CE66058 - Dynamically Switch CSD Bleed Resistor

Objective

You can achieve the proper sensitivity for sensors in an end system that have a considerable difference in parasitic capacitance (C_p) by tuning the raw counts of each sensor to 60 to 70 percent of the maximum value. This code example describes a method that compensates for differences in sensor raw counts due to variations in C_p by scanning the sensors with different bleed resistors.

Requirements

Project Name: Example_Dynamic_Bleed_Resistor

Programming Language: C

Associated Part Families: CY8C24x94

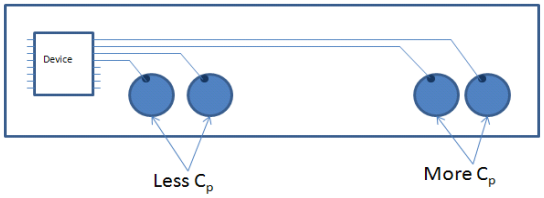
Software Version: PSoC Designer™ 5.4 Build 2946

Related Hardware: CY3280-24x94 UCC, CY3280-SLM, CY3240-I2USB Bridge/MiniProg1/MiniProg3

Overview

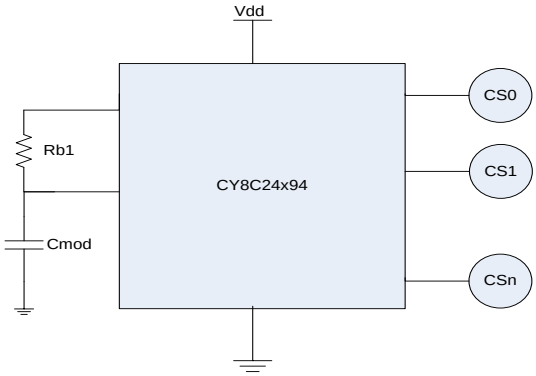
In some CapSense® designs, the dimensions of the sensing elements vary widely, which in turn causes variations in the C_p of the different sensors. For example, the C_p may vary because of a difference in size or the length of the trace from the button to the CapSense pin of the device. See Figure 1.

Figure 1. CapSense Design with Different Sensor C_p



To implement the CapSense Sigma Delta (CSD) functionality in a CY8C24x94 device, connect an external modulation capacitor (Cmod) and a bleed resistor (Rb) to the device, as shown in Figure 2. For more information about the CSD architecture in the CY8C24x94 CapSense controller, refer to the CY8C21x34/B CapSense Design Guide. To learn more about tuning the CSD User Module in the CY8C24x94 CapSense controller, refer to Appendix: CSD Calibration.

Figure 2. CapSense Design Using the CY8C24x94 Device



The following equation provides the relationship between the C_p and the raw counts in the CY8C24x94 device:

$$\text{Raw counts} = 2^N * Rb * \left[\frac{Sysclk}{4*Period}\right] * \left[\frac{1}{\frac{1}{4}+\frac{REF}{16}} - 1\right] * Cp \qquad \text{Equation 1}$$

In this equation:

N = Resolution
Rb = Bleed resistor (Ω)
Sysclk = Internal main oscillator (IMO) frequency (Hz)
Period = Prescaler period (Period = 1 if prescaler is not used)
REF = Voltage reference set in CSD User Module parameters
C_p = Sensor parasitic capacitance (F)

From Equation 1, if N, Rb, Sysclk, and REF have the same value for all the sensors, then the raw count is directly proportional to the sensor's C_p. If the sensor's C_p value is different, then the raw count value will also differ. As the raw counts must be 60 to 70 percent of the maximum value to achieve the best sensitivity, tuning one sensor for 60 to 70 percent can yield hugely different raw counts for the other sensor because of the difference in their C_p values.

If this difference is not great, you can use different V_{ref} values while scanning different sensors (choosing a high value for V_{ref} will result in lower sensitivity). However, if the difference in raw counts is very great, then using different V_{ref} values cannot help you to tune the sensors for the same raw counts (60 to 70 percent of the maximum value). For instance, this may happen in a project where a matrix and buttons are implemented at the same time, and the sensing elements of the matrix show a higher capacitance compared to a single button's capacitance.

As a result, if a group of sensing elements cannot be properly tuned, even with different values of V_{ref}, you must use different bleed resistors for different sensors.

To find the value of a bleed resistor, use the following equations:

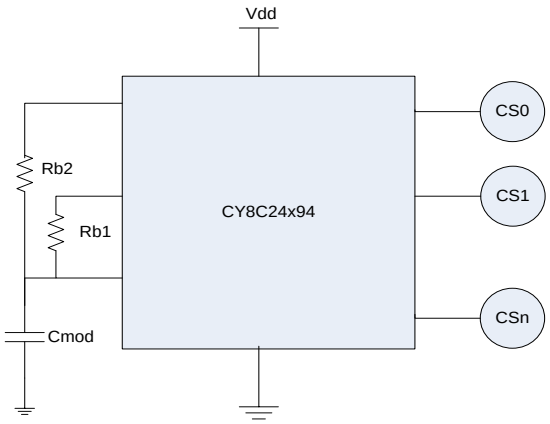
$$Rbmax = \frac{0.70*2^N}{2^N*\left[\frac{Sysclk}{4*Period}\right]*\left[\frac{1}{\frac{1}{4}+\frac{REF}{16}}-1\right]*Cp} \qquad \text{Equation 2}$$

$$Rbmin = \frac{0.60*2^N}{2^N*\left[\frac{Sysclk}{4*Period}\right]*\left[\frac{1}{\frac{1}{4}+\frac{REF}{16}}-1\right]*Cp} \qquad \text{Equation 3}$$

In these equations, the bleed resistor, Rb_{max}, gives a raw count value that is 70 percent of the maximum value, and Rb_{min} gives a raw count value that is 60 percent of the maximum value.

Figure 3 shows a CapSense design in which two bleed resistors, Rb1 and Rb2, of different values are used to tune all the sensors to 60 to 70 percent of the maximum value.

Figure 3. CapSense System with Two Different Bleed Resistors



The PSoC Designer™ project associated with this code example, Example_Dynamic_Bleed_Resistor, demonstrates how to switch the bleed resistor during run time to scan a sensor with a different bleed resistor. This is evaluated using the CY3280-SLM board. The SLM board has five buttons with different Cp values, as shown in Table 1. The value of CMOD and R_B used for the current project is shown in Table 2.

Table 1. Buttons C_p Values

Name	Parasitic Capacitance
Button 0	16 pF
Button 1	14 pF
Button 2	13 pF
Button 3	14 pF
Button 4	11 pF

Table 2. Cmod and Rbx Values

Name	Value
Cmod	10 nF
Rb1	7.5 kΩ
Rb2	10 KΩ

As shown in Table 1, button 4 has the lowest C_p of 11 pF. Using Equation 1, the raw count value for button 4 is calculated as:

$$\text{Counts} = 2^{12} * 7.5K * \left[\frac{24MHz}{4*1}\right] * \left[\frac{1}{\frac{1}{4}+\frac{4}{16}} - 1\right] * 11\ pF \qquad \text{Equation 4}$$

In this equation:

Resolution = 12 bits
Rb = 7.5 kΩ
Sysclk = 24 MHz
Period = 1
REF = 4
Cp = 11 pF

Therefore,

$$\text{Counts} = 4096 * 7.5K * [6MHz] * [1] * 11pF = 2027$$

Since the raw count value is 2027, this value is 49 percent of the maximum raw count value. Therefore, to ensure that the raw count is in the range of 60 to 70 percent of the maximum value, a different Rb is used for button 4.

Using Equation 3 and Equation 4:

$$\text{Rbmax4} = \frac{0.70*2^N}{2^N*\left[\frac{Sysclk}{4*Period}\right]*\left[\frac{1}{\frac{1}{4}+\frac{REF}{16}}-1\right]*Cp}$$

$$\text{Rbmax4} = \frac{0.70*2^{12}}{2^{12}*\left[\frac{24MHz}{4*1}\right]*\left[\frac{1}{\frac{1}{4}+\frac{4}{16}}-1\right]*11pF}$$

$$\text{Rbmax4} = \frac{0.70}{6MHz*11pF} = 10.6\ \text{k}\Omega$$

Similarly,

$$Rbmin4 = \frac{0.60}{6MHz * 11pF} = 9\ k\Omega$$

In this code example, Rbmax4 is used to tune button 4 to get a raw count value of 70 percent of the maximum value. The following sections describe the requirements, setup, and steps for running the code example.

Hardware Setup

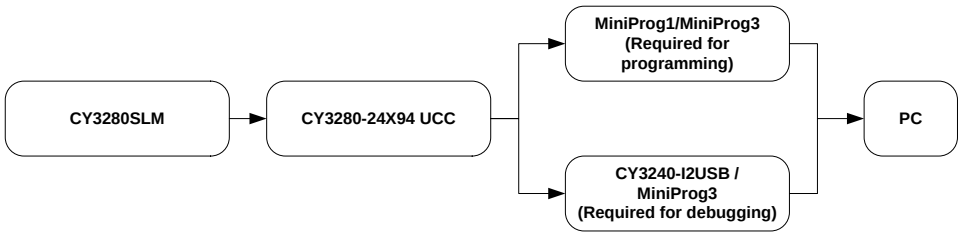
You need the following hardware:

- CY3280-24x94 Universal CapSense Controller Board
- CY3280-SLM Linear Slider Module
- CY3240-I2USB Bridge/MiniProg3
- CY3217-MiniProg1 Programmer Kit/MiniProg3
- USB A to Mini-B cable

Assembly

Figure 4 represents the hardware setup required to test the project. The CY3280-24x94 UCC Kit connects to the CY3280-SLM module through a 22x2_RA_Receptacle. The setup uses the MiniProg1 or MiniProg3 for programming and the CY3240-I2USB Bridge or MiniProg3 to send the data to a PC using the BCP. The MiniProg1/3 and CY3240-I2USB Bridge connect to the PC through a USB cable.

Figure 4. Hardware Setup Block Diagram



Setting Up the Board

Make the following hardware connections:

1. Unsolder the pull-up resistors, R57 and R58, which are connected to P1[5]–P1[7] on jumper J2, as shown in Figure 5.
2. Short the resistor pads, R15 and R20, as shown in Figure 5. This setting connects the P1[5]–P1[7] pins from the CY8C24x94 device to the jumper J2.
3. Add Rb2 by connecting the 10-kΩ resistor between P0[5] and P1[5], as shown in Figure 6.
4. Connect header J1 of the CY3280-SLM daughter card to the 22x2_RA_Receptacle (connector P2) on the CY3280-24x94 UCC board.
5. Place a jumper on header J1 to short the V_{CC} and 5-V pins of the UCC board. This setting allows you to power the CapSense controller from the in-system serial programming (ISSP) connector.
6. Place a jumper on header J4 to short the XRES and XRES/INT pins (pins 1 and 2) of the UCC board. This setting routes the XRES pin of the CapSense controller to pin 3 of the ISSP connector J3; this is required only if you need to program the device in the reset mode.
7. Place a jumper on header J2 to short the GND and SHIELD pins (pins 2 and 3) of the CY3280-SLMboard. This setting allows the board hatch pattern on the CY3280-SLM board to connect to the ground.
8. Connect the MiniProg1/MiniProg3 to the ISSP header on J3 of the UCC board. This connection is required to program the device with the generated hex file. This connection must be replaced with the CY3240-I2USB Bridge/MiniProg3 when the CapSense data is read in the BCP software.
9. Use the USB A to Mini-B cable to connect the other end of the MiniProg1/CY3240-I2USB Bridge/MiniProg3 to the PC.

Figure 5. Removing Pull-Up Resistors

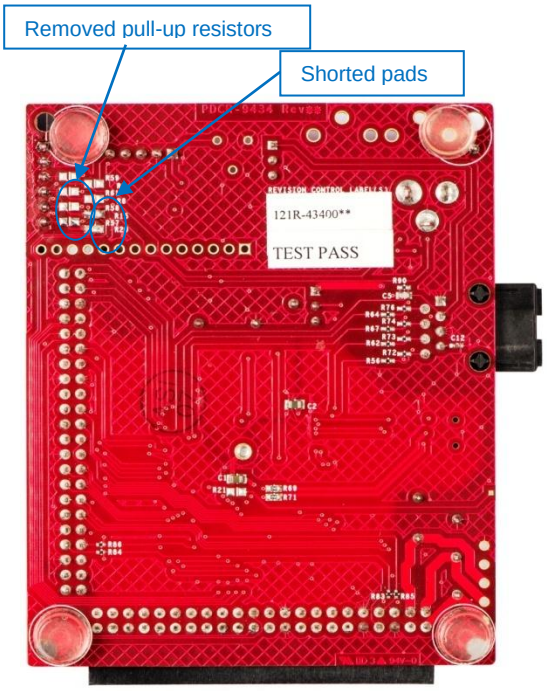
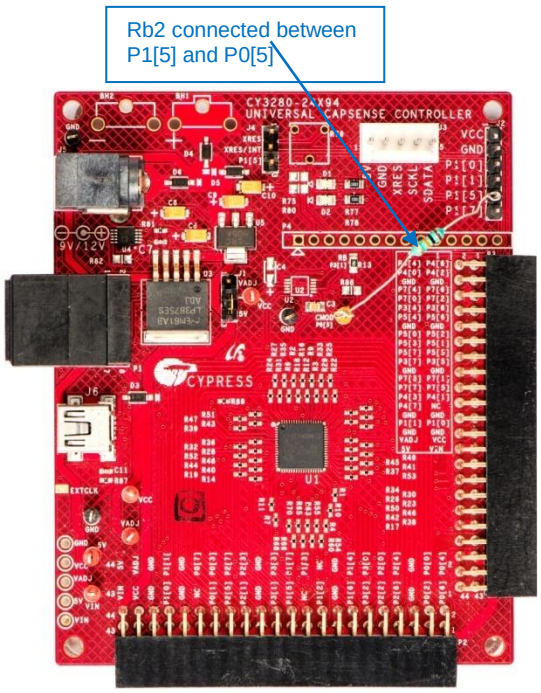


Figure 6. Connecting Rb2 between P1[5] and P0[5]



Schematic

The schematic diagram for the project is shown in Figure 7. The modulator capacitor, Cmod, of 10-nF is connected to P0[5]. Rb1 is connected to P3[1], and Rb2 is connected to P1[5]. A 560-Ω resistor is connected in series with each CapSense button to reduce RF interference.

Figure 7. Schematic Diagram

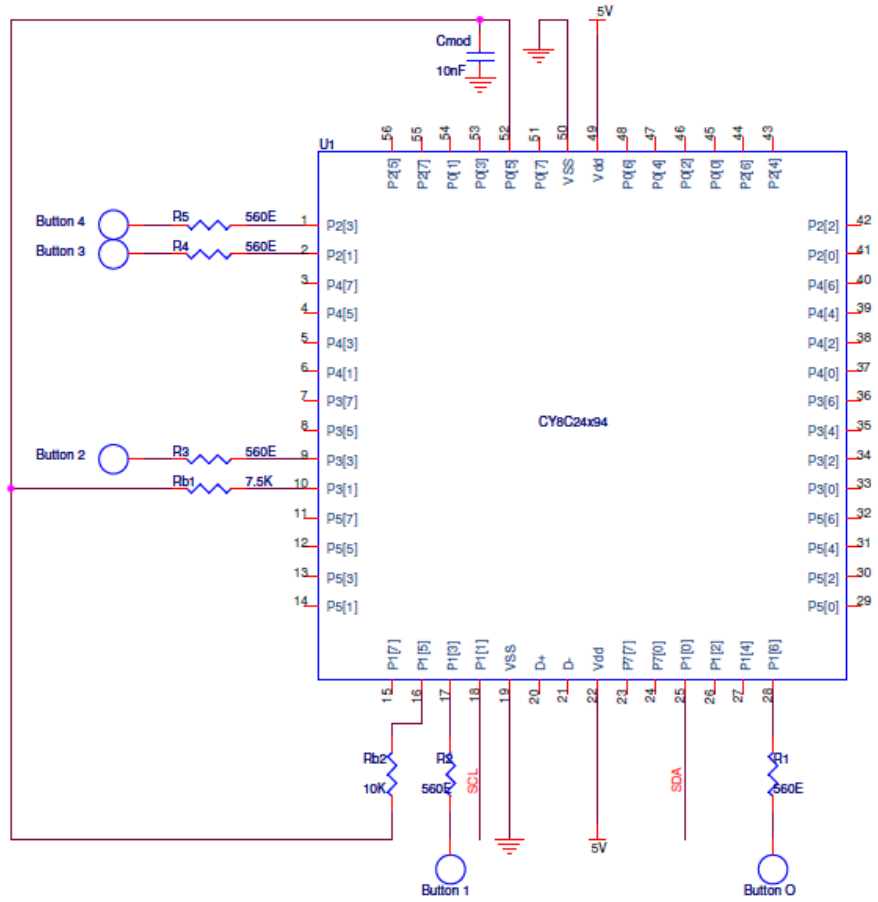


Table 3. Buttons, Cmod, and Rbx Pin Assignment

Name	Pin Number
Button 0	P1[6]
Button 1	P1[3]
Button 2	P3[3]
Button 3	P2[1]
Button 4	P2[3]
Cmod	P0[5]
Rb1	P3[1]
Rb2	P1[5]

Software Setup

You need the following tools to set up the software:

- PSoC Designer (version 5.3 or higher)
- PSoC Programmer (version 3.13 or higher)
- BCP

The user modules (UMs) used in this project and the hardware resources occupied by each user module are listed in Table 4.

Table 4. User Modules and Hardware Resources

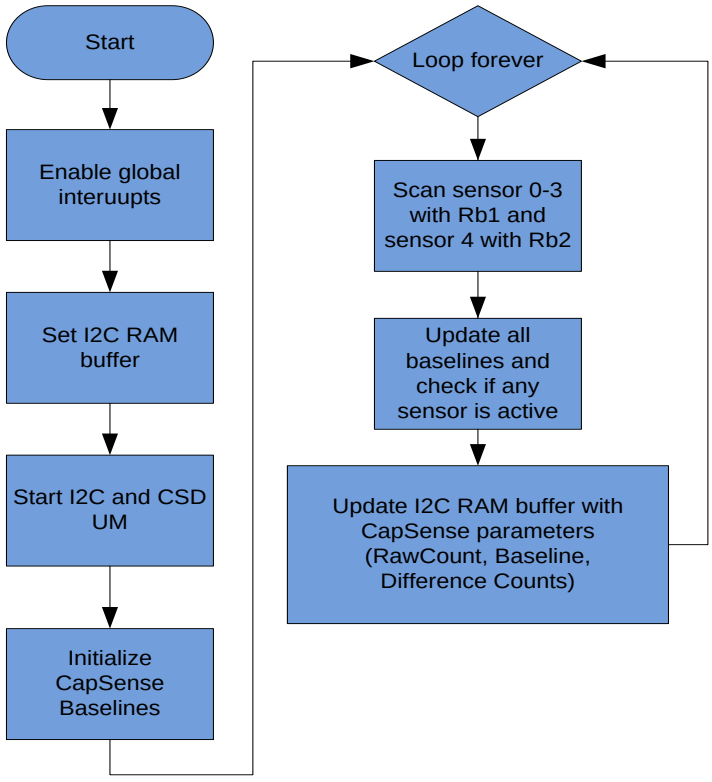
User Module	Hardware Resources
CSD	ACB01, DBB00, DBB01, DCB02
EzI2Cs	I^2C Block

The *ReadMe.txt* file, available in the project, describes the parameter settings for the user modules used in this code example. It also includes a list of the global resources.

Operation

Figure 8 shows the firmware flow of the project.

Figure 8. Functional Flow of Firmware



On reset, all the hardware settings from the device configuration are loaded into the device and *main.c* is executed.

Next, the firmware performs the following operations:

1. Defines a structure, MyI2C_Regs, to store the button number, raw count, difference count, baseline, and status of the CapSense button corresponding to the given button number.
2. Enables a global interrupt and starts the CSD User Module. The finger thresholds for buttons are set, and the baselines are initialized (with the respective feedback resistors connected).
3. Starts the EzI2Cs User Module and sets MyI2C_Regs as the I^2C buffer.
4. Performs the following operations in an infinite loop:
    - Scans buttons 0–3 with feedback resistor Rb1 and button 4 with feedback resistor Rb2. MyI2C_Regs is updated with the raw count, difference count, baseline, and status of the requested CapSense button.
    - The I2C master can request the CapSense data of a particular button by writing the button number into the first byte of the I2C buffer of the EzI2Cs slave.

Changing the Feedback Resistor

In the example project, buttons 0–3 use a bleed resistor connected to P3[1], while button 4 uses the bleed resistor connected to P1[5]. Before scanning, a function is called to modify the routing of the comparator output that drives the new bleed resistor. This is achieved by modifying these registers: PRTxDMx, PRTxGS, and CMP_GO_EN.

Following is a brief description of the functionality of these registers:

- **PRTxDMx**: These registers set the configuration of the output; the bleed resistor must be put in the open drain low mode. The pin that has to be disconnected from Rb must be made High Z. To switch between these modes, you need to change only the PRTxDM0 mode. You do not have to alter the DM1 and DM2 registers. The port number for the corresponding pin is x.
- **PRTxGS:** This register connects the pin of the resistor to the proper internal global bus. The port number for the corresponding pin is x.
- **CMP_GO_EN:** This register sets the comparator output to the global bus.

Note For more information about these registers, refer to the device TRM available at www.cypress.com/?rID=34621.

The values shown in Table 5 must be ORed with these registers to enable a pin for Rb operation.

Table 5. Values to Be ORed

Pin	PRTxDM0	PRTxGS	CMP_GO_EN
P1[1]	0x02	0x02	0x40
P3[1]	0x02	0x02	0x40
P1[5]	0x20	0x20	0x80
P3[5]	0x20	0x20	0x80
P5[1]	0x02	0x02	0x40
P5[5]	0x20	0x20	0x80

The code snippet used in this project to connect Rb to P5[5] is:

```
CMP_GO_EN |= 0x80
 PRT5DM0 |= 0x20
 PRT5GS |= 0x20
```

The values shown in Table 6 are ANDed with the same registers to disconnect Rb from the pin.

Table 6. Values to Be ANDed

Pin	PRTxDM0	PRTxGS	CMP_GO_EN
P1[1]	0xFD	0xFD	0xBF
P3[1]	0xFD	0xFD	0xBF
P1[5]	0xDF	0xDF	0xFB
P3[5]	0xDF	0xDF	0xFB
P5[1]	0xFD	0xFD	0xBF
P5[5]	0xDF	0xDF	0xFB

The following is the code snippet used in this project to connect Rb to P5[5]:

```
PRT5GS &= 0xDF
 PRT5DM0 &= 0xDF
 CMP_GO_EN &= 0xFB
```

Running the Code

Programming the Board

Program the board with the project, and then use the following procedure to run the code example. To learn how to program the UCC board, refer to the CY3280-24x94 UCC Kit Guide. Use the MiniProg1 or any power sources mentioned in the CY3280-24x94 UCC Kit Guide to power the board at 5 V.

Reading CapSense Data over I²C

Use this procedure to run the code example and read back the CapSense data to the BCP tool. For more information about the BCP, refer to the application note *AN2397 – CapSense Data Viewing Tools*.

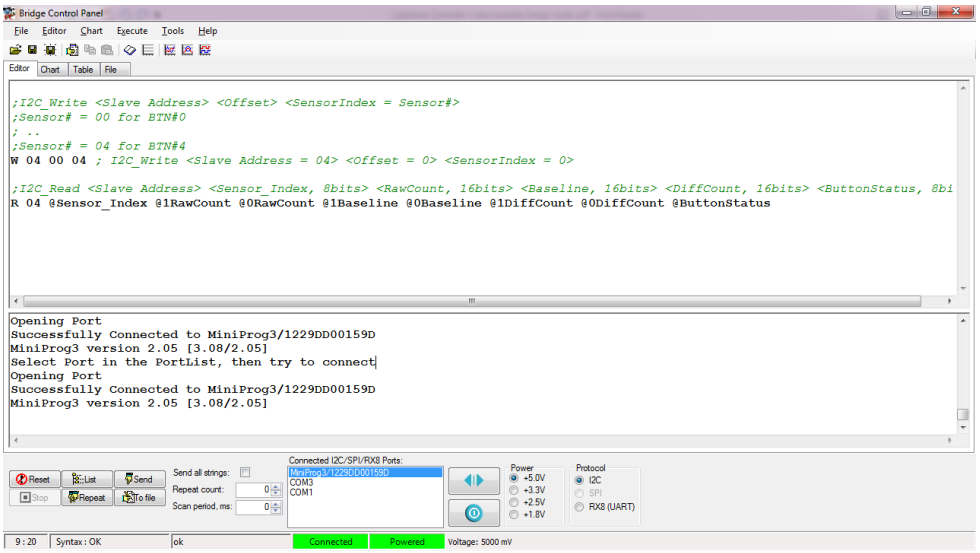
Viewing Data through the BCP

1. Use the CY3240-I2USB Bridge and a USB A to Mini-B cable to connect your computer to ISSP connector J3 of the CY3280-24x94 UCC board.
2. Go to **Start** > **All Programs** > **Cypress** > **Bridge Control Panel (version)** > **Bridge Control Panel (version)**. The BCP is a component that is installed in your computer during the PSoC Programmer installation.
3. Select the device from the port selection window.
4. Power the CY3280-24x94 UCC board at 5 V.
5. From the BCP, choose **File** > **Open**. Load the *BCP.iic* file from the BCP_Configuration_Files folder in the code example project folder.
6. Choose **Charts** > **Variable Settings**. Load the *BCP.ini* file from the BCP_Configuration_Files folder in the code example project folder.
7. Click **OK** to return to the main window.

To read the raw count, baseline, and difference count of BTN4, proceed as follows:

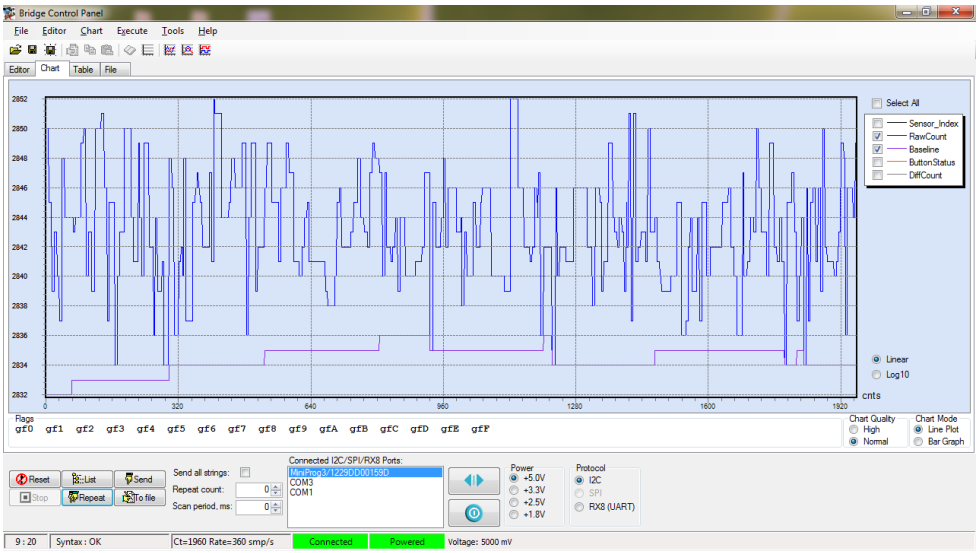
1. Place the cursor on the command line **W 04 00 04** and click on send button once to send the I²C write instruction.
2. Place the cursor on the command line R 04 @Sensor_Index @1RawCount @0RawCount @1Baseline @0Baseline @1DiffCount @0DiffCount @ButtonStatus and click on the **Repeat** button to send I²C read instruction continuously:

Figure 9. Command Editor Window in Bridge Control Panel



3. Click the **Chart** tab to view the raw count, baseline, and difference count of BTN4, as shown in Figure 10.

Figure 10. Chart Tab in Bridge Control Panel



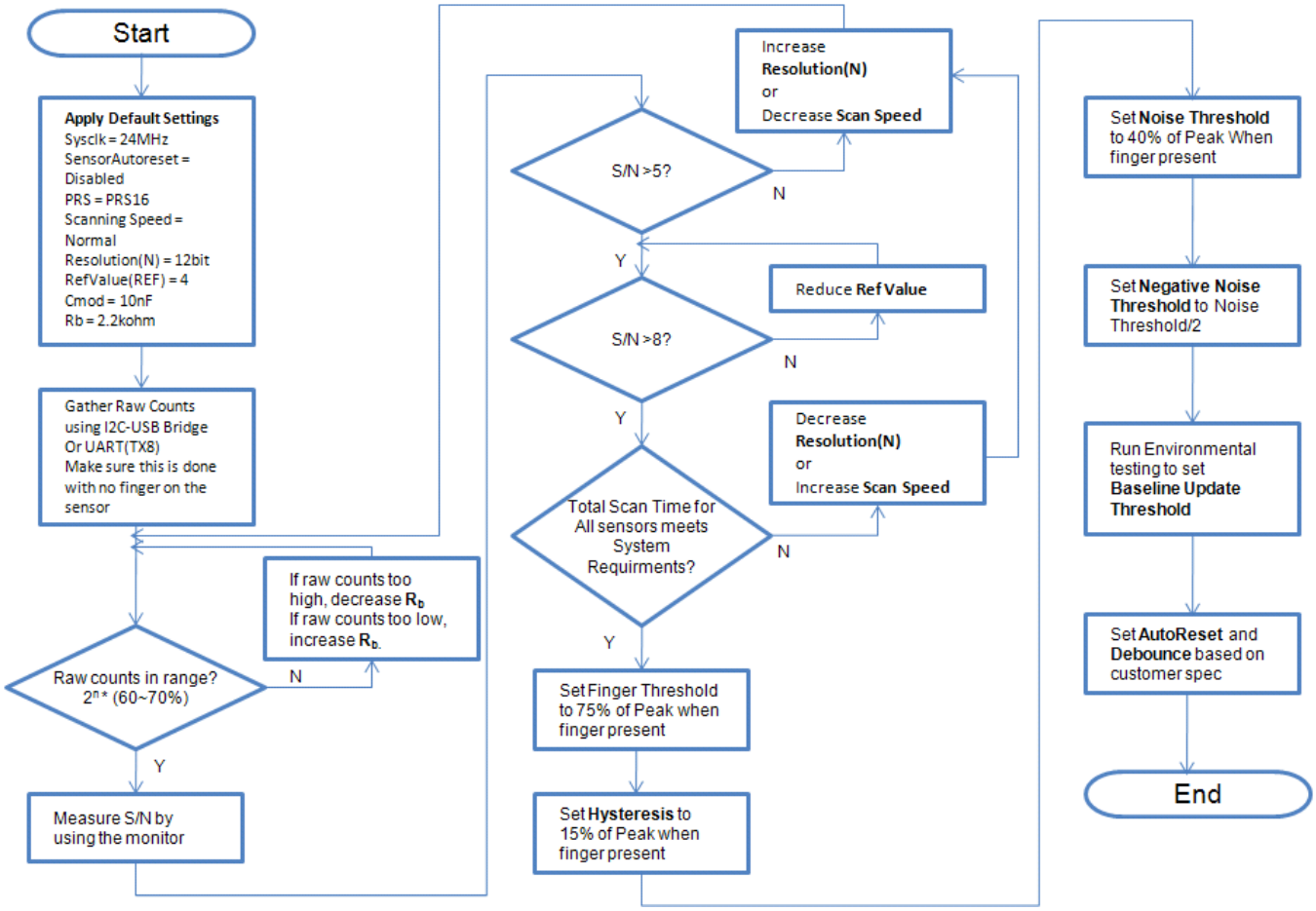
As shown in the BCP chart, the raw count is now calibrated to about 70 percent of the maximum value:

- Button 4 raw count with single Rb (Rb1) in CSD design = 2047 (49 percent of maximum value)
- Button 4 raw count with separate Rb (Rb2) in CSD design = 2844 (70 percent of maximum value)

Appendix: CSD Calibration

For optimum performance, the CSD parameters are tuned with the actual CapSense hardware and overlay. Figure 11 shows the steps necessary to calibrate the CSD.

Figure 11. CSD Calibration Flowchart



1. Start with the default settings of the CSD User Module.
2. Using the CY3240-I2USB Bridge/UART with the actual hardware and overlay, capture the raw counts, baseline, and difference counts for the sensors.
3. Coarse tuning: Check if the signal-to-noise ratio (SNR) is greater than 5. If the SNR is less than 5, increase the SNR using the recommended PCB guidelines, increasing the resolution of the CSD and reducing the scan speed of the CSD. For more information about the PCB guidelines, refer to the Getting Started with CapSense Design Guide.
4. Fine tuning: Check if the SNR is greater than 8. If it is less than 8, reduce the **Ref Value** parameter to increase the SNR.
5. Check if the total scan time for all the sensors meets the requirement. If it does not, reduce the resolution and/or increase the scan speed. As these parameters also affect the SNR, go back to step 3. With a couple of passes, arrive at the optimum resolution and scan the speed parameters that produce the best SNR and the desired scan time.
6. Capture the difference counts when the button is activated. Set the **finger threshold** parameter to 75 percent of the peak value.
7. Set the noise threshold to 40 percent of the peak value.
8. Set the negative noise threshold to half of the noise threshold.

9. Set the finger thresholds for individual sensors if necessary, by writing to the `CSD_baBtnFThreshold` array in the firmware.

   Set the baseline update threshold according to the requirements. The frequency that the baseline is updated must be determined on a project-by-project basis. The baseline should be a slow-moving reference, which helps to reduce the effects of noise and temperature on the capacitive sensor.

   - Fast update baseline rates: This can cause problems if you move your finger slowly to the button. It is called “baselining out the finger.”
   - Slow update baseline rates: This can leave the buttons vulnerable to temperature fluctuations, and potentially lead to a “button lock.”

10. Set the **AutoReset** and **Debounce** parameters as required. Refer to the CSD User Module datasheet for more information about these parameters.
11. For any other parameter, refer to the user module datasheet.

Document History

Document Title: CE66058 - Dynamically Switch CSD Bleed Resistor

Document Number: 001-66058

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3112684	PRIA	12/16/2010	Initial release.
*A	4075620	DCHE	07/16/2013	Updated Setting Up the Board: Updated Figure 5 (with high-resolution image). Updated Figure 6 (with high-resolution image).
*B	4089222	DCHE	08/07/2013	Added updated associated project files.
*C	4219404	DCHE	12/13/2013	No technical updates. Completing Sunset Review.
*D	4582144	DCHE	12/12/2014	Replaced “I2USB Bridge” with “CY3240-I2USB Bridge” in all instances across the document.
*E	5570474	DCHE	01/11/2017	Updated to new template. Completing Sunset Review.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2010-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.