Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CE54365 – CSD with I2CHW Slave on CY8C21x34/B

Objective

This code example demonstrates how to use CapSense Sigma Delta (CSD) User Module (UM) in CY8C21x34/B device to scan CapSense button sensors and send the sensor data to the master using I2C protocol.

Overview

This code example shows how to implement CapSense button sensors with the CY8C21x34/B device using the CSD UM in the PSoC Designer™ Integrated Design Environment (IDE). It also shows how to send the sensor data to the I^2C master using the I2CHW UM.

This code example can be tested with the CY3280-21x34 Universal CapSense Controller (UCC) board.

PSoC Resources

Cypress provides a wealth of data at www.cypress.com to help you to select the right PSoC® device for your design and quickly and effectively integrate the device into your design. For a comprehensive list of resources, see KBA92181 – Resources Available for CapSense Controllers. The following is an abbreviated list for CapSense devices:

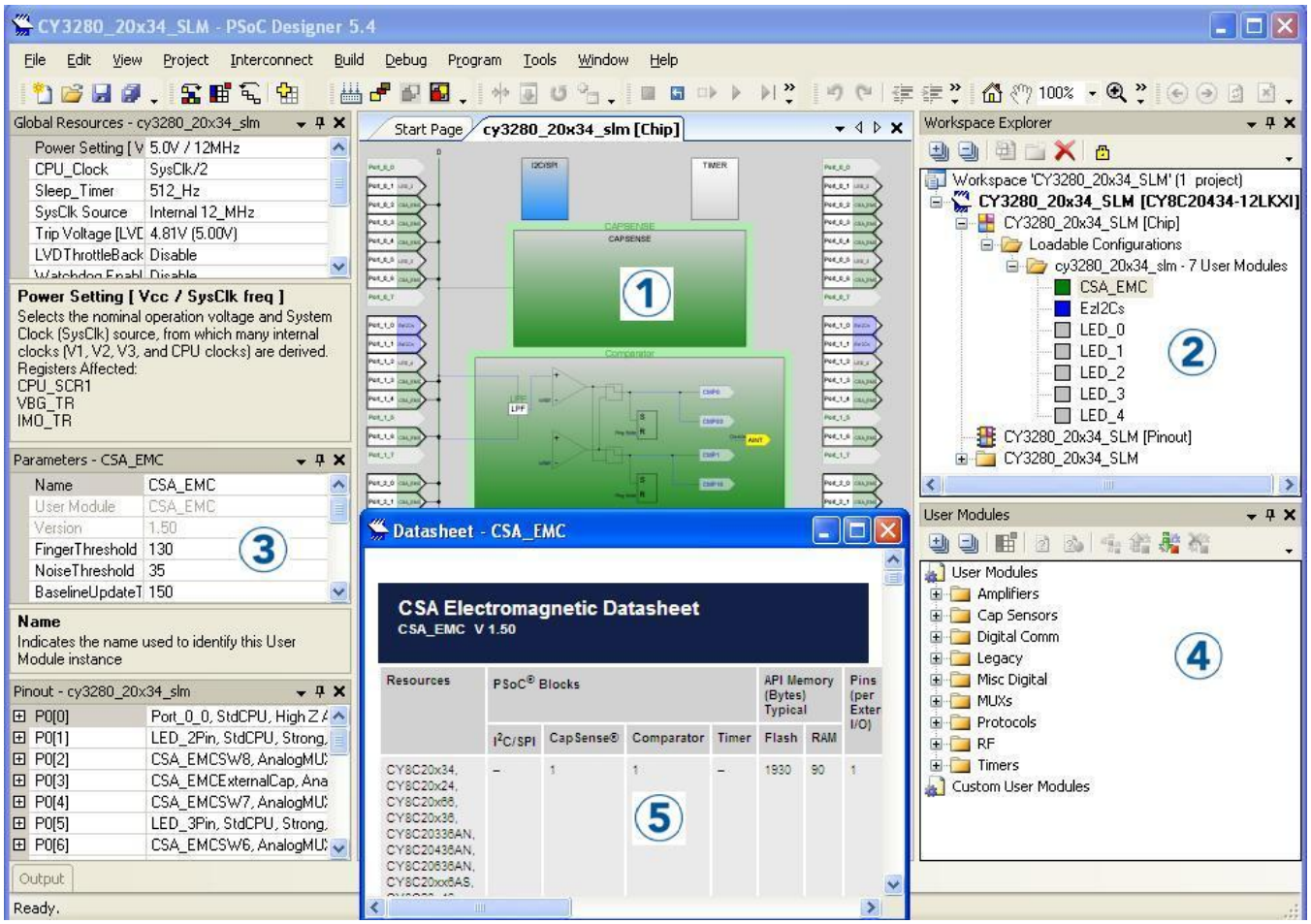
- **Overview:** CapSense Portfolio, CapSense Roadmap
- **Product Selectors:** CapSense, CapSense Plus™, CapSense Express™, PSoC 3 with CapSense, PSoC 5 with CapSense, PSoC 4. In addition, PSoC Designer offers a device selection tool when you create a new project.
- **Datasheets**: Describe and provide electrical specifications for the CapSense, PSoC 1, PSoC 3, PSoC 4, and PSoC 5LP device families.
- **Application Notes:** Cypress offers CapSense application notes covering a broad range of topics, from basic to advanced level. Recommended application notes for getting started with CapSense are the following:
  - AN64846 – Getting Started With CapSense
  - CY8C21x34/B – CapSense Design Guide
  - AN2397 – PSoC 1 and CapSense Controllers – CapSense Data Monitoring Tools
- **Technical Reference Manuals (TRM):** Provide detailed descriptions of the architecture and registers in the CapSense, PSoC 1, PSoC 3, PSoC 4, and PSoC 5LP device families.
- **Development Kits:**
  - CY3280-BK1 – Universal CapSense Controller – Basic Kit 1 features a predefined control circuitry and plug-in hardware to make prototyping and debugging easy. The kit includes MiniProg1 programming hardware for device programming and CY3240-I2USB board hardware for tuning and data acquisition.
  - CY3280 – SLM Linear Slider Module Kit consists of 5 CapSense buttons, 1 linear slider (with 10 sensors), and 5 LEDs. This module connects to the CY3280 UCC board, including the CY3280-21x34 UCC board.
  - CY3280-BBM – Matrix Button Module Kit consists of 8 LEDs as well as 8 CapSense sensors organized in a 4x4 matrix format to form 16 physical buttons.
- **Programming Resources:**
  - PSoC supports a number of different programming modes and tools. For more information, see the General Programming page.

PSoC Designer

PSoC Designer is a free, Windows-based IDE. It enables the hardware and firmware to be designed concurrently for systems based on PSoC 1 and CapSense devices; see Figure 1. With PSoC Designer, you can:

- Drag and drop UMs to build your hardware system design in the main design workspace
- Codesign your application firmware with the PSoC hardware
- Configure UMs
- Explore the library of UMs
- Review UM datasheets

Figure 1. PSoC Designer Features



Requirements

Tool: PSoC Designer 5.4 SP1, Bridge Control Panel (BCP)

Programming Language: C (ImageCraft STD and PRO compilers)

Associated Parts: CY8C21x34/B device series

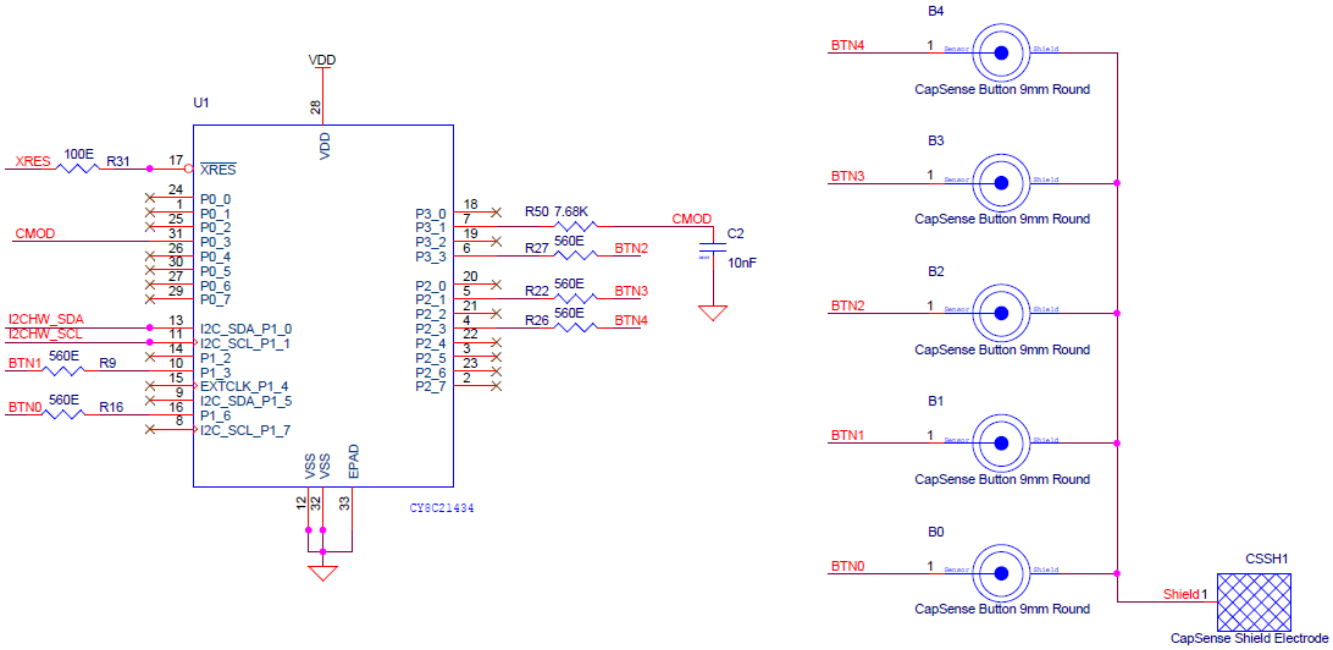
Related Hardware: CY3280-BK1 – Universal CapSense Controller – Basic Kit 1

Design

Figure 2 shows the schematic design of the code example. The code example features the following:

- CSD UM to detect a finger touch on the CapSense button sensor
  - The CSD UM is configured to scan five CapSense button sensors.
  - All CSD parameters are manually set to achieve optimum touch sensing performance.
- I2CHW UM to send the CapSense button sensor data to the I²C master
  - The sensor data (Raw Count, Baseline, Difference Count, and Status) of the specified button sensor is stored in the I2C read buffer.

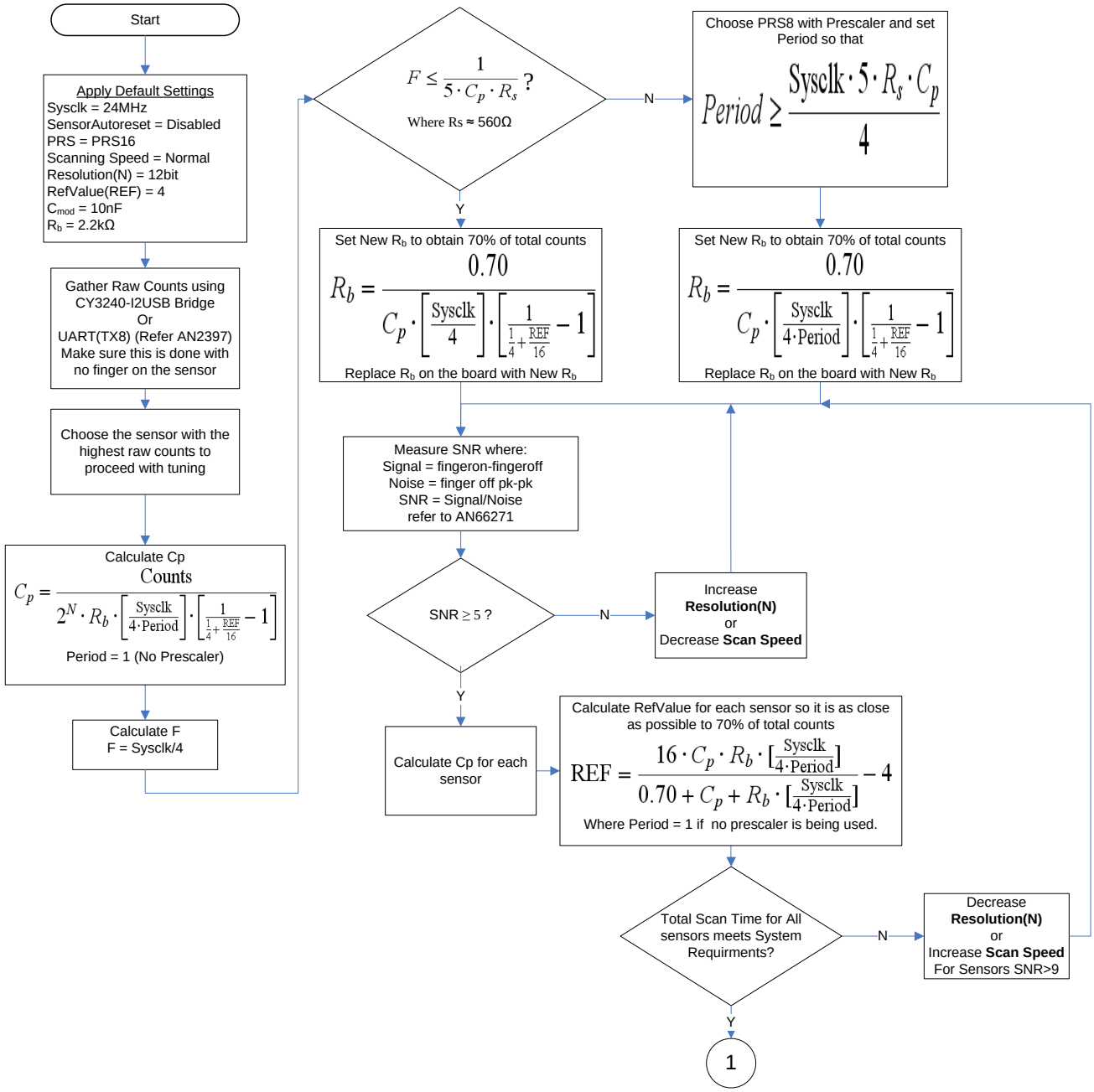
Figure 2. CSD Design with CY8C21434

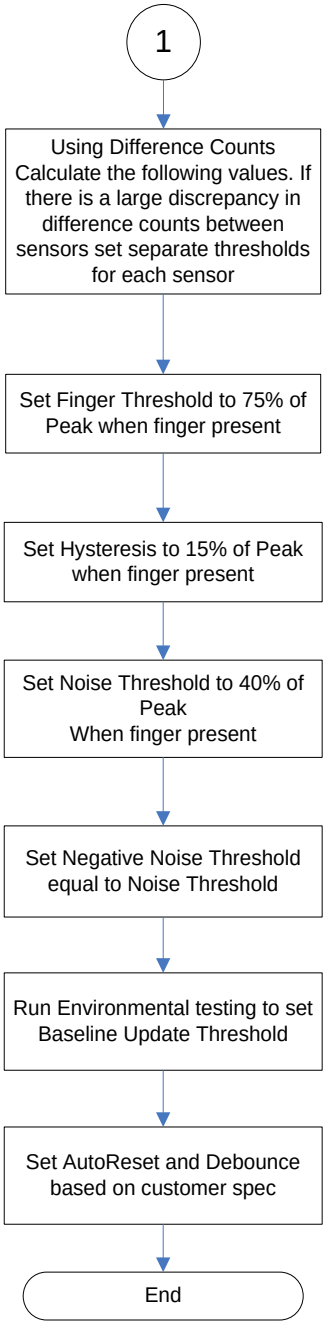


Design Considerations

In the code example, the resolution parameter is set to '12', assuming the overlay thickness on the sensor board is 1.5 mm. To test the code example with an overlay of a different thickness, set the resolution parameter so that the ratio of difference count to noise count is greater than 5:1. Thicker overlays require you to scan at a higher resolution to achieve an SNR greater than 5:1. Refer to the flow chart in Figure 3 to tune the CSD parameters to achieve an SNR > 5:1. For the detailed tuning procedure, refer to the CY8C21x34/B – CapSense Design Guide.

Figure 3. CSD Tuning Flow Chart





Hardware Setup

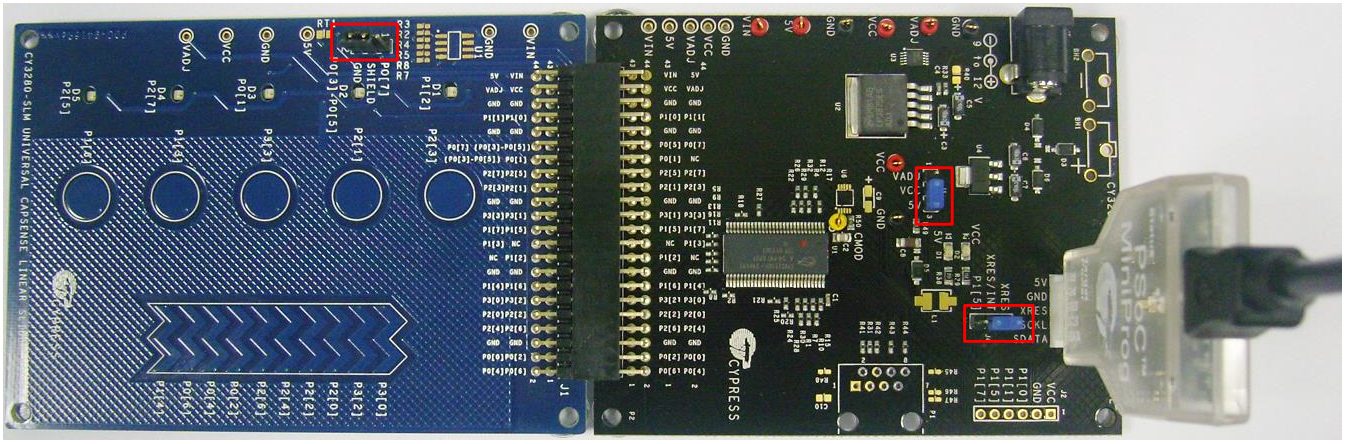
For the basic kit board setup, see the corresponding kit user guide.

The code example can be tested on CY3280-21x34 UCC board with the CY3280-SLM board. Follow these steps to set up the hardware:

1. Connect the CY3280-SLM board to the P2 receptacle connector of the CY3280-21x34 UCC board, as shown in Figure 4.
2. On the CY3280-21x34 UCC board, place a jumper on header J1 to short pins 2 and 3. This setting allows the CapSense controller to be powered from ISSP connector J3.

3. On the CY3280-21x34 UCC board, place a jumper on header J4 to short pins 1 and 2. This setting routes the XRES pin of the CapSense controller to pin 3 of ISSP connector J3.
4. On the CY3280-SLM board, place a jumper on header J2 to short pins 2 and 3. This setting connects the shield traces on the CY3280-SLM board to ground.
5. Connect your PC to the CY3280-21x34 UCC board's ISSP connector J3 using MiniProg1 or MiniProg3 and a USB cable.

Figure 4. Hardware Setup for Testing Code Example



Software Setup

The code example works on CY3280-21x34 UCC board with the CY3280-SLM board without any modification. To test the code example on a custom board, reassign the sensor pins and retune the CSD parameters according to the flow chart in Figure 3. To reassign the sensor pins, refer to the “CSD Wizard” section in the CapSense Sigma-Delta UM datasheet.

User Modules

Table 1 lists the PSoC Designer UMs used in the code example, as well as the hardware resources used by each.

Table 1. PSoC Designer User Modules

User Modules	Hardware Resources
CSD	ACE00, ACE01, ASE11, DBB00, DBB01, DCB02
I2CHW	No blocks occupied (uses dedicated hardware)

Parameter Settings

Table 2 to Table 4 show the parameter settings for the UMs used in the code example. Only the parameters that vary from the default are shown.

Table 2. CSD User Module Settings

Parameter	Value	Comments
Finger Threshold	75	After the difference count crosses the finger threshold plus hysteresis, the button is said to be in “ON” condition.
Noise Threshold	40	If the difference count is less than the noise threshold, it is treated as noise, and the baseline update algorithm handles it by putting it into the update bucket.
Baseline Update Threshold	100	As the noise increases, the update bucket is filled, and every time it crosses this threshold, the baseline is incremented by ‘1’ and the algorithm continues.

Parameter	Value	Comments
Sensors Autoreset	Disabled	When the parameter is set to “disabled,” the baseline is updated only when the raw count and baseline difference is below the noise threshold parameter.
Hysteresis	15	This parameter handles false ON and OFF situations whenever the button is pressed. Set it equal to the noise threshold.
Debounce	3	If the difference count is more than the finger threshold for less than the “debounce” number of samples, it is not taken as a button press.
Negative Noise Threshold	40	If the raw count is below the baseline and the difference count is more than this threshold, the baseline is not updated.
Low Baseline Reset	50	If the raw count is below the baseline and the difference count is more than the negative noise threshold for the number of samples given by this parameter, the baseline is reset to a new raw count.
Scanning Speed	Normal	This parameter decides the speed of the scanning process.
Resolution	12	The higher the resolution, the higher is the sensitivity.
Modulator Capacitor Pin	P0[3]	This parameter indicates to which pin Cmod is connected.
Feedback Resistor Pin	P3[1]	This parameter indicates to which pin Rb is connected.
Reference	ASE11	This parameter indicates the source for reference voltage. ASE11 is preferred.
Ref Value	2	This parameter sets the comparator reference value.
Prescaler Period	3	This parameter sets the prescaler period register and determines the precharge switch output frequency.
ShieldElectrodeOut	None	The shield electrode is not used in this code example.

Table 3. I2CHW User Module Settings

Parameter	Value	Comments
Slave Address	4	This parameter decides the address that is assigned to the slave. The value assigned can be any value from 0 to 127 (decimal).
Read_Buffer_Types	RAM ONLY	Only the RAM data buffer is used.
Communication_Service_Type	Interrupt	See the following notes.
I^2C Clock	100k Standard	This parameter decides the maximum clock speed at which the slave can operate.
I^2C Pin	P1[0]–P1[1]	This parameter specifies which pins are used as the SDA and SCL lines of I^2C.

Notes:

- When the Read_Buffer_Types is set to RAM ONLY, only RAM buffers may be transmitted over I^2C. If data from the flash buffer is to be read and transmitted, the read buffer type should be set to RAM OR FLASH.
- In an interrupt-based Communication_Service_Type, data is moved in and out of the buffer very quickly in the background using an ISR.
- The I^2C clock is dependent on the SysClk. The I^2C clock setting in the UM is based on a SysClk of 24 MHz. In devices that support slower a Sysclk, the I^2C clock is reduced by the same proportion. For example, if the I^2C clock is set to 400 kHz and SysClk is set to 6 MHz, the actual I^2C clock is only 100 kHz.

Table 4. Global Resources Settings

Parameter	Value	Comments
Power Setting (Vcc/SysClk frequency)	5.0 V/24 MHz	Selects 5-V operation and 24-MHz SysClk.
CPU_Clock	SysClk/2	Selects 12 MHz as CPU clock.

Note: Other parameters are left at their default values.

Operation

Build and install the example in the CY3280-21x34 UCC board . Refer to the kit user guide for the procedure to build and install the example project.

On reset, all the hardware settings from the device configuration are loaded into the device and *main.c* is executed. The following operations are performed by the firmware:

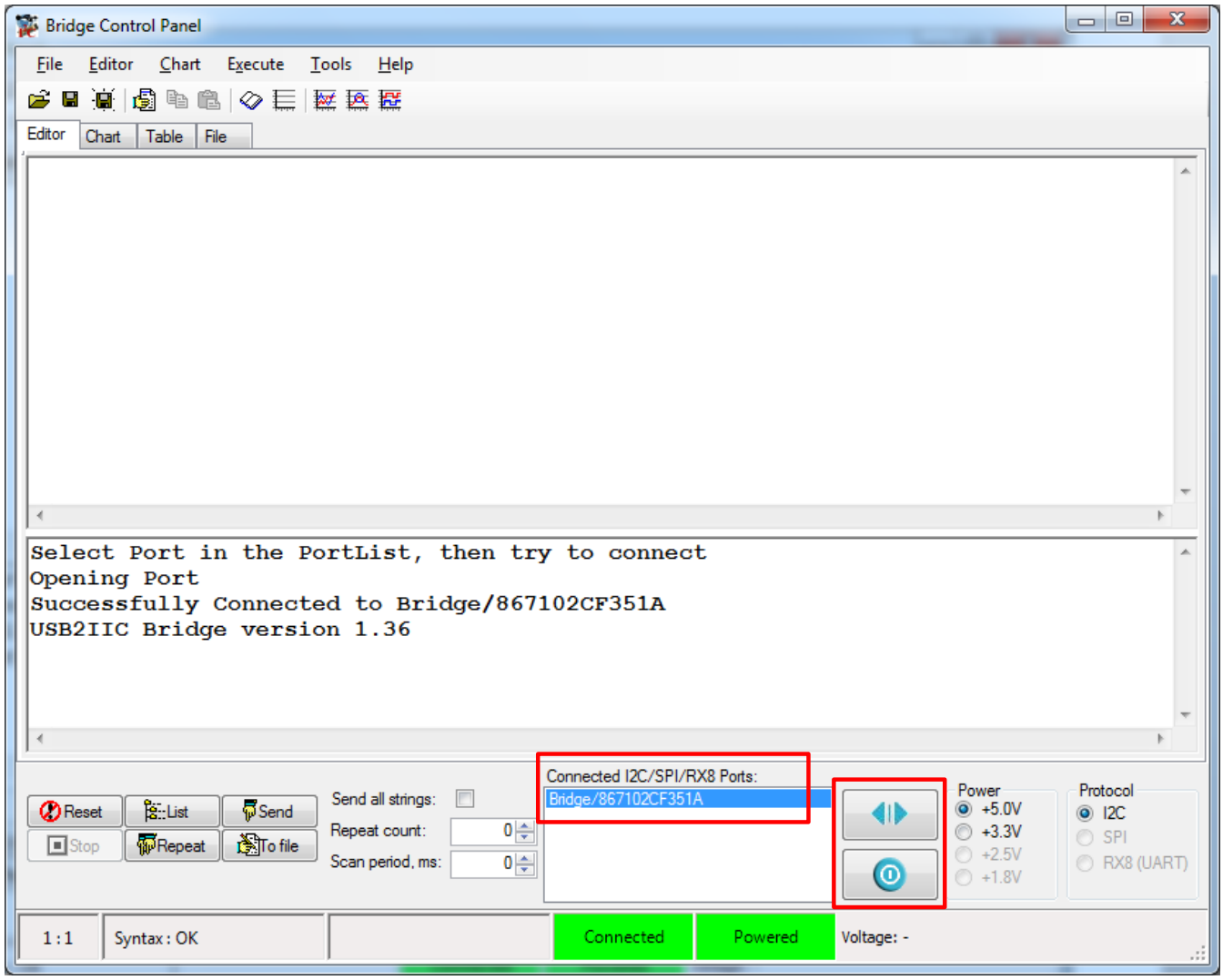
- A structure (sI^{2}CRegs) is defined to store the button number, raw count, difference count, baseline, and status of the CapSense button.
- A global interrupt is enabled and the CSD UM is started, finger thresholds for the buttons are set, and baselines are initialized.
- The I2CHW UM is started. Structure “sI^{2}CRegs” is set as the I^{2}C read buffer, and the BYTE variable “buttonNumber” is set as the I^{2}C write buffer.
- To get information about a button, the I^{2}C master must write the button number into the I^{2}C write buffer (buttonNumber). The I^{2}C master can get that button information by reading the I^{2}C read buffer (sI^{2}CRegs).
- In an infinite while loop, all the CapSense buttons are scanned, and all the baselines are updated. Also, sI^{2}CRegs is updated with the raw count, difference count, baseline, and status of the requested CapSense button.
- Whenever the I^{2}C master writes (or reads) into the buffer, the Write flag (Read flag) must be reset and the buffer must be set again. This operation is done in the while loop.

Test the example by doing the following:

1. Open the BCP software from **Start** > **All Programs** > **Cypress** > **Bridge Control Panel [version]** > **Bridge Control Panel [version].**
2. Connect the CY3240 USB-I2C Bridge to header J3 on the CY3280-21x34 UCC board. Refer to the CY3240-I2USB USB-I2C Bridge Guide for information on how to connect the CY3240 USB-I2C Bridge to any kit for I^{2}C communication.

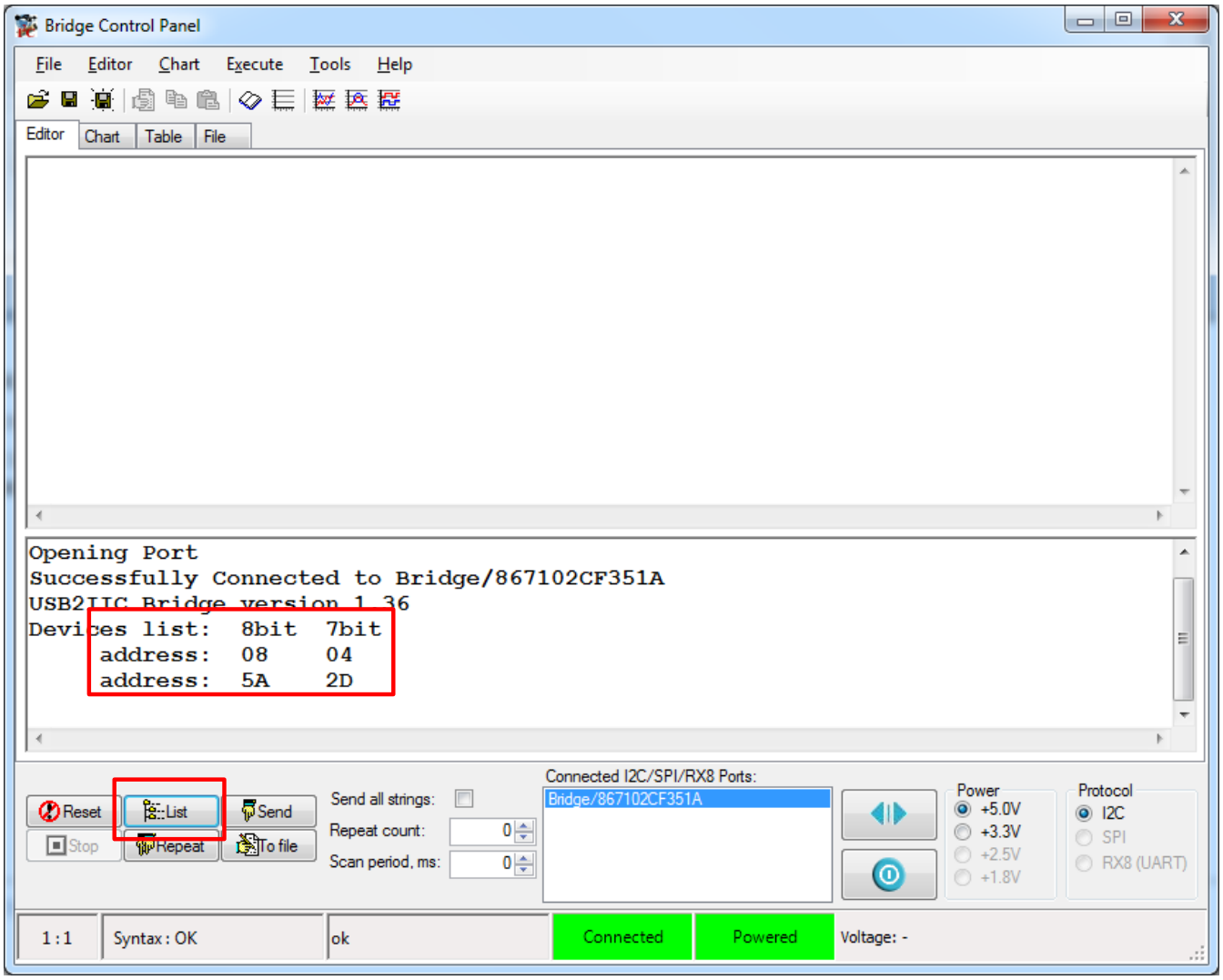
   If you are using MiniProg3, see the MiniProg3 User Guide for information on how to connect the I^{2}C pins of the CapSense controller to MiniProg3.
3. In the BCP software, ensure that the USB-I2C Bridge or MiniProg3 is listed in the **Connected I2C/SPI/RX8 Ports**: tab, as shown in Figure 5.

Figure 5. Selecting the USB-I2C Bridge or MiniProg3 in BCP



4. Choose **Tools** > **Protocol Configuration,** select **I2C Speed – 100 kHz**, and click **OK**.
5. At the right bottom corner under **Power**, select the **+5.0V** radio button. This powers the target device with 5 V
6. At the left bottom corner, click the **List** button. This lists all the slave device addresses. In this case, the device address 04 is the slave address of the CapSense controller, and it is displayed in the status window, as shown in Figure 6.

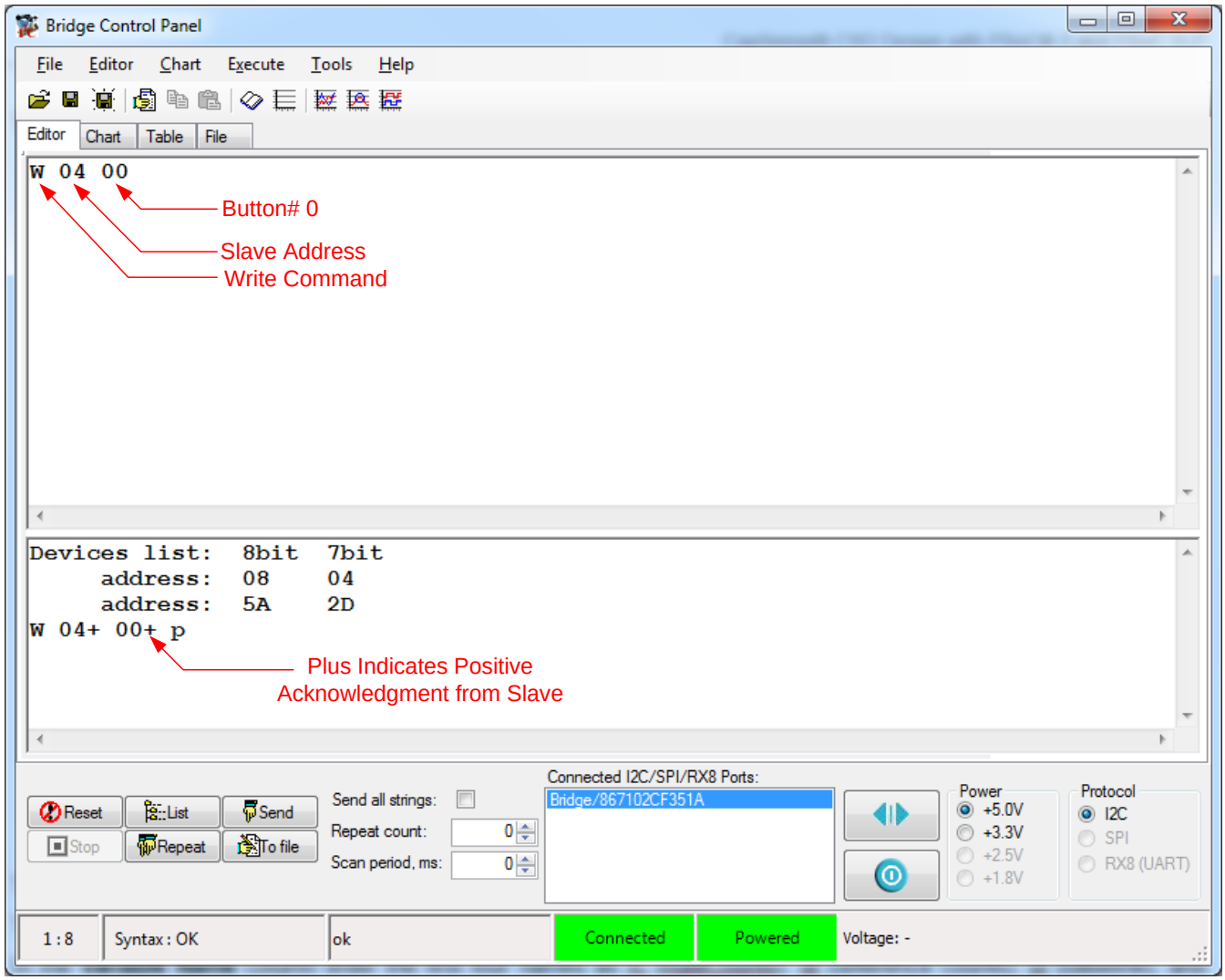
Figure 6. Display of Slave Address in Status Window



7. To monitor the CapSense data of button 0, write the following command in the command window (Figure 7) and press Enter.

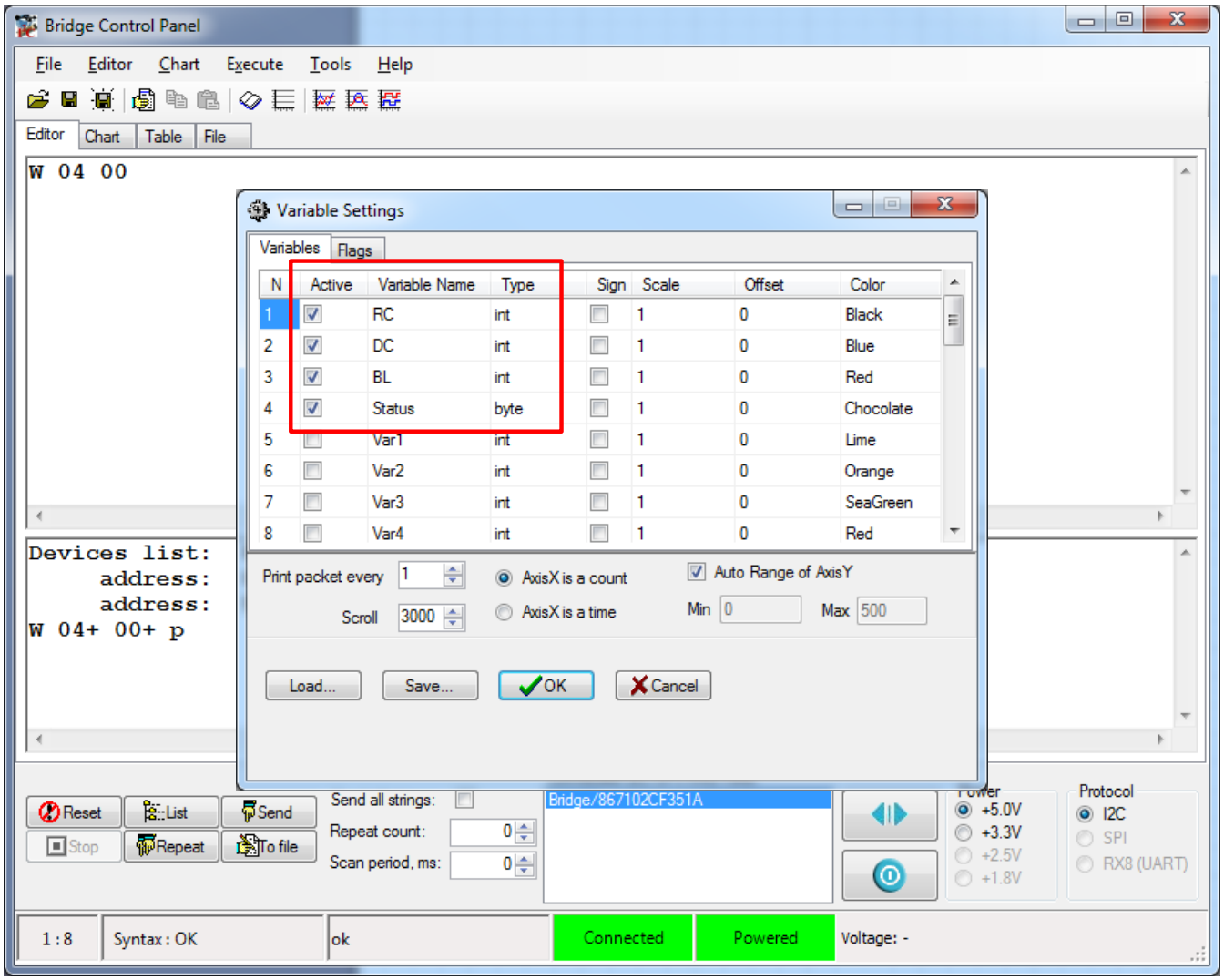
```
W 04 00
```

Figure 7. Write Command Execution



8. To view CapSense data as a graph, click the **Chart** tab and select **Variable Settings.**
9. In the **Variable Settings** window, select the **Variables** tab, as shown in Figure 8. In the **Active** column, select the first four check boxes.
10. In the **Variable Name** column, enter the first four names as RC (raw count), DC (difference count), BL (baseline), and Status (button ON/OFF status).
11. In the **Type** column, select **int** for RC, DC, and BL. Select **byt**e for Status and click **OK.**

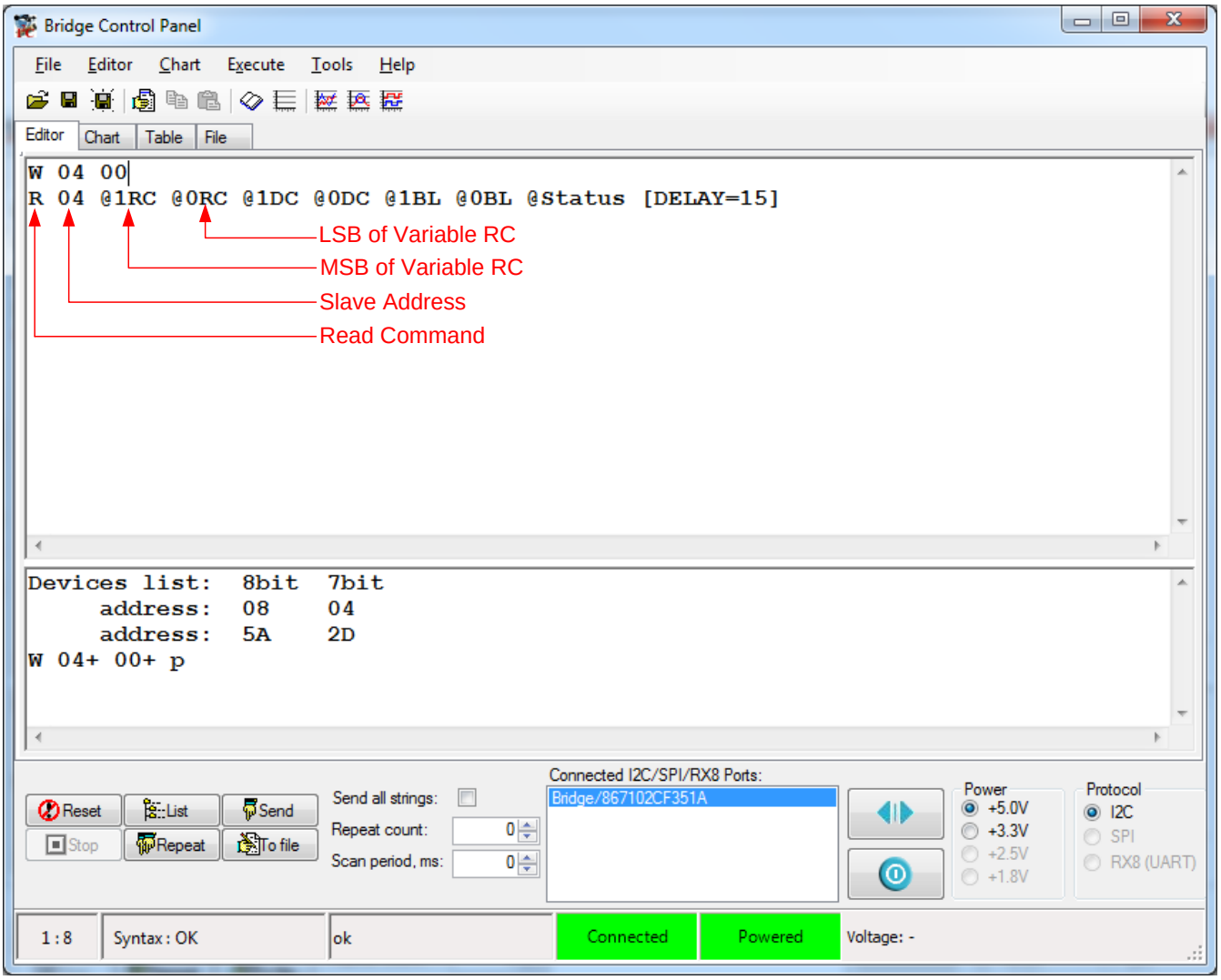
Figure 8. Variable Settings Window



12. In the BCP **Editor** window, press [**Ctrl**] [**Enter**] in the line “W 04 00” to move the cursor to the next line.
13. Write the following command in the command window, as shown in Figure 9:

    R 04 @1RC @0RC @1DC @0DC @1BL @0BL @Status [DELAY=15]

Figure 9. Read Command Execution



Note: The [DELAY=15] command provides a delay of 15 ms before the next I²C transaction occurs when the **Repeat** button is clicked. This delay provides sufficient time for the I²C buffer to reinitialize the read pointer to the beginning of the buffer after the previous read is completed. If a delay is not inserted, then the I²C master will read the values as zeros until the pointer is reinitialized.

14. Click the **Chart** tab and then click the **Repeat** button. Now you can see the CapSense data.
15. To monitor only raw counts, deselect **DC**, **BL**, and **Status**, as shown in Figure 10. The raw counts when the CapSense button sensor is touched is show in Figure 11.

Figure 10. Raw Counts Chart When CapSense Button Sensor is Not Touched

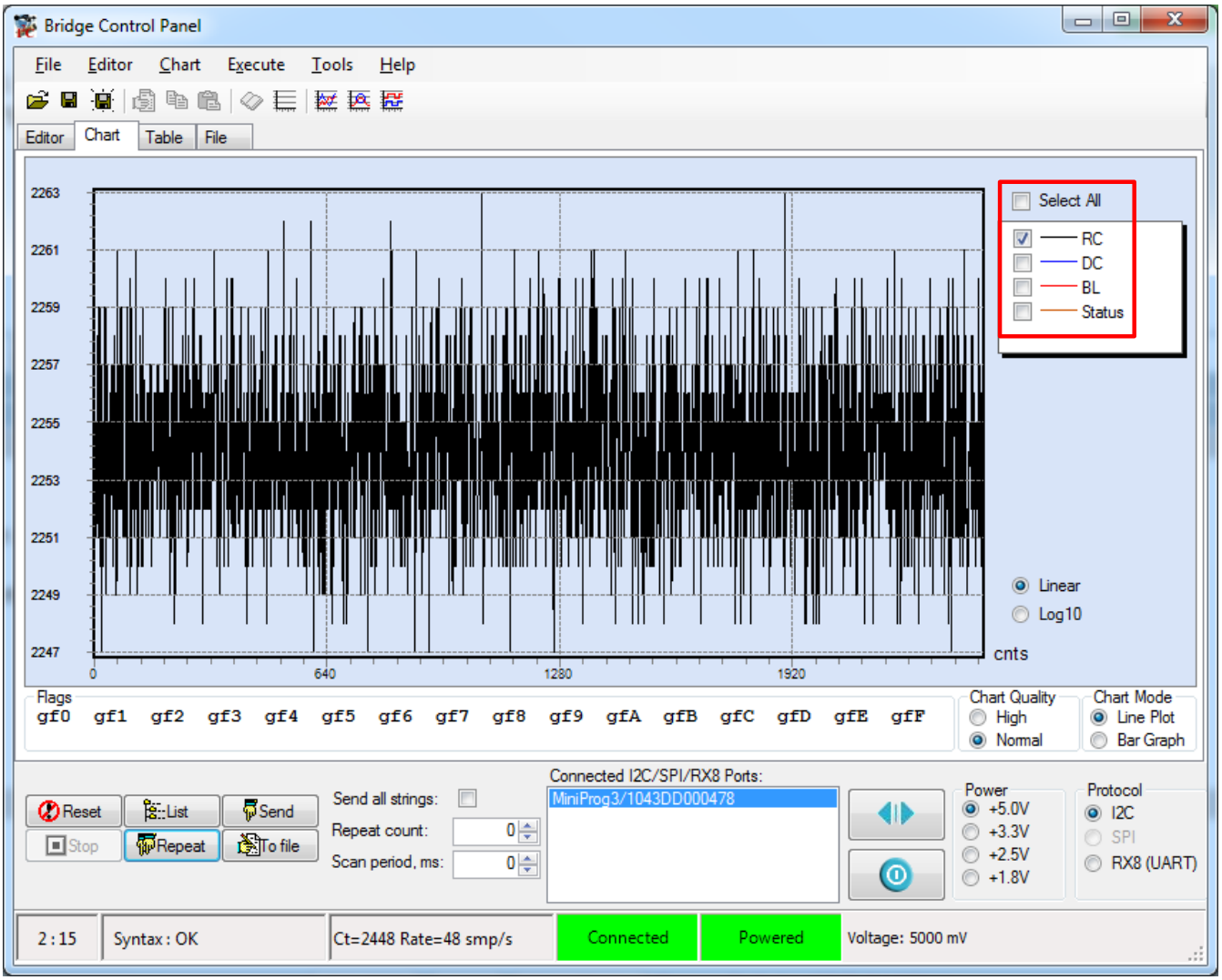
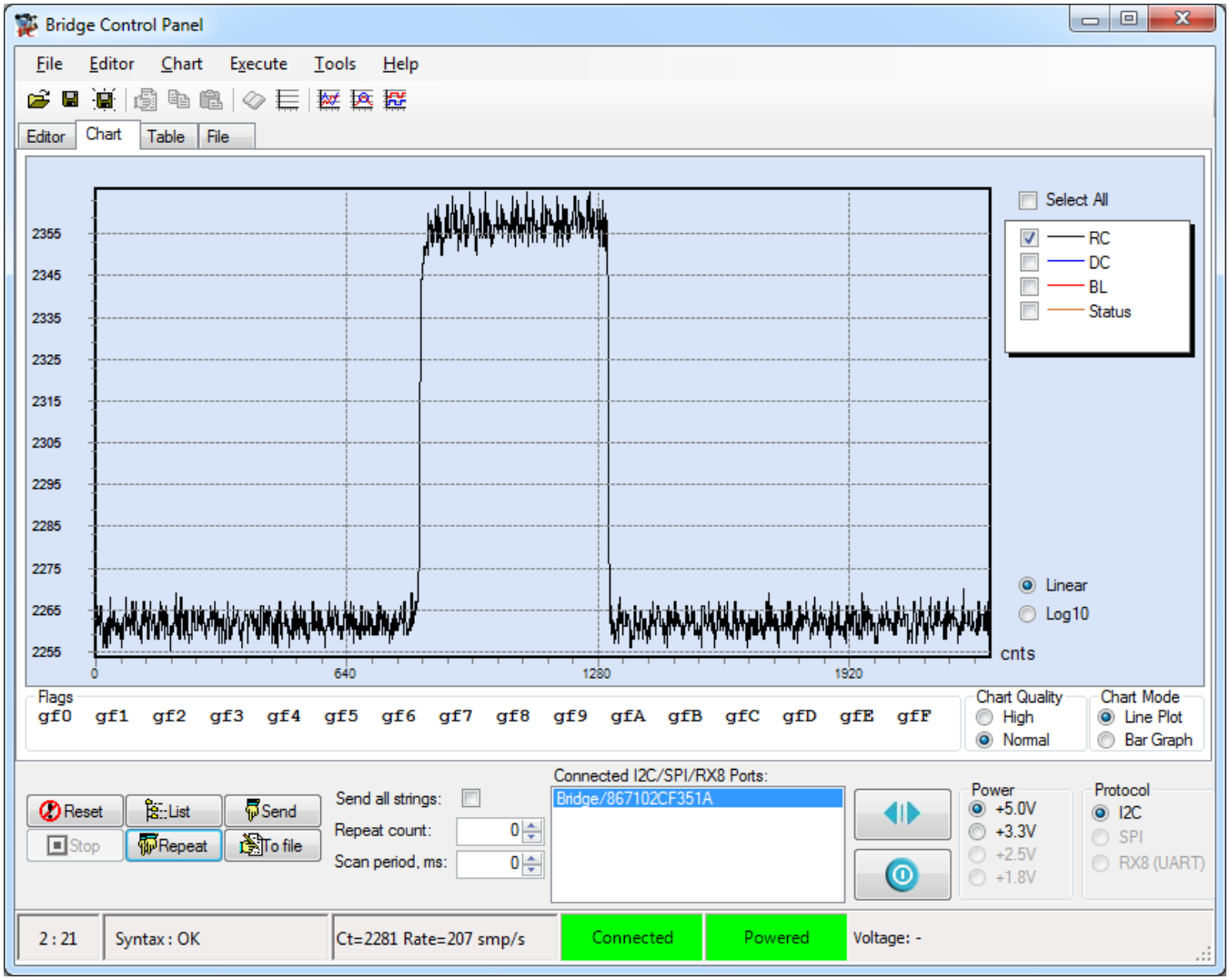


Figure 11. Raw Counts Chart when CapSense Button Sensor is Touched



Related Documents

Table 5 lists the relevant application notes, design guides, code examples, UM datasheets, and device and development kit documentation.

Table 5. Related Documents

Application Notes		
AN75320	Getting Started with PSoC 1	Learn about the PSoC 1 architecture and the development tools required to design an embedded system with the PSoC 1 device family.
AN50987	Getting Started with I2C in PSoC 1	Learn how I^2C communication works and how to implement I^2C communication in PSoC 1.
AN2397	PSoC 1 and CapSense Controllers – CapSense Data Monitoring Tools	Learn how to use I^2C communication to tune and debug a CapSense system.
Design Guides		

AN64846 – Getting Started with CapSense	This guide is an ideal starting point if you are new to capacitive touch sensing (CapSense). It is also useful for learning key design considerations and layout best practices to ensure design success.
CY8C21x34/B – CapSense Design Guide	This document provides design guidance for building CapSense applications with the CY8C21x34 family of devices.
Code Examples	
CapSense Controller Code Examples	This document shows how to measure sensor parasitic capacitance from a CapSense controller, read CapSense sensor data, and tune the CapSense parameters.
PSoC Designer User Module Datasheets	
CSD	The CSD UM controls the CapSense CSD block and detects changes in capacitance in applications such as touch sense buttons, sliders, touchpad, and proximity detection.
I2CHW	The I^2C Hardware UM implements an I^2C device in firmware. The I2CHW UM supports the standard mode with speeds up to 400 kbps.
Device Documentation	
CY8C21x34/B Datasheet	
CY8C21x34/B Technical Reference Manual	
Development Kit Documentation	
CY3280-BK1 – Universal CapSense Controller Basic Kit 1 User Guide	

Document History

Document Title: CE54365 - CSD with I2CHW Slave on CY8C21x34/B

Document Number: 001-54365

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2727566	ARVM	07/14/2009	New example project
*A	2973490	ARVM	07/20/2010	Updated the software version to "PD5.0 SP6 Build 1127". Added "Related Hardware" and "Author" fields on page 1, below the title.
*B	3124139	ARVM	12/30/2010	Updated the project to use 3280-21x34 UCC and SLM boards.
*C	3319209	UDYG/ARVM	07/19/2011	Updated the software version to “PD5.1 SP2 Build 2306”. Renamed Example projects to Code Examples in the title and document.
*D	3689951	SSHH	07/23/2012	No change. Completing sunset review.
*E	4871331	DCHE	08/04/2015	Update code example to latest template. Updated all figures in the document to match latest tool snapshot. Updated project to PSoC Designer 5.4 SP1.
*F	6291565	DIMA	08/27/2018	Completing sunset review.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturers' 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 | PSoC 6 MCU

Cypress Developer Community

Community Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

PSoC Creator is a trademark of Cypress Semiconductor Corp.

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

© Cypress Semiconductor Corporation, 2009-2018. 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. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. 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.