Application Note RRC-PMM240

Power Management Module
for Mobile Applications
P/N:100576

Scope

This application note is supplemental to the PMM240 specification document. It goes into the details of using the PMM240 to give you a better understanding of its operation. If you cannot find the information you need in either of these documents, please contact your local salesperson or email applicationsupport@rrc-ps.de.

1. Getting started with the PMM240

There are two essential steps to perform before connecting a battery:

- Select an appropriate power supply.
- Adjust the Input Current Limit according to the power supply, the User Application, and the battery.

This section shows how to perform these crucial steps, including an example. Figure 1 shows the relation between the Power Supply, the Input Current Limit, and the User Application. You can find further information in the PMM240 specification document.

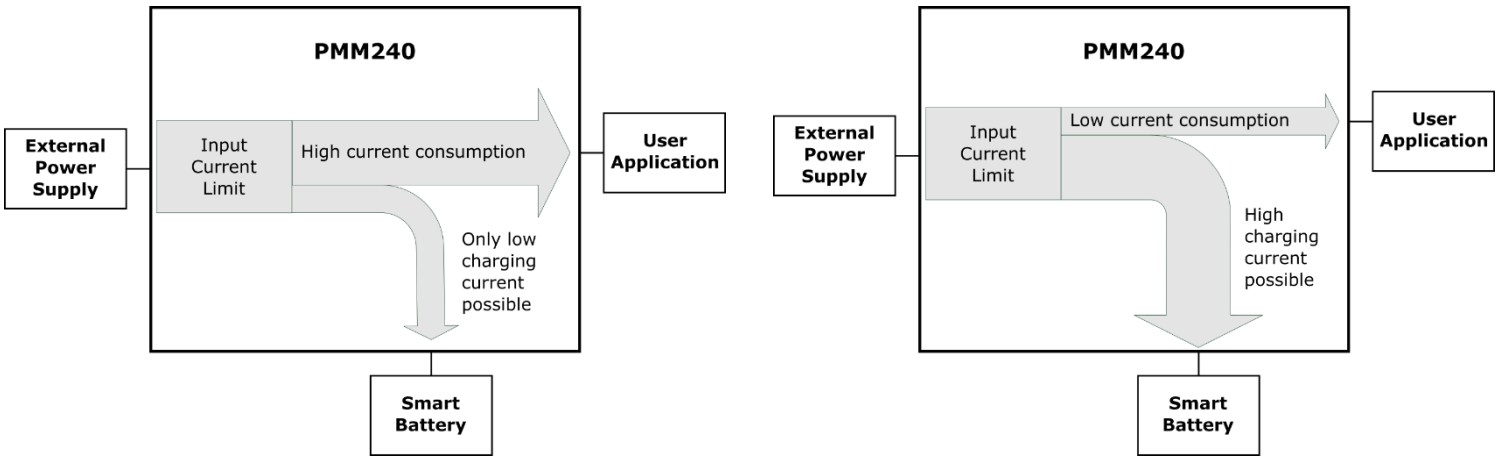


Figure 1 – The Input Current Setting adjusts the charging current depending on the User Application's current consumption.

1.1. Choice of Power Supply

Make sure to have an appropriate External Power Supply Unit (PSU). It is essential that the output voltage V_{PSU} is in the range of 7.5 V – 24 V, but at least 1 V higher than the battery's maximum charging voltage. Table 1 lists recommendations for output voltages of different smart batteries.

Battery architecture (ySxP) (y cells in series, x cells in parallel)	Typical PSU voltage
2SxP (e.g. RRC2037, RRC2057)	12 V
3SxP (e.g. RRC2020, RRC2040)	15 V
4SxP (e.g. RRC2024, RRC2054)	19 V

Table 1 – Recommended input voltage for different batteries.

The output current I_{PSU} shall be at least

$$I_{PSU} > 1.1 \cdot \left(\frac{I_{Battery} \cdot V_{Battery}}{V_{PSU}} + I_{UserApp} \right) \quad (1)$$

Where:

V_{PSU} = External Power Supply voltage
I_{PSU} = External Power Supply current
$V_{Battery}$ = maximum battery charging voltage
$I_{Battery}$ = maximum battery charging current
$I_{UserApp}$ = maximum current at V_{PSU} needed by the User Application

If you cannot ensure that I_{PSU} is always high enough according to the above equation, the power supply can overload, and the PMM240 can get damaged. In that case: Be sure to limit the "Input Current Limit" as described on the following pages *before* connecting a battery.

Please note that RRC offers its own AC adapters. Every AC adapter listed in Table 2 can be used with a PMM240, and they are suited to charge all RRC batteries. "M" devices are certified for medical applications, as well.

Device Name	Part number	Charging Power	Output voltage V_{PSU}	Output Current I_{PSU}
PS65	210919	65 W	19 V	3.42 A
PS65M	211249	65 W	19 V	3.43 A
PS90M	211380	90 W	24 V	3.75 A

Table 2 – AC adapters suited to charge all RRC batteries via a PMM240.

Let's take an example where the User Application demands a maximum current[1] of 1.5 A.
The battery used in this example is an RRC2054.

Figure 2 – RRC2054 label

The RRC2054 charging voltage is 16.8 V (see the label in Figure 2). According to Table 1, the recommended voltage for an External Power Supply would be 19 V. Ensure that the Power Supply delivers more than the current needed by the User Application, which in this example is 1.5 A.

Let's choose an RRC PS65 AC adapter. It delivers up to 65 W and an output current of 3.42 A (see Table 2). According to formula (1), we review if the PS65 is strong enough under full load:

$$I_{PSU} > 1.1 \cdot \left(\frac{I_{Battery} \cdot V_{Battery}}{V_{PSU}} + I_{UserApp}\right) = 1.1 \cdot \left(\frac{2.415\,A \cdot 16.8\,V}{19\,V} + 1.5\,A\right) = 4\,A \quad (2)$$

At first glance, the PS65 doesn't seem to be powerful enough as it delivers only 3.42 A. But, it still is a good choice because of its excellent price-to-value ratio. The PS65 will work if the User Application doesn't always need 1.5 A or the battery doesn't need to be charged with maximum charge current. But: We need to set the Input Current Limit.

[1] We are speaking of continuous current here. The inrush current, however, does not need to be taken into account, as long as you don't load the output of the PMM240 with hundreds of µF.

1.2. Determine the Input Current Limit

If I_{PSU} does not meet the requirements in equation (1), the power supply can be overloaded, and the PMM240 can be damaged. In that case, you must set an Input Current Limit. Use the following equation to determine the Input Current Limit value:

$$I_{ICL} = I_{PSU} - 250\ \text{mA} \tag{3}$$

where:

I_{ICL} = Input Current Limit

I_{PSU} = Maximum current that the PSU can supply

The subtraction of 250 mA is needed because of tolerances in the input current measurement.

Let's return to our previous example. The Input Current Limit value I_{ICL} calculates to:

$$I_{ICL} = I_{PSU} - \text{tolerance} = 3420\ \text{mA} - 250\ \text{mA} = 3170\ \text{mA} \tag{4}$$

While the User Application idles, the charging current corresponds to ChargingCurrent() as defined by the battery, i. e. 2415 mA (refer to the label in Figure 2). The battery charges at full speed. Under full load, however, the charging current typically may go down to

$$I_{ICL} - I_{UserApp} = 3170\ \text{mA} - 1500\ \text{mA} = 1670\ \text{mA}. \tag{5}$$

You see that the battery still gets charged, but at a slower rate.

In the PMM240, the current limit can be configured only with a minimum step size of 256 mA to 296 mA. This means that the I_{ICL} value may not be exactly what you have programmed, because the PMM240 rounds down to the next possible smaller value.

For example, if you send I_{ICL}=3170 mA. Suppose the step size is 256 mA, and because 3170/256 = 12.38, the exact value of the current limit that is programmed into the PMM240 will be 12 x 256 mA = 3072 mA.

In addition, there are tolerances of about ±250 mA.

Because of that step size, if the Input Current Limit is set too low, the register value could round down to 0 mA, and the PMM240 won't charge the battery. To avoid this issue, the Input Current Limit should always be configured above 285 mA.

In our example, the step size and the tolerances make that while I_{ICL} = 3170 mA, the actual range of the current limit can be between

$$I_{ICL,max} = I_{ICL} + 250\ \text{mA} = 3170\ \text{mA} + 250\ \text{mA} = 3420\ \text{mA} \tag{6}$$

and

$$I_{ICL,min} = I_{ICL} - 295\ \text{mA} - 250\ \text{mA} = 3170\ \text{mA} - 295\ \text{mA} - 250\ \text{mA} = 2625\ \text{mA} \tag{7}$$

1.3. Configure the Input Current Limit

You can quickly set up and program the PMM240 via SMBus / I^2C using any USB-to-I^2C converter. Refer to section 4 "Supplemental information on the SMBus communication" for details on how to send an SMBus command to the PMM240.

You can store the I_{ICL} value in the RAM and/or into the EEPROM. Values written to RAM are valid only until the next PMM240 startup. At every reboot, the value written to the EEPROM is transferred to the RAM. Hence, if you want to keep your settings beyond the next reboot, write the value into the EEPROM.

1. To write to the PMM240 registers, connect the User Application or a USB-to-I²C converter to the PMM240. (for details on connector pinouts and mating connectors, refer to the PMM specification document).

2. Do not connect the battery! If you connected the battery before setting the Input Current Limit, you would risk overloading the Power Supply and damaging the PMM240.

3. Connect the Power Supply to the DC Input of the PMM240.

4. Determine the I²C command using https://www.rrc-ps.com/pmm240-calculator.
   In our example, you would send this sequence of I²C Writes to write the precalculated I_{ICL} value into the EEPROM:

Address	Command	Byte Length	Config	3170 mA		PEC
0x20	0x3d	0x03	0x01	0x0c	0x62	0x1d

5. Restart the PMM240 by removing and reconnecting the Power Supply. This will load the programmed values from the EEPROM into RAM. The PMM240 is now configured, and the battery may be connected.

1.4. Verify the new configuration

"Input Current Limit" and "Charge Current Limit" are write-only registers. To verify your configuration, you need to measure the current under different load situations.

As shown in Figure 3,

- Measure the current I_{PSU} going into the PMM240.
- Connect the User Interface to read out the battery via SMBus. You could use a tool like the EV2400 from TI.
- And instead of the User Application, attach an electronic load to the output of the PMM240.

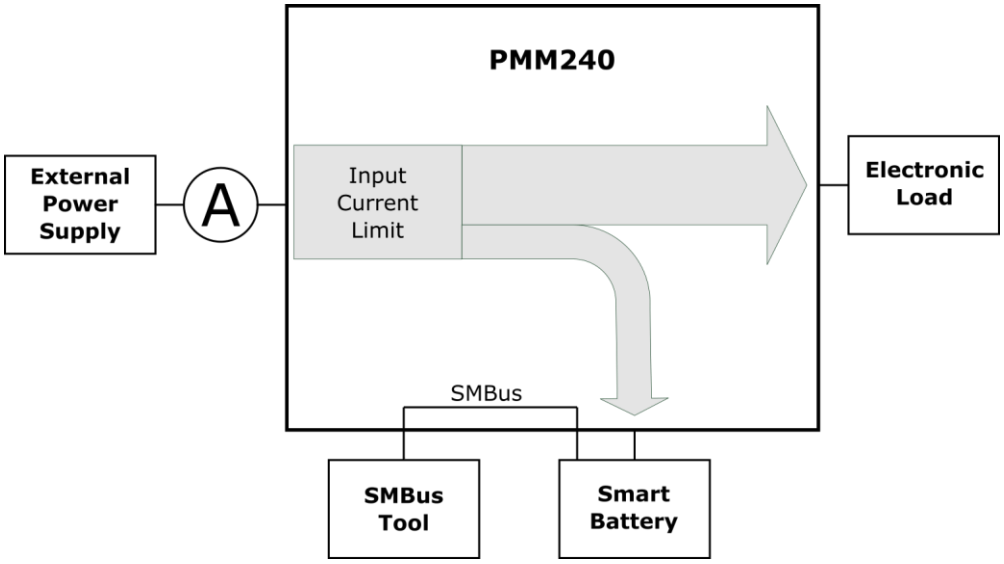


Figure 3 – To verify the register settings, add an Ampere meter, an Electronic Load, an empty battery, and an SMBus Tool.

Now, follow these steps to verify the register settings:

1. Set the electronic load to I_{UserApp} = 0 A and connect an empty battery. Switch on the input power supply.
2. Measure the current I_{PSU} going into the PMM240; Also read the battery registers Current() and Voltage() to retrieve I_{Battery} and V_{Battery}.

   The relation should be: $I_{\text{PSU}} = I_{\text{Battery}} \cdot \frac{V_{\text{Battery}}}{V_{\text{PSU}}} \cdot \eta$ (8)

   with η being the efficiency of the PMM240, typically having a value between 85 and 95%.
3. The charging current should not be higher than the Charging Current Limit setpoint[2].
4. Slowly change the electronic load to increase the output current I_{UserApp}.
   The input current I_{PSU} should increase *by the same amount* until you reach the Input Current Limit setpoint[2].
5. If you would increase the output current further, the charging current I_{Battery} would decrease accordingly, while the input current I_{PSU} would remain approximately the same.[3]

[2] The setpoint must not be exactly to the value that you've configured:
The Current Limit is subject to tolerances and granularity, as mentioned in section 1.2, equations (6) and (7).

[3] If you would increase I_{UserApp} beyond the point where I_{Battery} drops to 0, I_{PSU} would increase again.
This is because the PMM240 can limit the input current only by adapting the charging current I_{Battery}.

2. Power Modes and Standby Current Flowchart

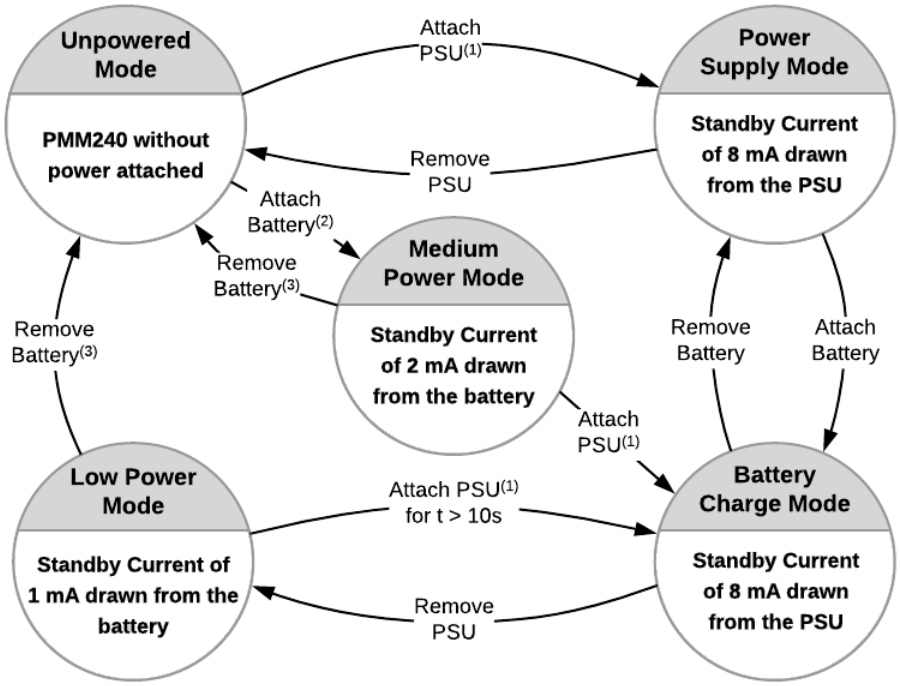


Figure 4 – This Flowchart shows the different power modes and their current consumption.

Explanation of the transition notes in the flowchart above:

(1) Attach the PSU (Power Supply Unit) with a valid input voltage of at least 1 V above the maximum charging voltage of the attached RRC smart battery.

(2) Attach a smart battery with a non-zero output voltage.
Batteries in shipping mode or deeply-discharged batteries won't result in a Power Mode change here!

(3) For this Power Mode change, instead of removing the battery, you can set it into shipping mode, too.

3. Charging two or more batteries

3.1. Parallel configuration

You can use any number of PMM240 connected in parallel. Figure 5 shows how:

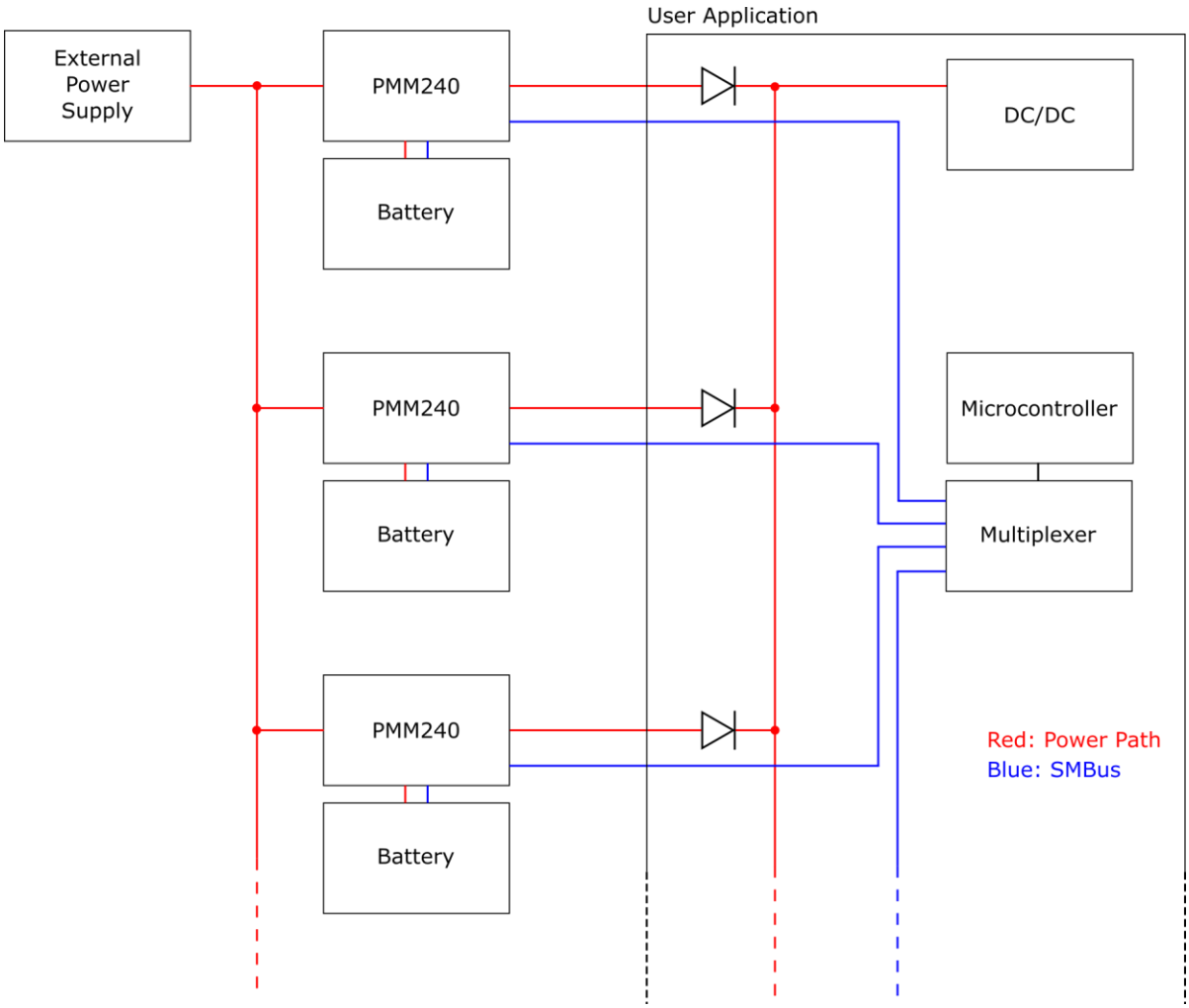


Figure 5 - Using PMM240 in a parallel configuration. The number of PMM240 is not limited.

- Use a dedicated PMM240 for each battery.
- Set the Input Current Limit for each PMM240. As you may use more than one PMM240 on one power supply, make sure that the external power supply is able to deliver the multiple of the configured input currents limits.
- To avoid back-driving currents, use ideal diodes (e.g., LM5050 from TI).
- Use a multiplexer (e.g., PCA9544a).
  This is necessary as per SMBus specification: All smart batteries share the same SMBus address.
- Take care of inrush currents, as they might not distribute equally among all PMM240s.
- To have balanced discharge rates between the batteries, ensure that the State-of-Charge of all batteries is approximately the same.
- Do not use batteries with considerably different "age" (the combination of the number of cycles and production dates).

Why is there a need to have equally charged batteries?

Imagine having 8 batteries in parallel from which 1 is full, and the 7 others are almost empty. In this extreme case, the full battery takes 100% of the load until its voltage falls at the level of the other 7 batteries. From that point on, all batteries share the current.

It is essential that the 1 battery that is fully charged does not fail while it supports the load "all alone".

At first, determine whether the most extreme case would even harm the 1 battery. If not, then there is no need for additional circuitry. If yes, consider using current limiting circuitry to limit the current under a no-harm level.

Depending on your application, you could also imagine doing a software solution: You could read out the SOC and the battery voltages. If the batteries are not balanced, you could consider throwing a message like "change battery xy" and wait until it's done before consuming more current.

3.2. Series configuration

While it is possible, **we don't recommend** using smart batteries in series. If any failure event leads to switching off one of the batteries in the series stack, there may be a transient over-voltage beyond our safety-tested levels, which could damage your batteries.

4. Supplemental information on the SMBus communication

You can find all the essential information on SMBus communication in the PMM240 specification document.

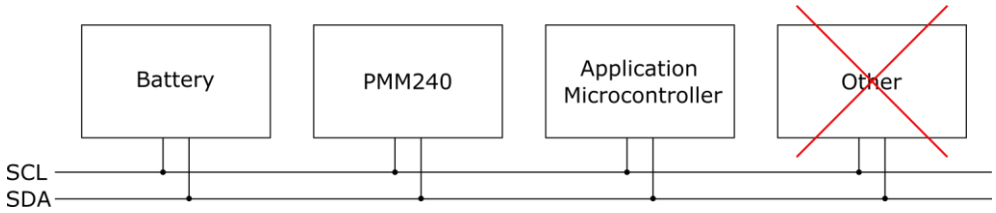
This section gives supplemental information that guides you in setting up the communication interface.

4.1. SMBus participants

Although you may use the PMM240 stand-alone, you can connect it – and the battery – to the User Application via the UI-connector. The User application can then configure the PMM240 and exchange status information with the battery.

It is crucial that:

- Only the battery, the PMM240 and the User Application may share the bus. On the side of the User Application, no other IC than the microcontroller must be connected to the bus. Any non-charging-related communication leads to dropouts on charging.



- The pull-up resistors are included on the PMM240. The User Application is not allowed to have additional pull-up resistors on these lines. If the User Application has its own pull-ups, remove them or add an SMBus Isolator (e. g. ISO1540) between the User Application and the PMM240.
- To avoid collisions, each master on the SMBus line must operate in a multi-master environment. You need to set up your microcontroller as multi-master by implementing the two steps below:
    - Don't interrupt ongoing communication. Wait for the stop condition, and only then initiate a message on the bus.
    - If two devices start to communicate on the bus simultaneously, the device that pulls the lines high first loses the arbitration and should immediately stop communicating on the bus.
- Once per second, the PMM240 needs to query the battery for status information. Give the PMM240 enough time to use the bus and exchange that information.

4.2. Checksum (PEC)

The last byte of each command is a Packet Error Checking (PEC) byte. It corresponds to the CRC-8 checksum over the leading 6 bytes of the same command. The polynomial used is $x^8 + x^2 + x + 1$ (= 0x107, which is the standard polynomial used for CRC-8).

We offer a tool to calculate the whole command, including the CRC8, on our website: https://www.rrc-ps.com/pmm240-calculator.

If you want to implement the calculation in your own software, the following website provides a good explanation about how the CRC-8 checksum works: http://www.sunshine2k.de/articles/coding/crc/understanding_crc.html. Tools like https://crccalc.com/ (make sure to check "Input Type" as "Hex") also provide a good start. Please note that RRC isn't linked to any of these websites and takes no responsibility for their content.

If you want to generate the PEC byte in your source code, you may also want to take this C snippet:

```
/**
* Bulk buffer CRC8 calculation without a lookup table
* (size optimized). Generic data types used.
*
* @param const void* data
* @param unsigned size
* @return unsigned char
*/
static unsigned char crc8(const void *data, unsigned size)
{
  const unsigned char *p = data;
  unsigned crc = 0;
  int i, j;
  for(j = size; j != 0; --j, ++p) {
    crc ^= ((*p) << 8);
    for(i = 8; i > 0; --i) {
      crc = ((crc & 0x8000) ? (crc ^ (0x1070 << 3)) : (crc)) << 1;
    }
  }
  return (unsigned char) (crc >> 8);
}
```

5. GPIO status

5.1. Read GPIO status via SMBus

It is not possible to read out the GPIO status *directly* from the PMM240. However, you can read BatteryStatus() from the battery and derive the same information. Table 3 and Table 4 below show you how. See the RRC2xxx Battery Design-in Guideline for more details on these registers.

GPIO1 status	Meaning	SMBus communication workaround
high	Battery is fully charged	Bit 5 from BatteryStatus() is set (=1)
blinking @ 0.5Hz	Battery is being charged	Bit 6 from BatteryStatus() is not set (=0)
blinking @ 2.5Hz	Battery error	BatteryStatus() shows miscellaneous errors that are reversible if the conditions change. If both the TCA and TDA bits are set, the battery has failed non-reversibly.
low	No battery connected or input voltage too low	No communication possible with the battery. Battery returns NACK when queried.

Table 3 – SMBus communication workaround for reading the status of GPIO1

GPIO2 status	Meaning	SMBus communication workaround
high	Power supply is present	Listen on the bus whether the PMM240 exchanges information with the battery approximately once per second.
Blinking @ 2.5Hz	Charger error (Table 6)	Board cannot charge the battery
low	Power supply is not present	Bit 6 from BatteryStatus() is set (=1)

Table 4 - SMBus communication workaround for reading the status of GPIO2

5.2. Conditions that trigger GPIO error status

If one GPIO shows an error status, refer to Table 5 (for GPIO1) and Table 6 (for GPIO2) for possible reasons that may have triggered that event.

	Error	Triggering Condition	Reset Measure
Irreversible Charger Failure	Bad Contact Failure	Power loss between smart battery connector and battery > 3W	Remove battery or power cycle
	No Charge Current	No current measured on the charger while the battery is being charged	Remove battery or power cycle
	Open Terminals	Smart battery connector voltage > Battery voltage (read from the battery)	Remove battery or power cycle
	Parameter Read	Permanent battery parameters read error	Remove battery or power cycle

Reversible Battery Errors	Temperature	Battery Temperature < -20°C(1)	Battery Temperature ≥ -20°C [4]
		Battery Temperature > 80°C(1)	Battery Temperature ≤ 80°C [2]
		Battery Over Temperature Alarm (OTA) set(1)	Reduce battery temperature [2]
		Charging interruption requested by the battery while T < 10°C or T > 40°C	Bring Battery Temperature to normal range
	Communication	No communication with the battery possible for 30 seconds during charge	Establish communication with battery during 30 seconds
Irreversible Battery Failure	Wake-up	No Communication with the battery possible after insertion for more than 15 minutes	Remove battery or power cycle
	Charge timeout	Battery not fully charged after 16 hours of charge	Remove battery or power cycle
	No Charge Current	Average battery current < 10mA during charge	Remove battery or power cycle
	Charging Interruption	Unexpected charging interruption requested by battery	Remove battery or power cycle
	Low State of Charge	Charging finished with low RSOC (< 80%)	Remove battery or power cycle
	Communication	Communication not possible during charging for more than 30min	Remove battery or power cycle

Table 5 – Battery error conditions as shown per GPIO1 error status.

Error	Condition	Reset
Reversible Charger Errors:		
Unexpected Voltage Error	Voltage > 3V at smart battery connector without battery detected for 10 seconds	Voltage ≤ 3V or battery detected within 10 seconds
Irreversible Charger Failures:		
Unexpected Voltage Failure	Voltage > 3V at smart battery connector without battery detected for 30 seconds	power cycle
Internal Failure	Charger internal communication failure	power cycle

Table 6 – Charger error conditions as shown per GPIO2 error status.

[4] Check the over temperature thresholds for charging and discharging of the battery used

6. Most common issues and how to prevent these

Please review the topics discussed in this section as these are widespread issues.

6.1. Hardware

- Don't connect I^2C chips to the SMBus.
- Don't put pull-up resistors on the SMBus.

For more information on the above points, see section 4.1 on page 11.

- Pay attention to inrush currents.

  The PMM240 **does not limit** any inrush current required by the User's Application (even with Input Current Limit set). The inrush peak current could damage the input MOSFET due to limited switching speed. Too much inrush current may happen

  - If you have a capacitor of several hundreds of µF connected to the output of the PMM240.
  - If you load the PMM240 with a small-in-value power resistor for testing purposes.

- Use plastic screws or plastic washers to fix the PMM240 on the mounting holes.

  Metal screws or metal washers may lead to short circuits. The green solder mask on the PMM240 is not insulating!

- Ensure that the User cannot connect the battery in the wrong way! See the PMM240 specification document for details on the insertion direction.

6.2. Software

- Configure your I^2C ports as open-drain.
- Make your SMB host multi-master capable.
- Don't communicate too often (do not stuff the bus) – especially if for any reason you cannot meet the multi-master requirement mentioned above.

For more information on the above points, see section 4.1 on page 11.

- Set the Input Current Limit. See the "Getting Started" section for details.

7. FAQ

In this section, you find answers to frequently asked questions that have not been answered in this document's other sections.

7.1. Do I need a protection circuit?

If the input voltage of the PMM240 risks jumping above the maximum allowed input voltage of 24 V, we recommend using a protection circuit. The easiest way is an LDO voltage regulator set to 24 V. This LDO could consist of a simple transistor combined with a Zener diode: Under normal circumstances, the LDO just passes the input voltage along. If the input voltage becomes higher than 24 V, the LDO limits it to 24 V.

7.2. How do I set the Charge Current Limit?

Similar to the Input Current Limit, the charge current can also be limited to a maximum value. That makes sense if the charging time is not crucial and you prefer to charge the battery more slowly, thus reducing thermal stress.

You can set the Charge Current Limit just like the Input Current Limit. The only differences are the command (0x3c instead of 0x3d) and the value range (256 mA to 6200 mA). For more details, refer to the PMM specification document.

7.3. Does the PMM240 wake batteries from shipping mode?

Yes, via the T-pin of the battery, the PMM240 senses the presence of a battery. It then provides a wake-up charge.

7.4. Does the PMM240 wake deep-discharged batteries?

Yes, see the answer to question 7.3.

Please note that sometimes the battery is so deeply-discharged that the protection fuse of the battery has triggered. In that case, even the PMM240 is not capable of waking the battery up.

7.5. How can I know whether the external power supply is connected?

There are three options:

The first option is to read the status of the GPIO2 pin. If GPIO2 is steady high (in contrast to blinking), the power supply is present. If it is steady low, no power supply is present. A blinking GPIO2 shows a charger error.

The second option is to listen on the SMBus. If the power supply is present, the PMM240 exchanges about one status information per second. This data exchange rate drops to one SMBus message every 10 to 60 seconds if no power supply is attached.

The third option is attempting to write a command to the PMM240 configuration register. The PMM240 responds to SMBus requests only if the PMM240 is powered by an external power supply (see also question 7.6).

7.6. Why does the PMM240 reply with NACK to all of my configuration attempts?

If no power supply is attached, the PMM240 saves power by disconnecting its SMBus communication. You can still communicate with the battery over SMBus, but the PMM240 will not reply to any configuration attempts.

8. Documentation Support

For related documentation, see the following:

- PMM240 specification
- RRC2xxx Design-In Guidelines
- https://www.rrc-ps.com/pmm240-calculator, containing links to the Application Note and the Datasheet.
- System Management Bus Specification (Rev 1.1, Dec 11, 1998)
- Smart Battery Data Specification (Rev 1.1, Dec 15, 1998)
- Smart Battery Charger Specification (Rev 1.1, December 1, 1998)

If there is no link, or you cannot follow it, please contact your local RRC salesperson.

9. Revision History

AppNote Revision	Valid from	Changes	Author	PMM240 Revision
A	11. Mar 2015	First release of the PMM240 Application Note.	Hugo Branco	-01
B	02. Mar 2016	Updated Application Note to new firmware revision	Hugo Branco	-03
C	17. Apr 2018	Structural and content update	Hugo Branco	-05
D	09. Sep 2019	Corporate Design update	Hugo Branco	-05
E	15. Mar 2020	PMM240 Application Note completely rewritten	Bernhard Krämer	-05
F	27. Jul 2021	Improvements on Table 3 and 4. Enhanced sec 3, correction in sec 8. Added a question to FAQ.	Bernhard Krämer	-05
G	14. Feb 2022	Referenced PMM240 online calculator. Added a section about verifying the Current Limit configuration. Moved parts of the SMBus communication to the specification document. Added a flow charts about the Power Modes.	Bernhard Krämer	-05

The AppNote refers to the PMM240 revision indicated above. The PMM240 revision is printed on the PMM240 packaging.

Germany/Headquarters
RRC power solutions GmbH
Technologiepark 1
D-66424 Homburg / Saar

Tel.: +49 6841 98090
Fax: +49 6841 9809280
Email: sales@rrc-ps.de
Web: www.rrc-ps.com

USA
RRC power solutions Inc.
18340 Yorba Linda Blvd.,
Suite 107-437
Yorba Linda, CA 92886

Tel.: +1 714 777 3604
Fax: +1 714 777 3658
Email: usa@rrc-ps.com
Web: www.rrc-ps.com

Hong Kong
RRC power solutions Ltd.
S-V,6/F, Valiant Industrial Centre
2-12 Au Pui Wan Street
Fo Tan, N.T., Hong Kong

Tel.: +852 2376 0106
Fax: +852 2375 0107
Email: hkrrc@rrc-ps.cn
Web: www.rrc-ps.cn

China
RRC power solutions Ltd.
Room 520, Yuanlin Building,
Aiguo Road No. 3066,
Luohu District,
Shenzhen 518021
Tel.: +86 755 8374 1908
Fax: +86 755 8374 1861
Email: hkrrc@rrc-ps.cn
Web: www.rrc-ps.cn