

Using the bq33100

Programming Data FLash

Contents

Overview	2
Hardware Interface	2
Software Interface	3
Standard Data Commands	3
Extended Data Commands.....	Error! Bookmark not defined.
Host Data and FAULT Interaction	Error! Bookmark not defined.
Appendix A – SMBus Electrical Characteristics.....	5
Appendix B - Main Differences Between SMBus and I2C	7
DC Specifications for SMBus and I2C	7
Timing specifications differences of I2C and SMBus	8
Other differences	9
ACK and NACK usage.....	9
SMBus protocols	9

Overview

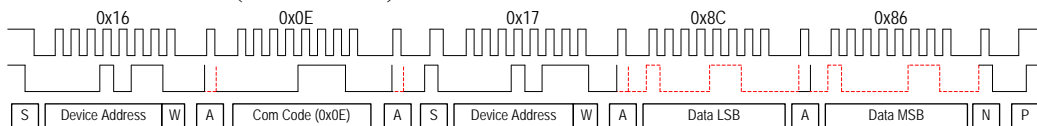
The bq33100 can be configured via programming of the integrated data flash which can be programmed via the SMBus interface. This document describes the features and procedure to update the data flash configuration.

Hardware Interface

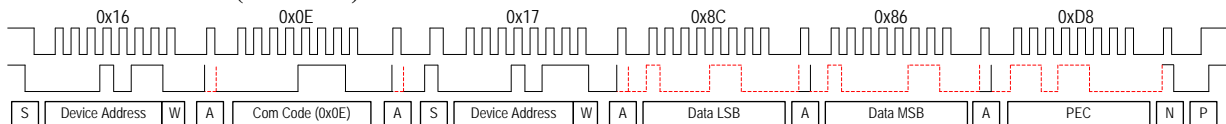
The data is accessed via the SMBus interface, SDA (pin14) and SCL (pin16) [See Appendix A – SMBus Electrical Characteristics for Electrical Characteristics of the SMBus Interface].

The bq33100 acts as a slave only with a fixed address of 0x16 and can be mastered by an SMBus 1.1 or 2.0 or I²C Master (See Appendix B - Main Differences Between SMBus and I2C) with the following data format:

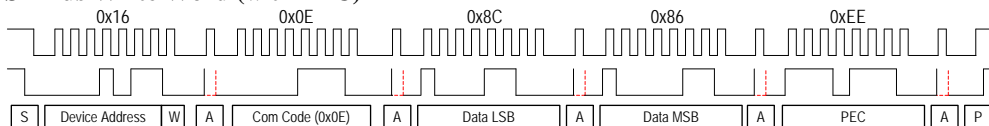
SMBus Read Word (without PEC)



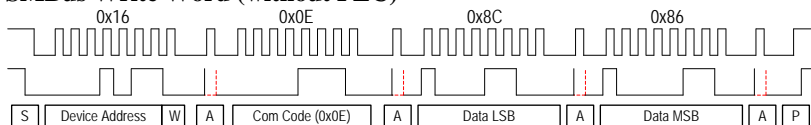
SMBus Read Word (with PEC)



SMBus Write Word (with PEC)



SMBus Write Word (without PEC)



Legend: - - - Slave control
 — Host Control

Figure 1: SMBus Protocol

Software Interface

The bq33100 data flash can be accessed through the following SMBus Word functions.

NOTE: The data flash interface commands are only accessible when the bq33100 is in the Unsealed mode (OperationStatus() [SS] = 0) and are NACK'ed if the bq33100 is in sealed mode, [SS] = 1.

Unsealing is a 2 step command performed by writing the 1st word of the UnSealKey to ManufacturerAccess followed by the second word of the UnSealKey to ManufacturerAccess . The unseal key can be read and changed via the extended SBS block command UnSealKey when in Unsealed Mode. To return to the Sealed mode, either a hardware reset is needed, or the ManufacturerAccess Seal command is needed.

Data Flash Access Commands

SBS Cmd	Mode	Name	Format	Size in Bytes	Offset Start	Offset End
0x77	R/W	DataFlashSubClassID	hex	2	na	na
0x78	R/W	DataFlashSubClassPage1	hex	32	0	31
0x79	R/W	DataFlashSubClassPage2	hex	32	32	63
0x7a	R/W	DataFlashSubClassPage3	hex	32	64	95
0x7b	R/W	DataFlashSubClassPage4	hex	32	96	127
0x7c	R/W	DataFlashSubClassPage5	hex	32	128	159
0x7d	R/W	DataFlashSubClassPage6	hex	32	160	191
0x7e	R/W	DataFlashSubClassPage7	hex	32	192	223
0x7f	R/W	DataFlashSubClassPage8	hex	32	224	255

DataFlashSubClassID()

This read- or write-word function set the Sub Class ID for the data flash space to be programmed using the *DataFlashSubClassPage1..8()* commands.

DataFlashSubClassPage1..8()

This read- or write-word functions provide the data access to the data flash block identified in the *DataFlashSubClassID()* command.

Data Flash Programming Overview

The bq33100 data flash is organized into subclasses where each data flash variable is assigned an offset within its numbered subclass. For example: the **Charge Voltage** threshold location is defined as:

- Class = Charge Control
- SubClass = Charge Cfg = 34
- Offset = 0

Each subclass can be addressed individually by using the **DataFlashSubClassID** command and the data within each subclass is accessed by using the **DataFlashSubClassPage1..8** commands.

Reading and Writing subclass data are block operations, which are each 32 Bytes long. Data can be written in shorter block sizes, however. The final block in one subclass can be shorter than 32 bytes so care must be taken not to write over the subclass boundary. **None of the values written are bounded by the bq33100 and the values are not rejected by the gas gauge.** Writing an incorrect value may result in hardware failure due to firmware program interpretation of the invalid data. The data written is persistent, so a Power On Reset does resolve the fault.

Reading a SubClass

Information required:

- SubClassID
- Number of bytes in the subclass
- Variable Offset

Procedure:

1. Write the SubClassID to bq33100 using *DataFlashSubClassID()* command.
2. Read a block of data using *DataFlashSubClassPage1..8()* command. A subclass can hold up to 256 bytes of data, but subclass data can only be read in 32 byte long data blocks. The *DataFlashSubClassPage1* command reads only the first 32 bytes in a subclass, the *DataFlashSubClassPage2* command reads the second 32 bytes in a subclass, and so on. For example: if the subclass has 40 bytes, *DataFlashSubClassPage1 + DataFlashSubClassPage2* is needed to read the whole subclass.

Writing a SubClass

Information required:

- SubClassID
- Number of bytes in the subclass
- 32 bytes of initialized data to be written. Less than 32 bytes is acceptable if a subclass contains less than 32 bytes in the last block.

Procedure:

1. Write the SubClassID to bq33100 using *DataFlashSubClassID()* command.
2. Write a block of data using *DataFlashSubClassPage1..8()* command. A subclass can hold up to 256 bytes of data, but subclass data can only be write in 32 byte long data blocks. The *DataFlashSubClassPage1()* command writes only the first 32 bytes in a subclass, the *DataFlashSubClassPage2()* command writes the second 32 bytes in a subclass, and so on. For example, if the subclass has 40 bytes and data in offset 34 of the subclass needs to be changed, use *DataFlashSubClassPage2()* to write data from byte 32 - 40 of the subclass.

Example

The following is an example of how to update a specific variable in the data flash configuration of the bq33100. This example is to write the value of **Min Voltage** to a value of 8.7 V the following sequence is used.

Read complete Monitoring-System Requirements subclass (SubclassID = 86) into RAM:

1. Write Subclass ID
 - a. SMB Slave Address (0x16)
 - b. SMB CMD 0x77 with 0x0056 as data (=86 decimal)

2. Read Subclass
 - a. SMB Slave Address (0x16)
 - b. SMB CMD 0x78 receiving 32 bytes of data

3. Overwrite offset 4 of received data with 8.7 V:
 - a. Update offset 4 with 0x21fc (=8700 decimal)

Write the complete subclass back to the bq33100:

4. Write Subclass ID
 - a. SMB Slave Address (0x16)
 - b. SMB CMD 0x77 with 0x0056 as data

5. Write Subclass
 - a. SMB Slave Address (0x17)
 - b. SMB CMD 0x78 with 32 bytes of data
 - c.

6. Read Subclass
 - a. SMB Slave Address (0x16)
 - b. SMB CMD 0x78 receiving 32 bytes of data

Verify read back was intended written data

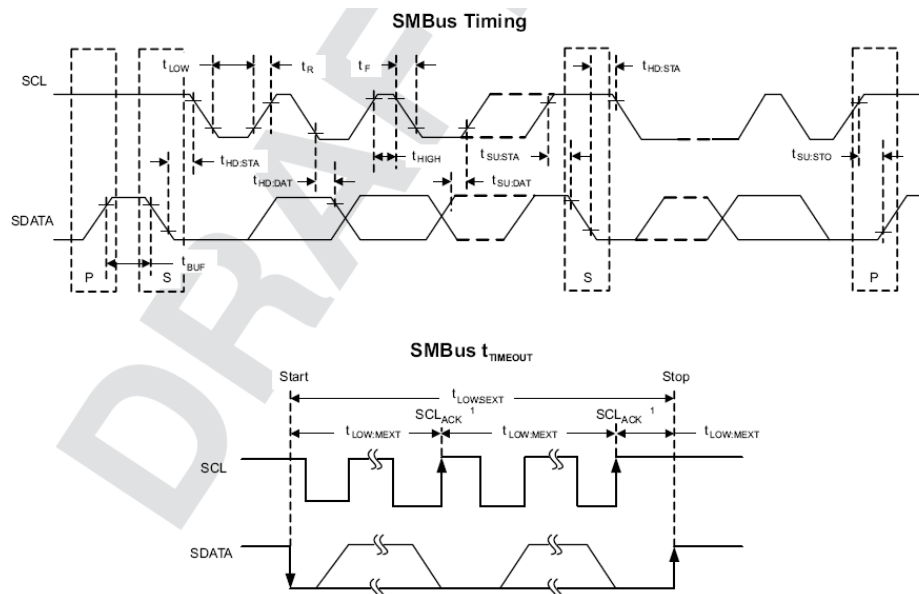
Appendix A – SMBus Electrical Characteristics

SMBus

Typical values stated where $T_A = 25^\circ\text{C}$ and $V_{CCPACK} = V_{CC} = 14.4\text{V}$, Min/Max values stated where $T_A = -40^\circ\text{C}$ to 85°C and $V_{CCPACK} = V_{CC} = 3.8\text{V}$ to 25V (unless otherwise noted)

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
f_{SMB}	SMBus operating frequency	Slave mode, SCL 50% duty cycle	10	100	kHz
f_{MAS}	SMBus master clock frequency	Master mode, no clock low slave extend	51.2		kHz
t_{BUF}	Bus free time between start and stop		4.7		μs
$t_{\text{HD:STA}}$	Hold time after (repeated) start		4.0		μs
$t_{\text{SU:STA}}$	Repeated start setup time		4.7		μs
$t_{\text{SU:STO}}$	Stop setup time		4.0		μs
$t_{\text{HD:DAT}}$	Data hold time	Receive mode	0		ns
		Transmit mode	300		
$t_{\text{SU:DAT}}$	Data setup time		250		ns
t_{TIMEOUT}	Error signal/detect	See (1)	25	35	ms
t_{LOW}	Clock low period		4.7		μs
t_{HIGH}	Clock high period	See (2)	4.0	50	μs
$t_{\text{LOW:SEXT}}$	Cumulative clock low slave extend time	See (3)		25	ms
$t_{\text{LOW:MEXT}}$	Cumulative clock low master extend time	See (4)		10	ms
t_{F}	Clock/data fall time	See (5)		300	ns
t_{R}	Clock/data rise time	See (6)		1000	ns

- (1) The bq33100 times out when any clock low exceeds t_{TIMEOUT}
- (2) $t_{\text{HIGH, Max}}$ is the minimum bus idle time. $\text{SCL} = \text{SDA} = 1$ for $t > 50 \mu\text{s}$ causes reset of any transaction involving bq33100 that is in progress. This specification is valid when the NC_SMB control bit remains in the default cleared state ($\text{CLK}[0]=0$). If NC_SMB is set then the timeout is disabled.
- (3) $t_{\text{LOW:SEXT}}$ is the cumulative time a slave device is allowed to extend the clock cycles in one message from initial start to the stop.
- (4) $t_{\text{LOW:MEXT}}$ is the cumulative time a master device is allowed to extend the clock cycles in one message from initial start to the stop.
- (5) Rise time $t_{\text{R}} = V_{\text{ILMAX}} - 0.15$ to $(V_{\text{IHMIN}} + 0.15)$
- (6) Fall time $t_{\text{F}} = 0.9V_{\text{DD}}$ to $(V_{\text{ILMAX}} - 0.15)$



(1) SCL_{ACK} is the acknowledge-related clock pulse generated by the master.

Appendix B¹ - Main Differences Between SMBus and I2C

The major differences between I2C and SMBus fall into several categories including electrical, timing, protocols and operating modes.

DC Specifications for SMBus and I2C

Both I2C and SMBus are capable of operating with mixed devices that either have fixed input levels (such as Smart Batteries) or their input levels are related to VDD. When mixing devices, the I2C specification defines the VDD to be 5.0 Volt +/- 10% and the fixed input levels to be 1.5 and 3.0 Volts.

Version 1.1 of SMBus instead of relating the bus input levels to VDD, it defines them to be fixed at 0.8 and 2.1 Volts. This SMBus specification allows for bus implementations with VDD ranging from 3 to 5 Volts +/- 10%. SMBus has relaxed in this version the initial requirement for fixed input levels of 0.6 and 1.4 Volts, in order to reduce the cost of SMBus compliant devices. Devices compliant with the 1.0 specification of SMBus will still operate with a version 1.1 SMBus. In addition they will be ready to operate with 2 Volt SMBus implementations in the future.

A second difference in the DC parameters between I2C and SMBus is in the power consumption of the bus.

SMBus was designed to accommodate extremely low power consumption devices, such as the control circuitry within a Smart Battery. These devices have limited current sinking capabilities and a low power consumption bus is essential for maintaining communications without draining the battery of a mobile computer. As a result, SMBus sets more stringent DC requirements than I2C. One of the main differences is the IOL specification for VOL = 0.4 Volts. SMBus devices are required to sink a minimum of 100 uA as opposed to 3mA specified for I2C devices for the same VOL.

A third difference is in the specification of the maximum leakage current for each device connected to the bus. I2C specifies the maximum leakage current to be 10 uA. SMBus version 1.0 specified maximum leakage current of 1 uA. Version 1.1 of the SMBus specification relaxes the leakage requirements to 5 uA, in order to reduce the cost of testing of SMBus devices.

Finally, SMBus does not specify a maximum bus capacitance. Instead it specifies the I_{PULLDOWN} maximum of 350 uA. Bus capacitance can be calculated taking into consideration the maximum rise time and I_{PULLDOWN}.

The following table lists the main differences among the DC parameters for I2C and SMBus.

Symbol	Parameter	Std I2C mode device		Fast I2C mode device		SMBus device		Units
		MIN	MAX	MIN	MAX	MIN	MAX	

¹ [Reference: System Management Bus Specification, Revision 1.1, December 11, 1998]

Symbol	Parameter	Std I2C mode device		Fast I2C mode device		SMBus device		Units
		MIN	MAX	MIN	MAX	MIN	MAX	
VIL	Fixed input level	-0.5	1.5	-0.5	1.5	-0.5	0.8	V
	VDD related input level	-0.5	0.3VDD	-0.5	0.3 VDD	N/A	N/A	V
VIH	Fixed input level	3.0	VDDmax+0.5	3.0	VDDmax+0.5	2.1	5.5	V
	VDD related input level	0.7VDD	VDDmax+0.5	0.7VDD	VDDmax+0.5	N/A	N/A	V
VHYS	VIH-VIL	N/A	N/A	0.05VDD	-	N/A	N/A	V
VOL	VOL @ 3mA	0	0.4	0	0.4	N/A	N/A	V
	VOL @ 6mA	N/A	N/A	0	0.6	N/A	N/A	
	VOL @ 350uA	N/A	N/A	N/A	N/A	-	0.4	
IPULLUP		N/A	N/A	N/A	N/A	100	350	uA
ILEAK		-10	10	-10	10	-5	5	uA

Table 1: DC Parameter Comparison Between Standard I2C and SMBus Devices

Timing specifications differences of I2C and SMBus

There are differences in the timing specifications between I2C and SMBus. As in the case of DC specification, proper understanding of the parameters is needed in order to combine reliably I2C with SMBus devices.

1. SMBus defines a minimum bus clock frequency F_{SMB} of 10 KHz. I2C does not specify any minimum bus frequency. Besides maintaining effective bus throughput, this SMBus specification parameter can be used as a simple way to detect a bus idle condition (in addition or in lieu of detecting each STOP condition) as well as to implement bit timeout.
2. Maximum clock frequency for SMBus is defined at 100 KHz. I2C provides two modes of operation. The STANDARD MODE up to 100 KHz and the FAST-MODE up to 400 KHz.
3. SMBus defines a clock low time-out, $T_{TIMEOUT}$ of 35 ms. I2C does not specify any timeout limit.
4. SMBus specifies $T_{LOW:SEXT}$ as the cumulative clock low extend time for a slave device. I2C does not have a similar specification.
5. SMBus specifies $T_{LOW:MEXT}$ as the cumulative clock low extend time for a master device. Again I2C does not have a similar specification.
6. SMBus defines both rise and fall time of bus signals. I2C does not.

The SMBus time-out specifications do not preclude I2C devices co-operating reliably on the SMBus. It is the responsibility of the designer to ensure that I2C devices are not going to violate these bus timing parameters.

Other differences

ACK and NACK usage

There are the following differences in the use of the NACK bus signaling:

- In I2C, a slave receiver is allowed not to acknowledge the slave address, if for example is unable to receive because it's performing some real time task. SMBus requires devices to acknowledge their own address always, as a mechanism to detect a removable device's presence on the bus (battery, docking station, etc.)
- I2C specifies that a slave device, although it may acknowledge its own address, some time later in the transfer it may decide that it cannot receive any more data bytes. The I2C specifies, that the device may indicate this by generating the not acknowledge on the first byte to follow.

Besides to indicate a slave device busy condition, SMBus is using the NACK mechanism also to indicate the reception of an invalid command or data. Since such a condition may occur on the last byte of the transfer, it is required that SMBus devices have the ability to generate the not acknowledge after the transfer of each byte and before the completion of the transaction. This is important because SMBus does not provide any other resend signaling. This difference in the use of the NACK signaling has implications on the specific implementation of the SMBus port, especially in devices that handle critical system data such as the SMBus host and the SBS components.

SMBus protocols

Each message transaction on SMBus follows the format of one of the defined SMBus protocols. The SMBus protocols are a subset of the data transfer formats defined in the I2C specifications. I2C devices that can be accessed through one of the SMBus protocols are compatible with the SMBus specifications.

I2C devices that do not adhere to these protocols cannot be accessed by standard methods as defined in the SMBus and ACPI specifications.