LTC2978 Firmware Programmers Getting Started Guide – Margining Output Voltage

Revision 0.4

Author: Mike Holloway, Linear Technology

Date: 2/25/2010, 10:38 AM

Revision History:

0.4	Corrected command code for PAGE from erroneous 0x01 to correct value 0x00
0.3	Terminology Change: changed ‘command byte’ to ‘command code’
0.2	Document Formatting
0.1	Initial Draft

Purpose

The LTC2978 provides a great deal of power and configurability for custom applications. The part additionally provides onboard non-volatile memory (NVM) or EEPROM to store and recall configuration parameters.

This architecture allows the chip to power up and load the desired customer configuration **autonomously with no I2C/firmware interaction required**.

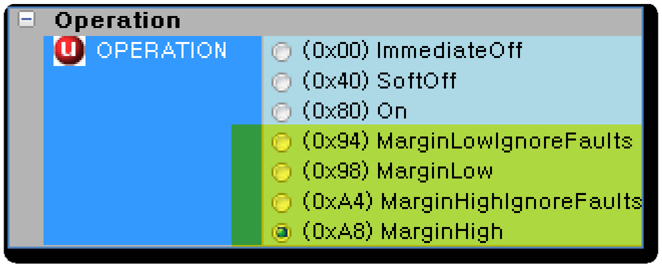
However, there are some circumstances where a customer may wish to author firmware to communicate with the part. This document provides a quick overview of one common firmware task – Margining the output voltage.

LTpowerPlay™ is a powerful tool that can be used to understand how to author the firmware to talk to the LTC2978. You can download LTpowerPlay™ from http://ltpowerplay.com.

Margining the Output Voltage

Margining can be accomplished by programming 3 paged registers. VOUT_MARGIN_HIGH, VOUT_MARGIN_LOW, and OPERATION. The first two registers set up the margin levels – that is, VOUT_MARGIN_HIGH programs the voltage to which the output will margin when the device is margining high.

In order to actually margin the output voltage, you need to send the OPERATION register to one of the Margin* values highlighted in yellow below.



Operation

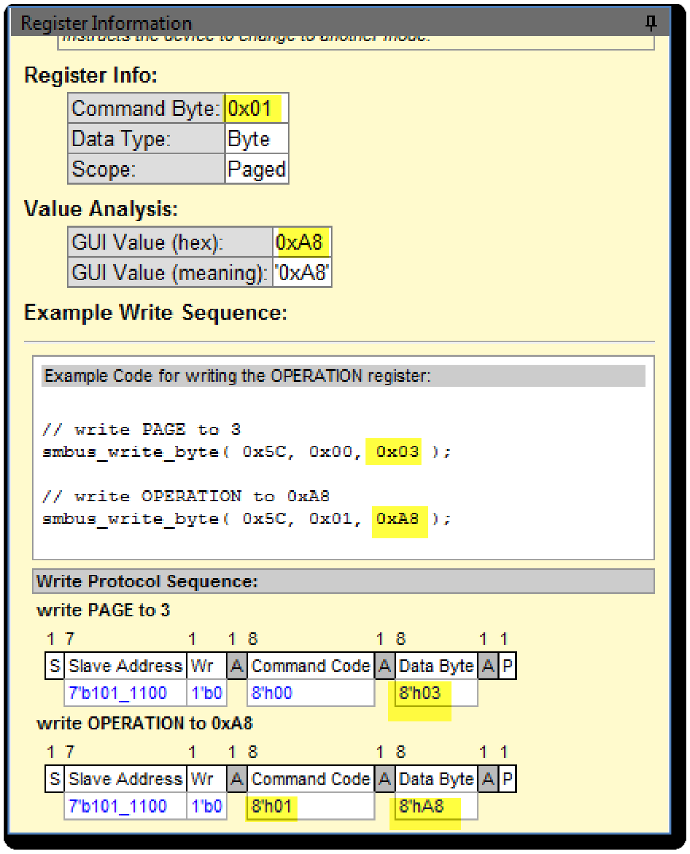
OPERATION

○ (0x00) ImmediateOff
○ (0x40) SoftOff
○ (0x80) On
○ (0x94) MarginLowIgnoreFaults
○ (0x98) MarginLow
○ (0xA4) MarginHighIgnoreFaults
● (0xA8) MarginHigh

Note that the LTpowerPlay™ GUI shows you the hex values corresponding to each particular human readable value. For example, MarginHigh is equivalent to the hex value 0xA8.

Also note the 'Register Information' tab in the GUI. This tab shows you more detailed information about the selected register which can be a great help in understanding how to author the firmware.

Below is an example of the Register information for the OPERATION register when page 3 is selected:



instructs the device to change to another mode.

Register Info:

Command Byte:	0x01
Data Type:	Byte
Scope:	Paged

Value Analysis:

GUI Value (hex):	0xA8
GUI Value (meaning):	'0xA8'

Example Write Sequence:

Example Code for writing the OPERATION register:

```
// write PAGE to 3
smbus_write_byte( 0x5C, 0x00, 0x03 );

// write OPERATION to 0xA8
smbus_write_byte( 0x5C, 0x01, 0xA8 );
```

Write Protocol Sequence:

write PAGE to 3

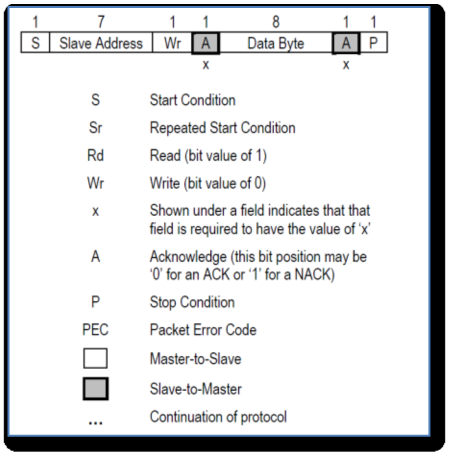
1	7	1	1	8	1	8	1	1
S	Slave Address	Wr	A	Command Code	A	Data Byte	A	P
	7'b101_1100	1'b0		8'h00		8'h03		

write OPERATION to 0xA8

1	7	1	1	8	1	8	1	1
S	Slave Address	Wr	A	Command Code	A	Data Byte	A	P
	7'b101_1100	1'b0		8'h01		8'hA8		

Note that the information in this tab is 'context sensitive'. In other words, it shows you detailed information based on what is selected in the GUI. In this particular case it shows the steps required to write the OPERATION register to the value 0xA8 (Margin High) for page 3 for the chip at 7-bit I2C address 0x5C.

Because the OPERATION is a paged register, and page 3 is selected in the GUI, the examples include instructions on writing the PAGE Register to the value 3. The view shows example C code as well as detailed SMBus protocol diagrams. The protocol diagrams follow the conventions established in the SMBus specification. The protocol diagram legend, or element key is shown in the figure below:



S	Start Condition
Sr	Repeated Start Condition
Rd	Read (bit value of 1)
Wr	Write (bit value of 0)
x	Shown under a field indicates that that field is required to have the value of 'x'
A	Acknowledge (this bit position may be '0' for an ACK or '1' for a NACK)
P	Stop Condition
PEC	Packet Error Code
□	Master-to-Slave
▨	Slave-to-Master
...	Continuation of protocol

Figure: SMBus packet protocol diagram element key

So in our specific case the steps required to command channel 3 of device at 7-bit I2C address 0x5C to margin it's voltage to its pre-programmed VOUT_MARGIN_HIGH level are as follows:

Write PAGE to 3

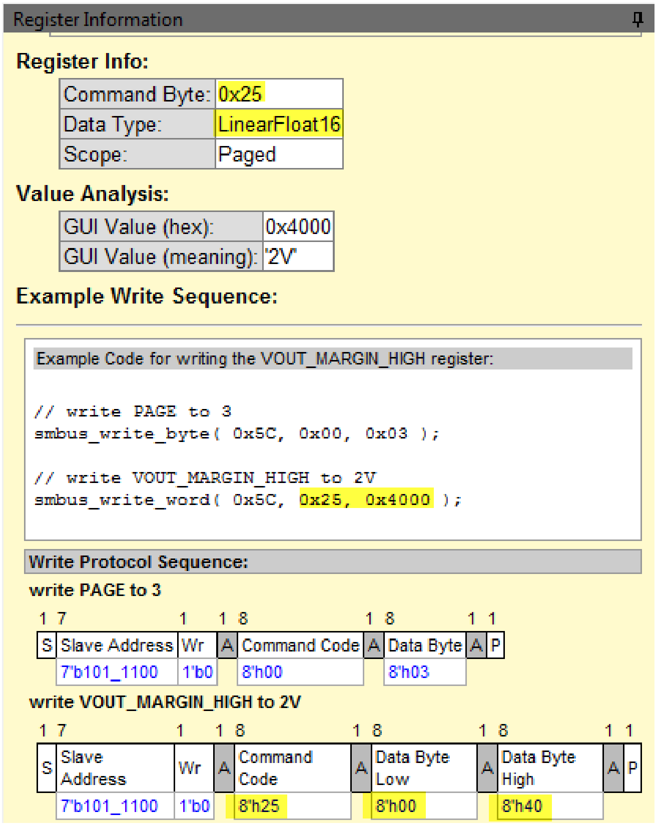
- <Start Condition>
- Transmit Byte 0xB8 (0x5C<<1 with Write bit set to 0)
- Transmit Byte 0x00 (command code for PAGE register)
- Transmit Byte 0x03 (to select PAGE=3)
- <Stop Condition>

Write OPERATION to 'MarginHigh' (0xA8)

- <Start Condition>
- Transmit Byte 0xB8 (0x5C<<1 with Write bit set to 0)
- Transmit Byte 0x01 (command code for OPERATION register)
- Transmit Byte 0xA8 (to select OPERATION='MarginHigh')
- <Stop Condition>

When the device receives the above sequence, it will margin the channel 3 output to it's pre-programmed value for VOUT_MARGIN_HIGH.

Now let's say we want to margin the output voltage to 2.0V. We need to program the VOUT_MARGIN_HIGH level to be 2.0V. Select page 3 in the GUI, and select the paged VOUT_MARGIN_HIGH register. Type '2.0' and press Enter. Now view the register information tab:



Register Information

Register Info:

Command Byte:	0x25
Data Type:	LinearFloat16
Scope:	Paged

Value Analysis:

GUI Value (hex):	0x4000
GUI Value (meaning):	'2V'

Example Write Sequence:

Example Code for writing the VOUT_MARGIN_HIGH register:

```
// write PAGE to 3
smbus_write_byte( 0x5C, 0x00, 0x03 );

// write VOUT_MARGIN_HIGH to 2V
smbus_write_word( 0x5C, 0x25, 0x4000 );
```

Write Protocol Sequence:

write PAGE to 3

1	7	1	1	8	1	8	1	1
S	Slave Address	Wr	A	Command Code	A	Data Byte	A	P
	7'b101_1100	1'b0		8'h00		8'h03		

write VOUT_MARGIN_HIGH to 2V

1	7	1	1	8	1	8	1	8	1	1
S	Slave Address	Wr	A	Command Code	A	Data Byte Low	A	Data Byte High	A	P
	7'b101_1100	1'b0		8'h25		8'h00		8'h40		

Note that because VOUT_MARGIN_HIGH is a paged register, this view also tells us that we have to write the PAGE register to select PAGE=3. It also tells us that the command code for VOUT_MARGIN_HIGH is 0x25, and the data type/format is 'LinearFloat16'. This is a simple PMBus numeric format that represents an unsigned 16-bit mantissa which when multiplied by 2^(-13) determines a voltage. The value 2V is thus represented by the mantissa 2*8192, or 16384, or 0x4000 hex.

Note in the protocol diagram that two bytes are sent to transfer the 16-bit 'word' to the slave. In SMBus the lower byte is always sent first. So in order to write the VOUT_MARGIN_HIGH register to 2V (assuming PAGE has already been written), the following would occur on the I2C bus:

Write VOUT_MARGIN_HIGH to '2.0V' (0x4000)

- <Start Condition>
- Transmit Byte 0xB8 (0x5C<<1 with Write bit set to 0)
- Transmit Byte 0x25 (command code for VOUT_MARGIN_HIGH register)
- Transmit Byte 0x00 (lower byte of 0x4000)
- Transmit Byte 0x40 (upper byte of 0x4000)
- <Stop Condition>

The full set of commands required to margin the output voltage of channel 3 of the chip at 7-bit I2C address 0x5C voltage to 2.0V is as follows:

```
// write PAGE to 3
smbus_write_byte( 0x5C, 0x00, 0x03 );

// write VOUT_MARGIN_HIGH to 2V
smbus_write_word( 0x5C, 0x25, 0x4000 );

// write OPERATION to 0xA8
smbus_write_byte( 0x5C, 0x01, 0xA8 );
```

See the protocol diagrams above to see how this translates into primitive I2C elements on the wire.