin

[Quoted text hidden]

Anke Ge <anke.ge@meanwellusa.com> Tue, Jun 1, 2021 at 3:11 PM
To: Devin Malanaphy <[redacted]

Hi Devin,

Please note that RPB-1600 uses a direct data format so that the entire 2 bytes are data. I tested an RPB-1600-48, and CURVE_FV (Command Code B2) reads 0x676E (51.7V)

I also tried to overwrite the CURVE_CONFIG to 0x4040, and no problem was found. I could read CURVE_CONFIG back as 0x4040 immediately and after a restart. I would recommend double-checking your data format in your code.

Please let me know if I can be of further help

Best Regards,

Anke Ge
Technical Support Engineer | Eng. Dept.
T: 510-683-8886 ext.210 or 669-297-0797
MEAN WELL USA, INC. | w: www.meanwellusa.com | E: anke.ge@meanwellusa.com
SDG related companies : MEAN WELL | POWERNEX | SHARE WELL | FOUNDATION
Pioneering Global Standard Power Supply Industry

From: Devin Malanaphy [redacted]
Sent: Saturday, May 29, 2021 12:30 PM
To: Anke Ge <anke.ge@meanwellusa.com>
Subject: Re: 【Technical Service】MEAN WELL Switching Power Supply

Anke,

Thank you for your prompt reply. I apologize but I'm not very familiar with the PMBus specification or how to calculate the bytes to send to the charger. I'm trying to follow the linear data format as it's outlined in this document.

PMBus Power System Mgt Protocol Specification – Part II – Revision 1.1

7.1. Linear Data Format

The Linear Data Format is typically used for commanding and reporting the parameters such as (but not only) the following:

- Output Current,
- Input Voltage,
- Input Current,
- Operating Temperatures,
- Time (durations), and
- Energy Storage Capacitor Voltage.

The Linear Data Format is a two byte value with:

- An 11 bit, two's complement mantissa and
- A 5 bit, two's complement exponent (scaling factor).

The format of the two data bytes is illustrated in Figure 4.

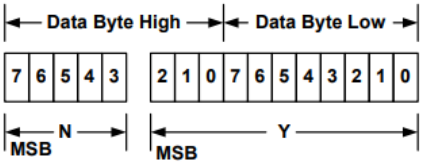


Figure 4. Linear Data Format Data Bytes

The relation between *Y*, *N* and the "real world" value is:

$X = Y \cdot 2^N$

Where, as described above:

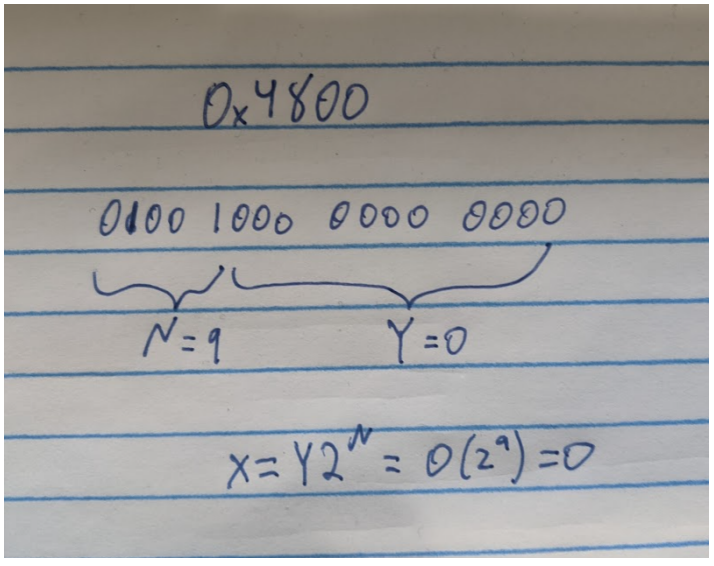
X is the "real world" value;

Y is an 11 bit, two's complement integer; and

N is a 5 bit, two's complement integer.

Devices that use the Linear format must accept and be able to process any value of *N*.

Following that description above, my math is as follows:



[Quoted text hidden]
[Quoted text hidden]

Devin Malanaphy [redacted] Wed, Jun 2, 2021 at 12:59 PM
To: Anke Ge <anke.ge@meanwellusa.com>

Anke,

I really appreciate you testing this out for me. I was able to verify that my setup was not correctly parsing the response from CURVE_CONFIG and it was in fact writing/recalling the correct values. I was also able to modify/parse my CURVE_CV & CURVE_FV to match the "Direct data format" you described in your email, however I'm still a little confused.

The RPB-1600 datasheet mentions the data format for all of the commands, and for the CURVE_CONFIG commands it specifies them as a linear data format

Valid when charging according to charge curve (D0=0)

Command Code	Command Name	Transaction Type	# of data Bytes	Description
B0h	CURVE_CC	R/W Word	2	Constant current setting value of charging curve (format: Linear, N= -2)
B1h	CURVE_CV	R/W Word	2	Constant voltage setting value of charging curve (format: Linear, N= -9)
B2h	CURVE_FV	R/W Word	2	Constant voltage setting value of charging curve (format: Linear, N= -9)
B3h	CURVE_TC	R/W Word	2	Taper current setting value of charging curve (format: Linear, N= -2)
B4h	CURVE_CONFIG	R/W Word	2	Configuration setting of charging curve
B5h	CURVE_CC_TIMEOUT	R/W Word	2	CC stage timeout setting value of charging curve (format: Linear, N= 0)
B6h	CURVE_CV_TIMEOUT	R/W Word	2	CV stage timeout setting value of charging curve (format: Linear, N= 0)
B7h	CURVE_FLOAT_TIMEOUT	R/W Word	2	Floating timeout setting value of charging curve (format: Linear, N= 0)
B8h	CHG_STATUS	READ Word	2	Charger's status reporting

Additionally, the VOUT_MODE command returns 0x17, which decodes to Mode == Linear and an exponent of -9 for output voltage related commands. See below the PMBus 1.1 section that covers this command.

8.2. VOUT_MODE Command

The data byte for the VOUT_MODE command is one byte that consists of a three bit Mode and a five bit Parameter as shown in Figure 5. The three bit Mode sets whether the device uses the Linear, VID or Direct modes for output voltage related commands. The five bit Parameter provides more information about the selected mode, such as which manufacturer's VID codes are being used.

Sending the VOUT_MODE command with the address set for writing sets the Mode and Parameter into the PMBus device, if it accepts changes to these values.

PMBus devices may have the Mode and Parameter set at the time of manufacture and may not permit the user to change these values. In this case, if a host send a VOUT_MODE command for a write to a PMBus device, the device shall reject the VOUT_MODE command, declare a communication fault for invalid data, and respond as described in section 10.2.2.

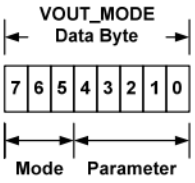


Figure 5. VOUT_MODE Command Data Byte Structure

If a device accepts the VOUT_MODE command, the Mode and Parameter are retained until changed with another VOUT_MODE command or until the bias power is removed.

Sending the VOUT_MODE command using the SMBus Read Byte protocol returns one byte with the Mode and Parameter as shown in Figure 5.

Table 2 shows the permitted values and format of the VOUT_MODE data byte. More information on the VOUT_MODE command is used with output voltage related commands is given below in Section 8.3.

Table 2. Summary Of The VOUT_MODE Data Byte Format

Mode	Bits [7:5]	Bits [4:0] (Parameter)
Linear	000b	Five bit two's complement exponent for the mantissa delivered as the data bytes for an output voltage related command.
VID	001b	Five bit VID code identifier per
Direct	010b	Always set to 00000b

From my limited testing it seems as though the charger reports certain values (such as CURVE_CV, CURVE_FV, etc.) in a "direct data" format where N = -9 and others, such as READ_FAN_SPEED_1, READ_FAN_SPEED_2, CURVE_CC, etc. in the linear data format with various N values (N=5 for the fan speed, -2 for the constant current setting, etc.).

Once again, your assistance throughout this has been incredibly helpful and I really appreciate you taking your time to help explain all of this to me.

Thanks,
Devin

[Quoted text hidden]

Anke Ge <anke.ge@meanwellusa.com> Wed, Jun 2, 2021 at 4:49 PM
To: Devin Malanaphy [redacted]

Hi Devin,

Sorry, I'm not an expert on PMbus. I think CURVE_CV and CURVE_FV are considered output voltage related parameters, so their linear format is the following

8.3.1. Linear Mode

The data bytes for the VOUT_MODE and VOUT_COMMAND when using the Linear voltage data format are shown in Figure 6.

Note that the VOUT_MODE command is sent separately from output voltage related commands and only when the output voltage format changes. VOUT_MODE is not sent every time an output voltage command is sent.

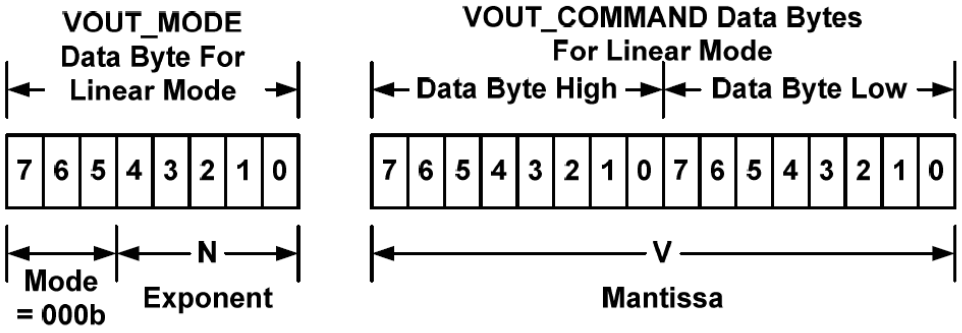


Figure 6. Linear Format Data Bytes

The Mode bits are set to 000b.

The Voltage, in volts, is calculated from the equation:

$$Voltage = V \bullet 2^{N}$$

For other parameters such as current, and temperature readings are linear data format as described in 7.1 of PMbus specification.

Please let me know if this makes sense. Sorry about the confusion

Best Regards,

Anke Ge
Technical Support Engineer | Eng. Dept.
T: 510-683-8886 ext.210 or 669-297-0797
MEAN WELL USA, INC. | w: www.meanwellusa.com | E: anke.ge@meanwellusa.com
SDG related companies : MEAN WELL | POWERNEX | SHARE WELL | FOUNDATION
Pioneering Global Standard Power Supply Industry

From: Devin Malanaphy [redacted]
Sent: Wednesday, June 2, 2021 10:00 AM
To: Anke Ge <anke.ge@meanwellusa.com>
Subject: Re: 【Technical Service】MEAN WELL Switching Power Supply

Anke,

I really appreciate you testing this out for me. I was able to verify that my setup was not correctly parsing the response from CURVE_CONFIG and it was in fact writing/recalling the correct values. I was also able to modify/parse my CURVE_CV & CURVE_FV to match the "Direct data format" you described in your email, however I'm still a little confused.

The RPB-1600 datasheet mentions the data format for all of the commands, and for the CURVE_CONFIG commands it specifies them as a linear data format

Valid when charging according to charge curve (D0=0)

Command Code	Command Name	Transaction Type	# of data Bytes	Description
B0h	CURVE_CC	R/W Word	2	Constant current setting value of charging curve (format: Linear, N= -2)
B1h	CURVE_CV	R/W Word	2	Constant voltage setting value of charging curve (format: Linear, N= -9)
B2h	CURVE_FV	R/W Word	2	Constant voltage setting value of charging curve (format: Linear, N= -9)
B3h	CURVE_TC	R/W Word	2	Taper current setting value of charging curve (format: Linear, N= -2)
B4h	CURVE_CONFIG	R/W Word	2	Configuration setting of charging curve
B5h	CURVE_CC_TIMEOUT	R/W Word	2	CC stage timeout setting value of charging curve (format: Linear, N= 0)
B6h	CURVE_CV_TIMEOUT	R/W Word	2	CV stage timeout setting value of charging curve (format: Linear, N= 0)
B7h	CURVE_FLOAT_TIMEOUT	R/W Word	2	Floating timeout setting value of charging curve (format: Linear, N= 0)
B8h	CHG_STATUS	READ Word	2	Charger's status reporting

Additionally, the VOUT_MODE command returns 0x17, which decodes to Mode == Linear and an exponent of -9 for output voltage related commands. See below the PMBus 1.1 section that covers this command.

8.2. VOUT_MODE Command

The data byte for the VOUT_MODE command is one byte that consists of a three bit Mode and a five bit Parameter as shown in Figure 5. The three bit Mode sets whether the device uses the Linear, VID or Direct modes for output voltage related commands. The five bit Parameter provides more information about the selected mode, such as which manufacturer's VID codes are being used.

Sending the VOUT_MODE command with the address set for writing sets the Mode and Parameter into the PMBus device, if it accepts changes to these values.

PMBus devices may have the Mode and Parameter set at the time of manufacture and may not permit the user to change these values. In this case, if a host send a VOUT_MODE command for a write to a PMBus device, the device shall reject the VOUT_MODE command, declare a communication fault for invalid data, and respond as described in section 10.2.2.

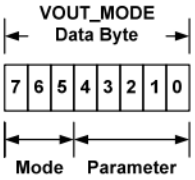


Figure 5. VOUT_MODE Command Data Byte Structure

If a device accepts the VOUT_MODE command, the Mode and Parameter are retained until changed with another VOUT_MODE command or until the bias power is removed.

Sending the VOUT_MODE command using the SMBus Read Byte protocol returns one byte with the Mode and Parameter as shown in Figure 5.

Table 2 shows the permitted values and format of the VOUT_MODE data byte. More information on the VOUT_MODE command is used with output voltage related commands is given below in Section 8.3.

Table 2. Summary Of The VOUT_MODE Data Byte Format

Mode	Bits [7:5]	Bits [4:0] (Parameter)
Linear	000b	Five bit two's complement exponent for the mantissa delivered as the data bytes for an output voltage related command.
VID	001b	Five bit VID code identifier per
Direct	010b	Always set to 00000b

[Quoted text hidden]
[Quoted text hidden]