How to change a GreenPAK comparator's threshold voltage using I2C

Author: David Riedell

Date: December 07, 2015

Introduction

This app note explains how to reconfigure the SLG46531's registers via I2C. Specifically, it shows how to change the thresholds of the GreenPAK's analog comparators to adjust the measurement window.

It is important to note that after reconfiguring the register bits, the GreenPAK will only remain in that configuration while it remains powered on; if the GreenPAK is reset, it will revert back to its initially programmed settings.

Hardware Setup

For this exercise, a Sparkfun Bus Pirate [1] was used to communicate with the GreenPAK via I2C. However, any I2C compatible microcontroller should work. Within the SLG46531, Pin 8 is the I2C macro-cell's clock signal (**SCL**) and Pin 9 is the I2C macro-cell's data signal (**SDA**). These pins, in addition to ground, must be connected to the Bus Pirate via the GreenPAK Universal Dev Board Expansion Connector pins, as shown in Figure 1. The Bus Pirate's VPU pin must also be connected to its +3V3 pin to connect its on-board pull-up resistors to +3.3V.

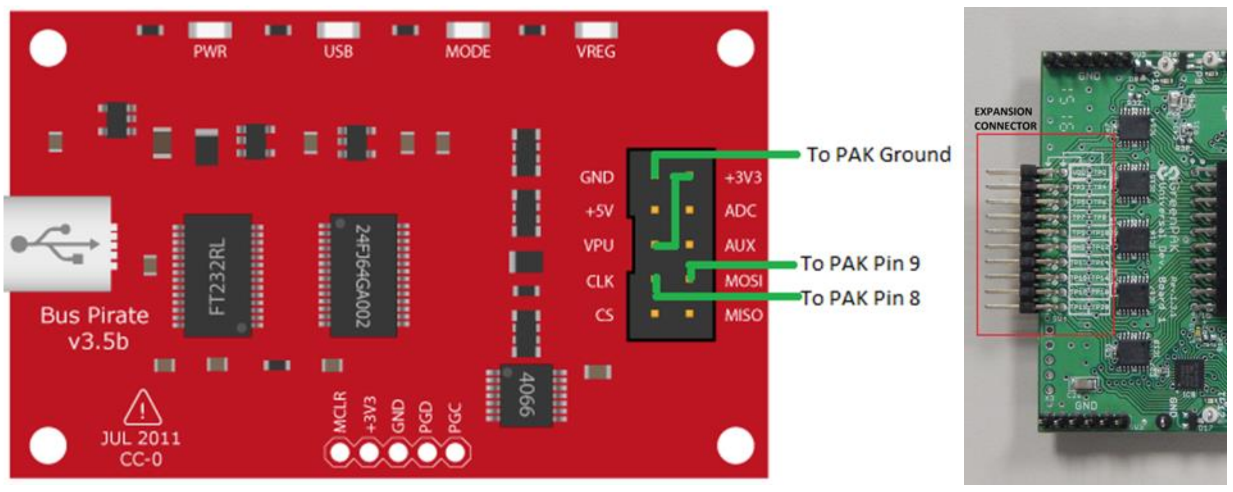


Figure 1. Bus Pirate to GreenPAK Connections

GreenPAK Configuration

The GreenPAK design shown in Figure 2(a) implements a simple analog window comparator. On the GreenPAK Universal Dev Board Expansion Connector, connect both Pin 6 and Pin 10 to a low input voltage (Vin) between 0-1.2V relative to the Universal Dev Board's ground pin.

In default configuration, Pin 3 will output high when Vin is greater than the ACMP0 reference voltage and below the ACMP1 reference voltage. In this example, if Vin is between 50mV and 500mV, Pin 3 will be high and its LED will be lit. If Vin is less than 50mV or greater than 500mV, Pin 3 will be low and its LED will be off.

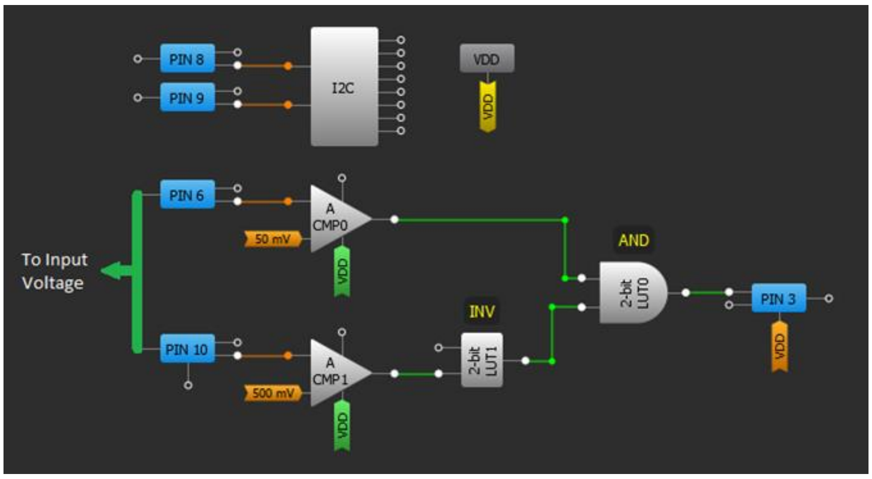


Figure 2(a). GreenPAK Block Diagram with Default Comparator References of 50mV and 500mV

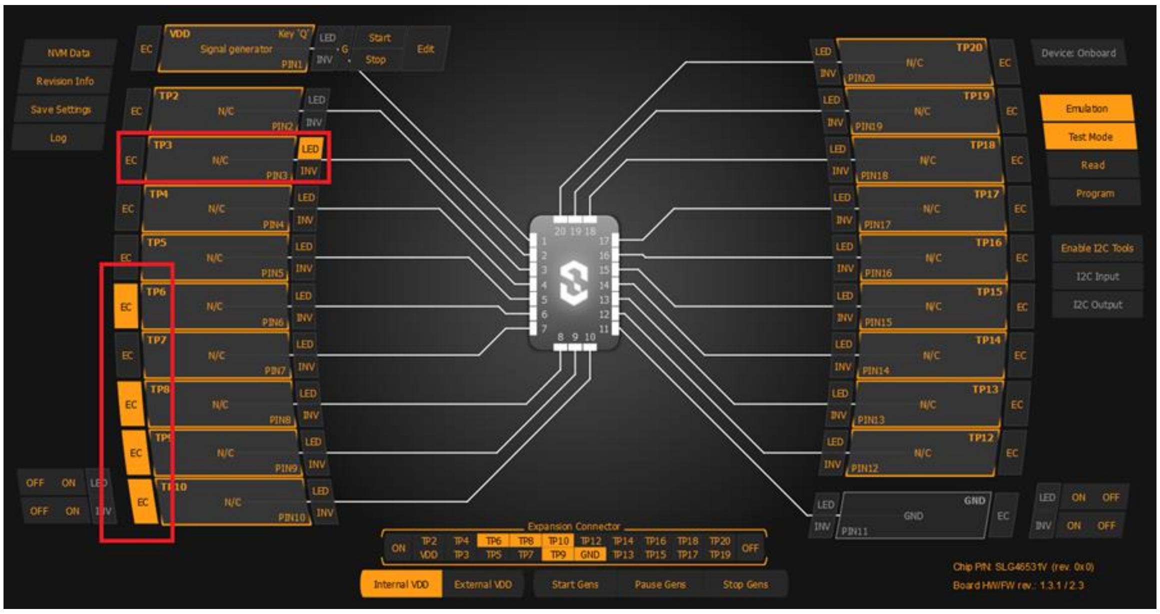


Figure 2(b). GreenPAK Emulator. Enable EC on Pins 6, 8, 9, & 10. Enable LED on Pin 3

Figure 2(a) includes a screen cap of the GreenPAK's block diagram, as well as where to connect the external V_{in} signal. When emulating, be sure to enable the Expansion Connector (EC) on Pins 6, 8, 9, & 10. Also, enable LED on Pin 3 as shown in Figure 2(b).

Creating an I2C Command to Read and Write Register Bits

We are now ready to start creating I2C commands to read and write the GreenPAK's register bits. Let's say that we want to increase the lower threshold from 50mV to **300mV**, so that our window comparator now outputs high only when V_{in} is between **300mV and 500mV**.

Read Data at Byte Address

Before writing any data to our byte address, we first want to find out the value it currently holds. The I2C write command writes a whole byte at a time, which means that if we aren't reconfiguring every single bit in the byte, we want to make sure that the unchanging bits are not overwritten.

Figure 3 shows the formula for writing a byte to the register, and is found on the SLG46531's Data Sheet in the I2C Communications section. The SLG46531 follows standard 400kHz I2C protocol.

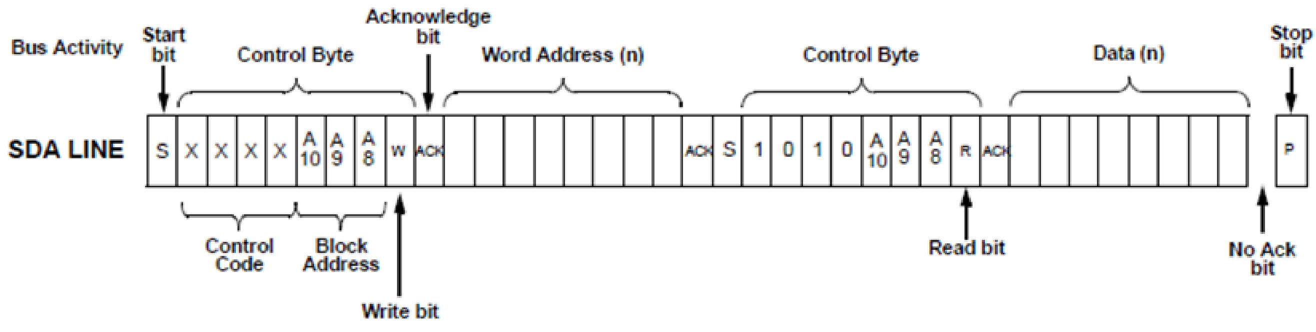


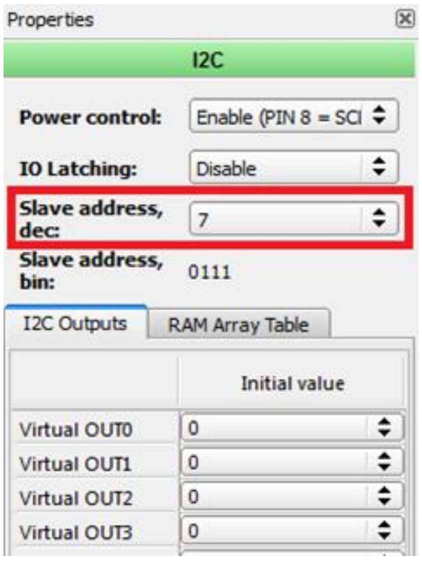
Figure 3. I2C Read Command Structure

The I2C Read command begins with a start bit, followed by a control byte (write), word address byte, second start bit, and a second control byte (read). The control byte contains the following items:

- Control code: This is set within GreenPAK Designer by clicking on the I2C bus and selecting a Slave address as shown in Figure 5. Since the control code is 4 bits long, you can have up to 16 slaved GreenPAK's with distinct addresses. In this example, we will use the slave address **7$_d$**, or **0111.**

- Block address: This selects the block within the register data that you wish to access. The SLG46531 has only one block with 256 bytes. The block's address is **000**.

- Write bit: This bit controls whether your master will read from the GreenPAK's register data (**1**) or write to the GreenPAK's register data (**0**).



Power control: Enable (PIN 8 = SCl
IO Latching: Disable
Slave address, dec: 7
Slave address, bin: 0111

	Initial value
Virtual OUT0	0
Virtual OUT1	0
Virtual OUT2	0
Virtual OUT3	0

Figure 4. Slave Address

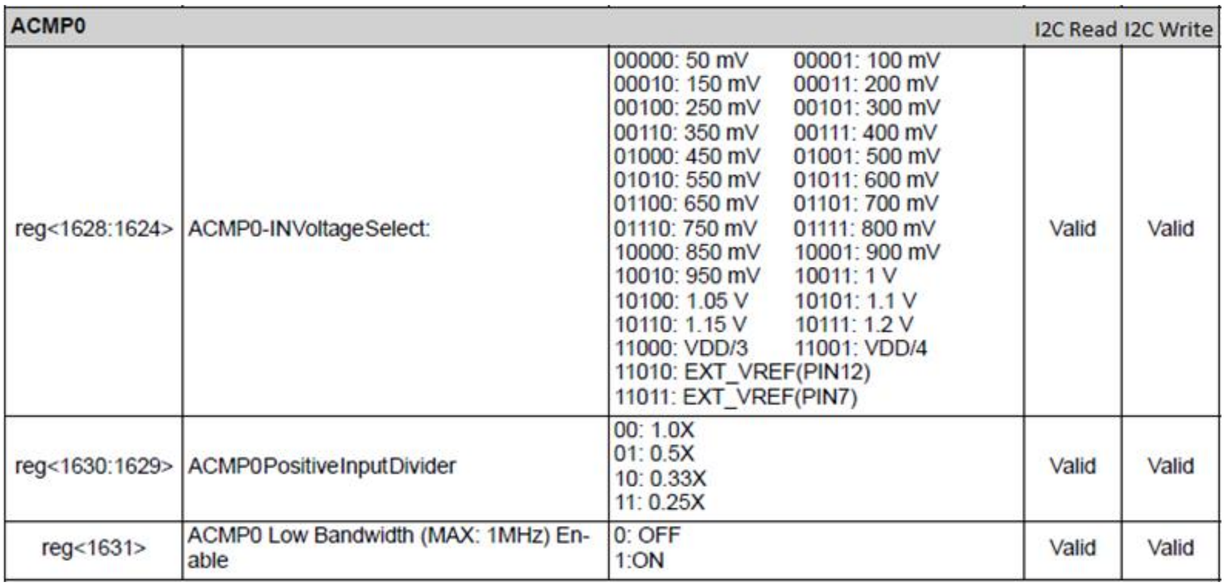
Find Byte Address

The SLG46531 has 2048 addressable register bits, as seen in the register definition appendix of its datasheet. In order to change the GreenPAK's operation after it has been programmed, we need to edit specific register bits, which will change its functionality.

Each of the SLG46531's four analog comparators has a 1-byte long section of the register to control their internal settings. The first 5 bits control the ACMP's reference voltage, the next 2 bits determine whether there is an input divider, and the last bit determines whether or not a low pass filter is connected for low bandwidth uses. Figure 5 is taken from the SLG46531 datasheet's register definition index. It shows the addresses for the ACMP0 register control bits and whether they can be read or written via I2C.

As you can see, the bits we are interested in are **reg<1631:1624>**. Before we can read those eight bits, we need to find their byte's address in hexadecimal notation. Use the following process:

1. Find the lowest bit address in the byte. In this case it is address **1624**.
2. Divide the bit address by 8 to find its byte address. In this case, 1624/8 = **203**$_d$.
3. Convert the byte address to Hexadecimal. In this case, 203$_d$ = **0xCB**.
4. Now we know that the word address of the byte we are interested in **0xCB**, which includes the 8 bits from reg<1631:1624>.



ACMP0			I2C Read	I2C Write
reg<1628:1624>	ACMP0-INVoltageSelect:	00000: 50 mV 00001: 100 mV 00010: 150 mV 00011: 200 mV 00100: 250 mV 00101: 300 mV 00110: 350 mV 00111: 400 mV 01000: 450 mV 01001: 500 mV 01010: 550 mV 01011: 600 mV 01100: 650 mV 01101: 700 mV 01110: 750 mV 01111: 800 mV 10000: 850 mV 10001: 900 mV 10010: 950 mV 10011: 1 V 10100: 1.05 V 10101: 1.1 V 10110: 1.15 V 10111: 1.2 V 11000: VDD/3 11001: VDD/4 11010: EXT_VREF(PIN12) 11011: EXT_VREF(PIN7)	Valid	Valid
reg<1630:1629>	ACMP0PositiveInputDivider	00: 1.0X 01: 0.5X 10: 0.33X 11: 0.25X	Valid	Valid
reg<1631>	ACMP0 Low Bandwidth (MAX: 1MHz) Enable	0: OFF 1:ON	Valid	Valid

Figure 5. ACMP0 Register Control Bits from SLG46531 Datasheet

Now that we know that process, we can calculate the rest of the ACMP register byte locations:

4.1.2 Construct I2C Read Command

We used a program called Terminal v1.91b [2] to send I2C commands to our GreenPAK. (Mac users can use ZTerm [3].)

Comparator	Register Bit Addresses	Byte Address (Decimal)	Byte Address (Hex)
ACMP0	reg<1631:1624>	203	0xCB
ACMP1	reg<1639:1632>	204	0xCC
ACMP2	reg<1647:1640>	205	0xCD
ACMP3	reg<1655:1648>	206	0xCE

Table 1. ACMP register byte locations

Once you have the hardware connected properly as outlined in Section 2.0, update your Terminal settings to those shown in Figure 6, then click "Connect" in the top left corner. The COM Port may be different than the one shown in Figure 6.

Once connected, you need to select your protocol and speed settings through text commands. In the lowest grey box, type the below commands, each followed by the **Enter** key:

m – This will pull up the menu system.

4 – This will select I2C from the available options.

4 – This will set the speed to ~400kHz.

You should now see a prompt in the center dialog box that says "I2C>". Next, we need to enable the power supplies and pull-up resistors.

In the same box, type:

WP – (They must both be capitalized.)

A message should appear in the middle dialog box that says:

WP

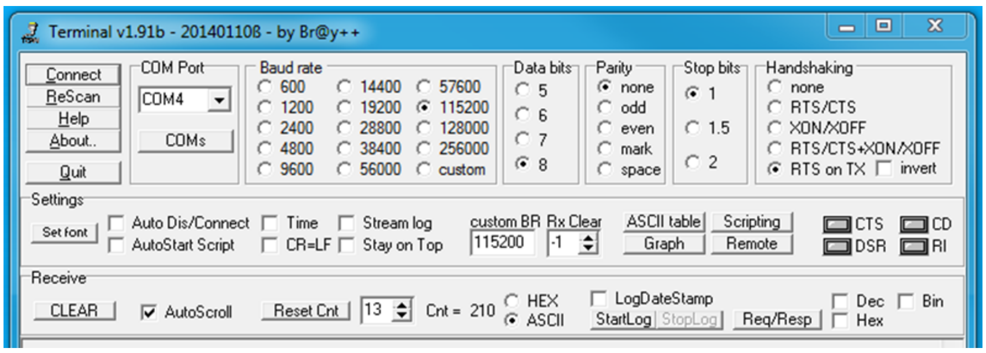
- Power supplies ON
- Pull-up resistors ON

You can now send your I2C Read command. In Terminal, a start bit is designated by a left bracket "**[**" and a stop bit is designated by a right bracket "**]**".

Additionally, in Terminal a Read command must end with "**r**". Therefore, the command we want to send to read the bits located at reg<1631:1624> is:

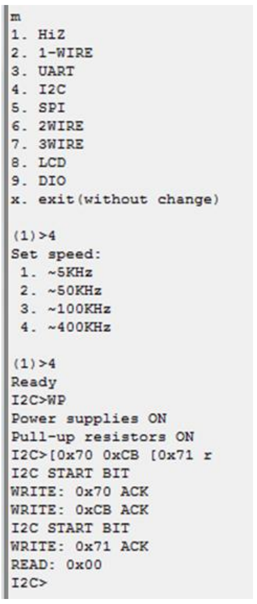
[0x70 0xCB [0x71 r

Figure 7 shows the output: The last line says "READ: 0x00", which means that initially all 8 of the bits from reg<1631:1624> are low, which is to be expected.


Terminal v1.91b - 20140110ß - by Br@y++

COM Port: COM4
Baud rate: 115200
Data bits: 8
Parity: none
Stop bits: 1
Handshaking: RTS on TX
custom BR: 115200
Rx Clear: -1
AutoScroll: ✓
Receive: ASCII
Cnt = 210

Figure 6. Terminal Settings


```
m
1. HiZ
2. 1-WIRE
3. UART
4. I2C
5. SPI
6. 2WIRE
7. 3WIRE
8. LCD
9. DIO
x. exit(without change)

(1)>4
Set speed:
 1. ~5KHz
 2. ~50KHz
 3. ~100KHz
 4. ~400KHz

(1)>4
Ready
I2C>WP
Power supplies ON
Pull-up resistors ON
I2C>[0x70 0xCB [0x71 r
I2C START BIT
WRITE: 0x70 ACK
WRITE: 0xCB ACK
I2C START BIT
WRITE: 0x71 ACK
READ: 0x00
I2C>
```

Figure 7. Terminal Output

Mask Data

In this case, we don't actually need to mask the data we obtained by using our Read command because every bit within byte 0xCB was set low. However, if any bits that we do not intend to change happen to be set high, we would need to use a mask. Since we will only be changing the bits in reg<1628:1624>, we want the bits in reg<1631:1629> to stay the same. We can achieve this by a simple bitwise AND:

This masked data will now be bitwise OR'd with the bits that we desire to write to reg<1628:1624>.

In our case, after masking our current data we have 0000 0000. After referencing Figure 5, we find that to create an ACMP0-INVoltageSelect of **300mV**, we need reg<1628:1624> to be **00101**. Performing a bitwise OR on these two bytes produces:

	Example where a mask **is not** necessary								Example where a mask **is** necessary							
Data	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	0
Mask	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0
AND Result	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 2. Mask Examples

0000 0000	|	**0 0101**	=	**0000 0101**	=	**0x05**
masked data	OR	desired data	=	result		

Table 3. Bitwise operation OR

4.3 Write Data to Byte Address

We are now ready to construct our I2C Write command. Figure 8 shows the formula for writing a byte to the register, and is found on the SLG46531's Data Sheet in the I2C Communications section.

Once you type that command into the Terminal dialog box and hit **Enter**, the register bits will be rewritten to reflect the new value. Once again, it is important to remember that they will only operate in this configuration while the GreenPAK is powered on; if the GreenPAK is powered down and restarted, it will revert back to its initially programmed settings.

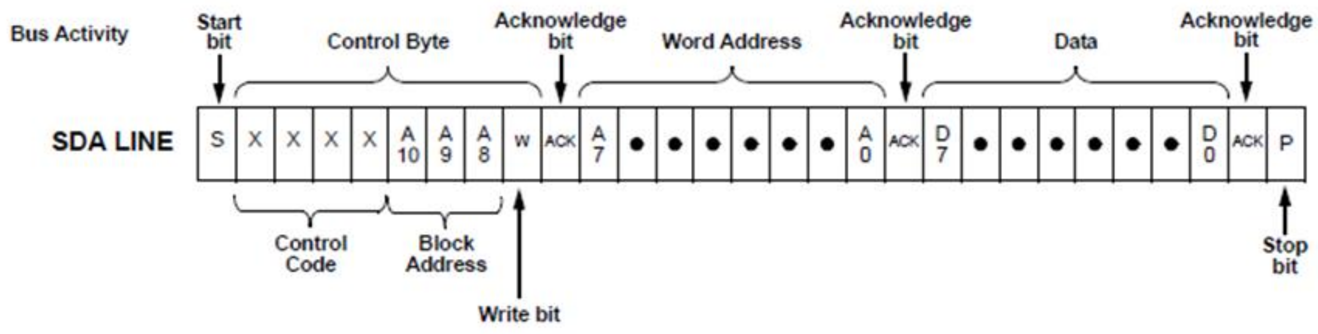


Figure 8. I2C formula for writing a byte to register

The Write command is simpler than the Read command. It only contains a start bit, control byte, word address, data byte, and stop bit. Looking at the components of this command individually, we have these values:

- Start bit = **[**
- Control Code = **0111**
- Block Address = **000**
- Write Bit = **0**
- Word Address = 203_d = **0xCB**
- Data = **0x05**
- Stop bit = **]**

After compiling these pieces together, our Write command is:

[0x70 0xCB 0x05]

To test the new comparator settings, connect a power supply to the GreenPAK Universal Development Board's Expansion Connector at Pin 6 or Pin 10 (which should be connected together). Then adjust the power supply voltage from 0V up to 1.2V, while watching the LED on Pin 3. The Pin 3 LED should now be lit only when V_{in} is between 300mV and 500mV. It is simple enough to also change the upper level of the window comparator. In section 4.1.1 we found that the byte address for ACMP1's register controls is 0xCC. If we wish to make the threshold of ACMP1 become **1V** (where reg<1628:1624> = 10011), our Write command would be this:

[0x70 0xCC 0x13]

Conclusion

The techniques described in this application note can be used to alter many of the 2048 register bits in the SLG46531. This makes the GreenPAK a much more versatile IC when used in combination with an I2C capable microcontroller. When using I2C to alter register bits, it is important to remember that the GreenPAK will only operate in the new configuration while the GreenPAK is powered on; if the GreenPAK is powered down and restarted, it will revert back to its original register settings.

About the Author

Name:	David Riedell
Background:	David attained a BS in Computer Engineering from North Carolina State University. He is currently working with CMICs as an applications engineer at Silego Technology.
Contact:	**appnotes@silego.com**

Document History

Document Title: How to change a GreenPAK comparator's threshold voltage using I2C

Document Number: AN-1091

Revision	Orig. of Change	Submission Date	Description of Change
A	David Riedell	08/12/2015	New application note.
B	David Riedell	12/07/2015	Added reference to ZTerm for Mac