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.

CE54362 – CSA with I2CHW Slave on CY8C20x34

Objective

CE54362 describes how to scan five CapSense® buttons regularly and send the data to the master using I²C protocol.

Overview

This code example incorporates CapSense Successive Approximation (CSA) module and I²CHW module to send the CapSense data to the I²C master. The CapSense module scans all the buttons continuously and stores the raw count, difference count, and baseline details in a structure defined by MyI²CRegs. This structure is used by the I²CHW module to send the data to master when required.

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 to help you to quickly and effectively integrate the device into your design. For a comprehensive list of resources, see the knowledge base article 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. In addition, PSoC Designer offers a device selection tool at the time of creating a new project.
- **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:
  - AN64846 – Getting Started With CapSense
  - CY8C20x34 CapSense® Design Guide.
  - AN2397 – CapSense® Data Viewing Tools
  - CapSense® Successive Approximation Datasheet
- **Technical Reference Manual (TRM):** PSoC® CY8C20x24, CY8C20x34 Family Technical Reference Manual
- **Development Kits:**
  - CY3280-20x34 Universal CapSense Controller Kit features a predefined control circuitry and plug-in hardware to make prototyping and debugging easy. Programming and I2C-to-USB Bridge hardware are included for tuning and data acquisition.
  - CY3280-SLM Linear Slider Module Kit consists of five CapSense buttons, one linear slider (with ten sensors) and five LEDs. This module connects to any CY3280 Universal CapSense Controller Board, including CY3280-20x34 kit.
  - CY3280-BBM Universal CapSense Prototyping Module Kit provides access to every signal routed to the 44-pin connector on the attached controller board including CY3280-20x34 kit.
- **Programming:** PSoC supports a number of different programming modes and tools. For more information, see the General Programming page.

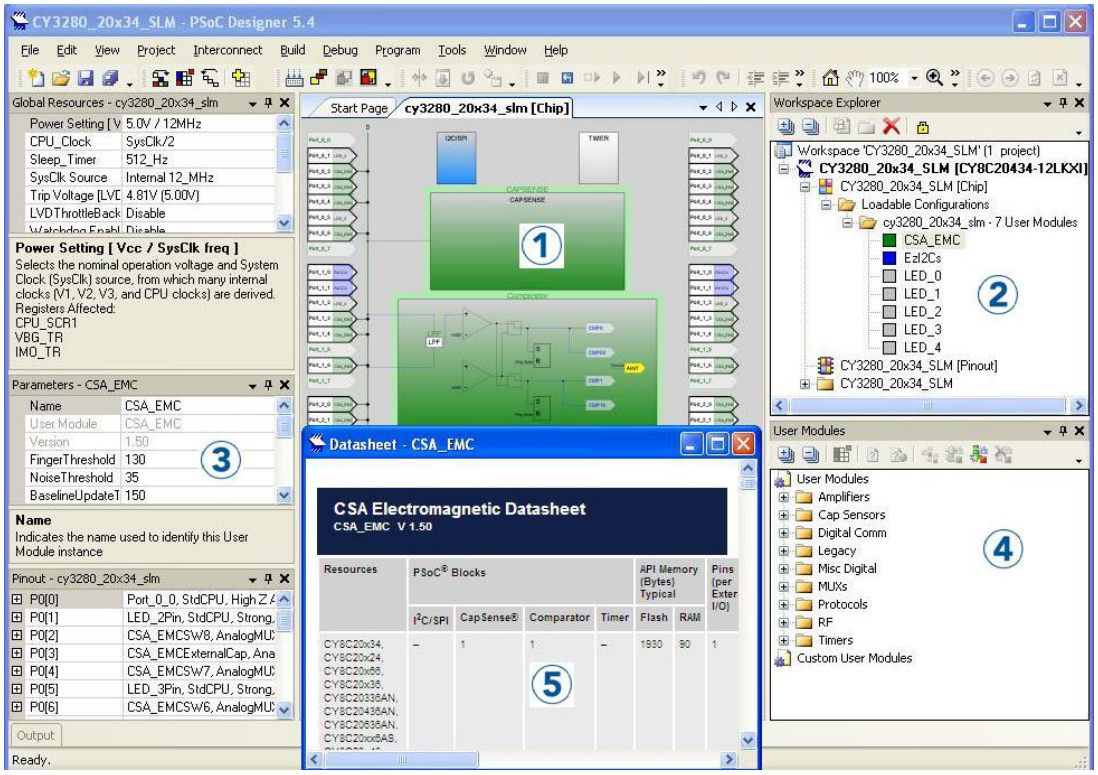
PSoC Designer

PSoC Designer is a free Windows-based Integrated Design Environment (IDE). It enables concurrent hardware and firmware design of systems based on CapSense (see PSoC Designer Features). With PSoC Designer, you can:

1. Drag and drop User Modules to build your hardware system design in the main design workspace
2. Codesign your application firmware with the PSoC hardware, using the PSoC Designer IDE C compiler
3. Configure User Module
4. Explore the library of user modules
5. Review user module datasheets

PSoC Designer Features

Figure 1. PSoC Designer Features



Requirements

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

Programming Language: C (ImageCraft STD and PRO compilers)

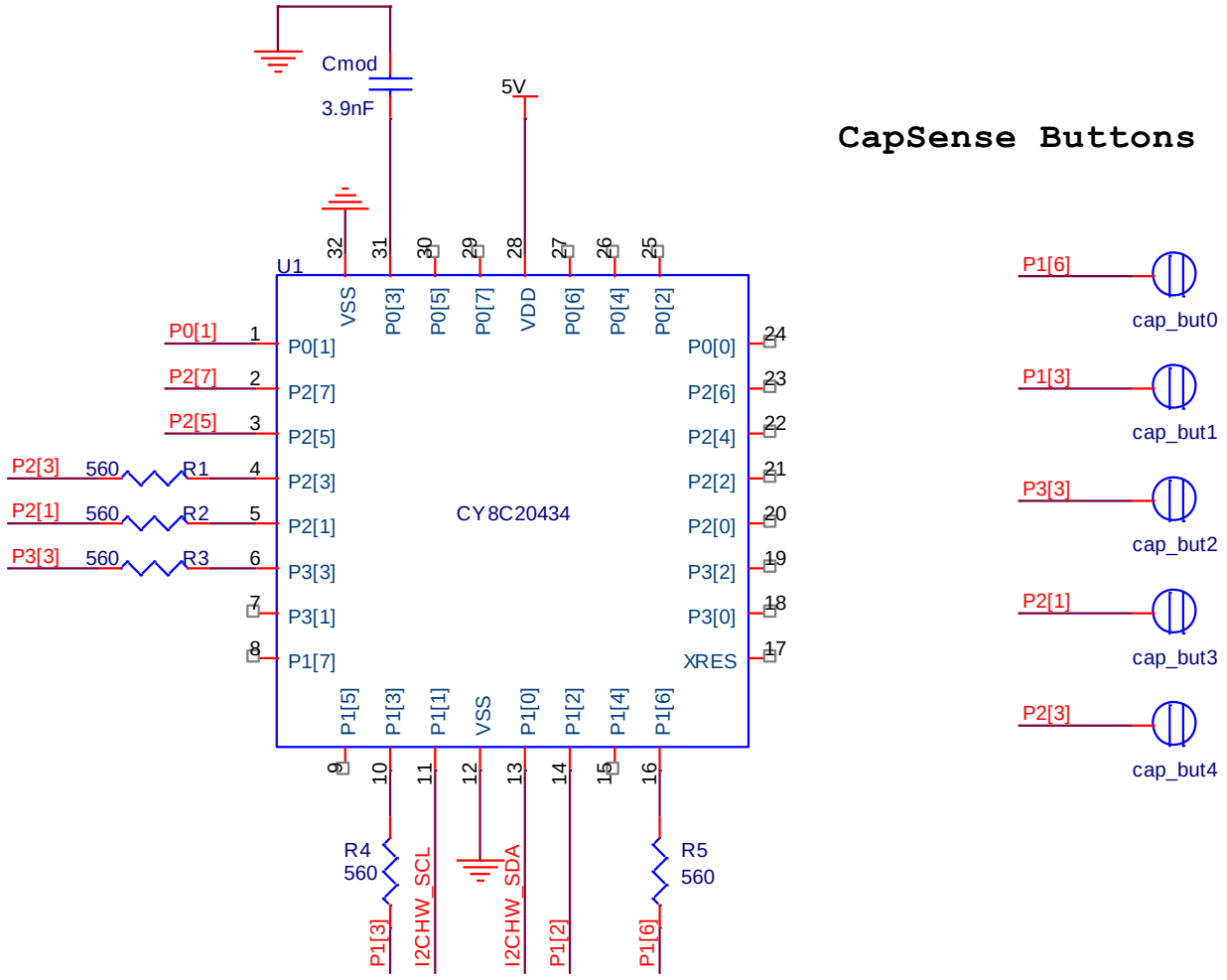
Associated Parts: CY8C20x34 device series

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

Design

The schematic diagram for the code example follows.

Figure 2. Schematic Diagram



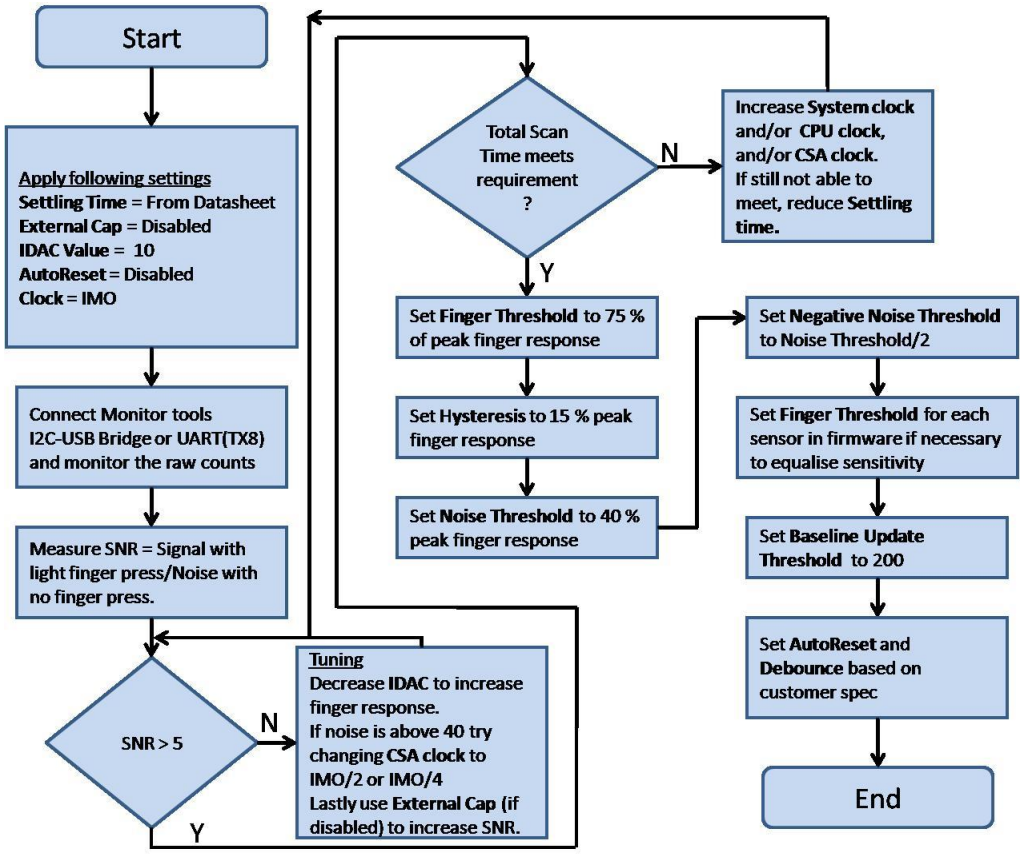
Design Considerations

Refer to the CSA Calibration section to tune the CSA parameters to achieve an SNR > 5:1. For the detailed tuning procedure, refer to the CY8C20x34 CapSense® Design Guide.

CSA Calibration

For optimum performance, CSA parameters are tuned with the actual CapSense hardware and overlay. The following flowchart shows the steps to be performed for calibrating CSA.

Figure 3. CSA Calibration Flowchart



1. Start with the default settings of the CSA user module.
2. Using the CY3240-I2USB bridge, MiniProg3, or UART and the actual hardware and overlay, capture the raw counts, baseline, and difference counts for the sensors.
3. **Coarse Tuning**. Check whether the signal-to-noise ratio (SNR) is greater than five. If SNR is less than five, increase SNR by following recommended PCB guidelines, decreasing the IDAC value, changing the CSA clock, and using an external capacitor. For PCB guidelines, refer to Section 3.7 of the *Getting Started with CapSense* document. For details about SNR and how to measure SNR, refer to Section 4.1.1 of *CY8C20xx6A/H CapSense® Design Guide*.
4. Check whether the total scan time for all the sensors meets requirements. If it does not, increase the frequency of the system clock, CPU clock, and/or CSA clock. Because these parameters also affect SNR, go back to Step 3. With a couple of passes, arrive at the optimum IDAC and CSA clock parameters that produce the best SNR and the desired scan time.
5. Capture the difference counts when the button is activated. Set the finger threshold parameter to 75 percent of the peak finger response.
6. Set the hysteresis parameter to 15 percent of the peak finger response.
7. Set the noise threshold to 40 percent of the peak finger response.
8. Set the negative noise threshold to half the noise threshold.
9. Set finger thresholds for individual sensors if necessary. This is done by writing to the `CSA_baBtnFThreshold` array in firmware.
10. Set the baseline update threshold according to requirements. The frequency with which the baseline is updated must be determined on a code example-to-code example basis. The baseline should be a slow-moving reference, which helps to reduce the effects of noise and temperature on the capacitive sensor.

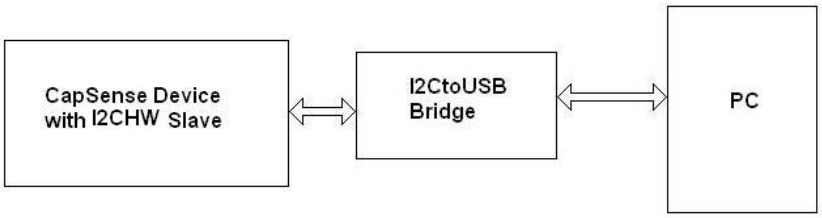
- **Fast update baseline rates:** This can create problems if a user moves their finger slowly to the button. This is called ‘baselining out the finger’.
- **Slow update baseline rates**: This can leave the buttons vulnerable to temperature fluctuations and potentially lead to ‘button lock’.

11. Set AutoReset and Debounce parameters as required. Refer to the CSA User Module datasheet for details of these parameters.
12. For any other parameters refer to the User Module datasheet.

Hardware Setup

Figure 4 shows the hardware setup required for this code example.

Figure 4. Hardware Setup



The CY3280–20x34 Universal CapSense Controller board along with the CY3280–SLM Linear Slider Module is suitable for this code example. Cmod is connected to P0[3]. CY3240-I2USB bridge or MiniProg3 can be used as I^2C master to get the CapSense data from the slave device. Because P1[0] and P1[1] are used as I^2C lines, I^2CtoUSB bridge can be connected to the ISSP header itself. The USB-I^2C Bridge software tool can be used to monitor the CapSense data. A 560-Ω resistor is connected in series with each CapSense button to reduce RF interference.

The pin assignment for CapSense buttons used in this code example is as follows:

Button 0 – P1[6]

Button 1 – P1[3]

Button 2 – P3[3]

Button 3 – P2[1]

Button 4 – P2[3]

Software Setup

The code example works on CY3280-20x34 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 flowchart in Figure 3. To reassign the sensor pins, refer to the “Wizard” section in the CSA_EMC UM datasheet.

User Module List and Placement

The following table lists user modules used in this code example and the hardware resources occupied by each user module.

User Module	Placement
CSA	CapSense and comparator
I^2CHW	I^2C/SPI block

User Module Parameter Settings

The following tables show the user module parameter settings for each of the user modules used in the code example.

CSA		
Parameter	**Value**	**Comments**
Finger Threshold	100	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 this, then it is treated as noise; the Baseline Update Algorithm takes care of this by putting it into Update Bucket.
Baseline Update Threshold	100	As the noise increases, the update bucket is filled and every time it crosses this threshold, baseline is incremented by ‘1’ and algorithm continues.
Settling Time	80	The Settling Time parameter controls the duration of the capacitance-to-voltage conversion phase. The parameter controls a software delay that enables the voltage on the Cmod and Cbus capacitance to stabilize. This parameter is dependent on the clock setting.
IDAC Setting	50	This parameter defines the amount of current that the source is going to pump in the second phase of CSA, i.e., during the Cmod ramping from Vstart to Vref.
Hysteresis	10	This takes care of false ON and OFF situations whenever a button is pressed. Set it as equal to the noise threshold.
Debounce	3	If the difference count is more than the finger threshold for less than 'Debounce' number of samples, it is not taken as a button press.
Negative Noise Threshold	20	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 resets to a new raw count.
Sensors Autoreset	Disabled	When the parameter is disabled, the baseline is updated only when the raw count and baseline difference are below the noise threshold parameter.
High Level API	Enabled	Enabling this parameter includes the high-level APIs provided by the User Module. Disabling this parameter saves RAM and ROM by excluding high-level APIs.
Clock	IMO	Normally, this parameter is left at the default IMO setting. Setting a larger divider of IMO increases the effective resistance of the sensor, compensating for the high capacitance.
ExternalCap	P0[3]	Cmod is connected to the specified pin.

Note

The parameters for CSA given in the table

User Module Parameter Settings are set to work without an overlay on CapSense buttons. If you have an overlay on CapSense buttons in the board, use the flowchart in CSA Calibration to set these CSA parameters.

I²CHW		
Parameter	**Value**	**Comments**
Slave Address	10	This parameter decides the address that is assigned to the slave. Can be any value from 0 to 127 (decimal).
Read_Buffer_Types	RAM ONLY	Only RAM data buffer used.
Communication_Service_Type	Interrupt	See Notes.
I²C Clock	100 k Standard	It decides the maximum clock speed that the slave can operate at.
I²C Pin	P1[0]-P1[1]	This indicates which pins are going to be used as SDA and SCL lines of I²C.

Notes

When the Read_Buffer_Types is set to RAM ONLY, only RAM buffers may be transmitted over I²C. If data from flash buffer is to be read and transmitted, the read buffer type should be set to RAM OR FLASH.

In interrupt-based Communication_Service_Type, the data is moved in and out of the buffer quickly in the background using an ISR.

The I²C clock is dependent on SysClk. The I²C clock setting in the User Module is based on SysClk of 24 MHz. In devices that support slower Sysclk, the I²C clock is reduced by the same proportion. For example, if the I²C clock is set to 400 kHz and SysClk is set to 6 MHz, the actual I²C clock is only 100 kHz.

Global Resources

Important Global Resources		
Parameter	**Value**	**Comments**
Power Setting [Vcc/SysClk frequency]	5.0 V/12 MHz	Selects 5-V operation and 12-MHz SysClk
CPU_Clock	SysClk/1	Selects 12 MHz as CPU clock.

Note

Other parameters are left at their default values.

Operation

On reset, all 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 (sI²CRegs) is defined to store the button number, raw count, difference count, baseline, and status of the CapSense button.

Global interrupt is enabled and the CSA User Module is started, finger thresholds for buttons are set, and baselines are initialized.

The I²CHW user module is started. The “sI²CRegs” structure is set as the I²C Read buffer, and the “bButtonNumber” BYTE variable is set as the I²C Write buffer.

To get information about a button, the I²C master must write the button number into the I²C write buffer (bButtonNumber). The I²C master can get that button information by reading the I²C read buffer (sI²CRegs).

In an infinite “while” loop, all CapSense buttons are scanned, and then all baselines are updated; sI²CRegs is updated with the raw count, difference count, baseline, and status of the requested CapSense button.

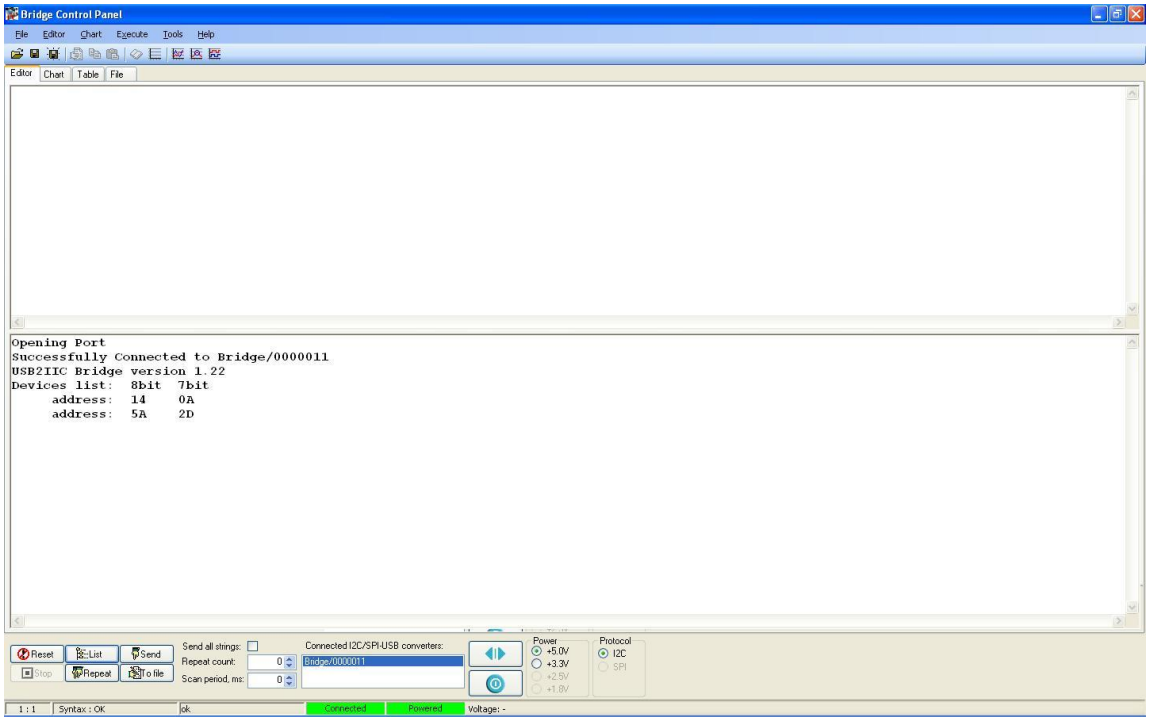
Whenever the I²C master writes (or reads) into the buffer, the Write flag (Read flag) must be reset and the buffer must be set again. This operation is also done in the “while” loop.

Instructions to Use Bridge Control Panel for Monitoring CapSense Data

1. Open the Bridge Control Panel software and connect the CY3240-I2USB bridge or MiniProg3 to a USB port.
2. Click on **Tools** and select Protocol Configuration. Select IIC Speed as 100 kHz.
3. At the right-bottom corner, click the **+5 V** radio button. This powers the target device with 5 V.

4. At the left-bottom corner, click the **List** button. This lists all the slave device addresses. In this case, device address is 10 (0x0A) and it is displayed in the status window, as shown in the following figure.

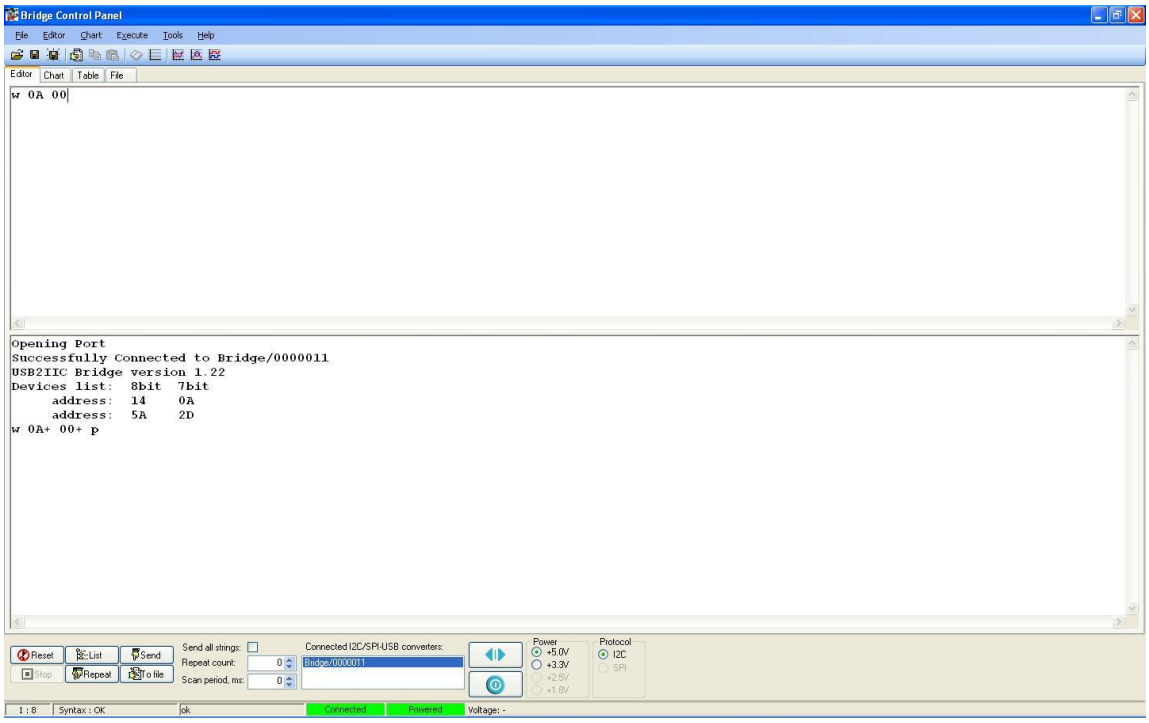
Figure 5. Display of Slave Address in Status Window



5. To monitor the CapSense data of button 0, write the following command in the command window and press **Enter**.

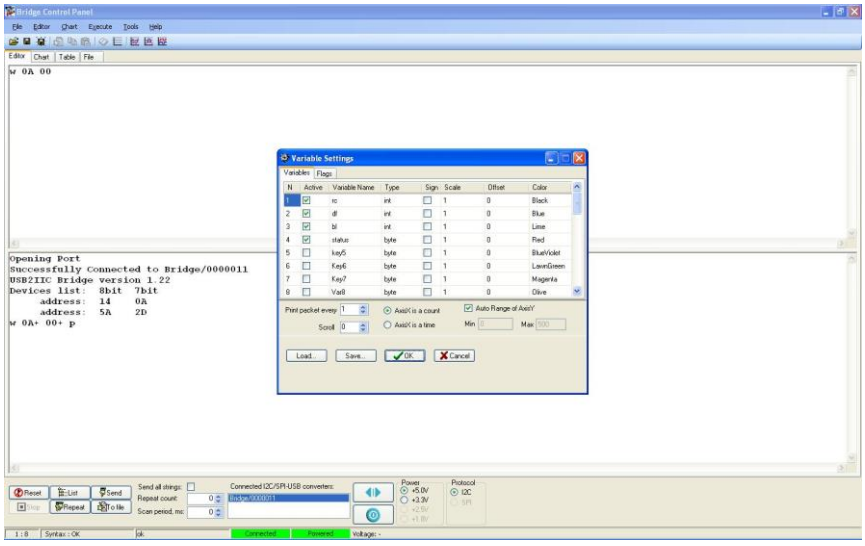
```
W 0A 00
```

Figure 6. Write Command Execution



6. To view CapSense data as a graph, click **Chart** and select **Variable Settings**.
7. The **Variables Settings** window appears. In the **Variables** tab, in the **Active** column check the first four check boxes.
8. In the **Variable Name** column, enter the first four names as rc (RawCounts), df (difference counts), bl (baseline), and status (button ON/OFF status).
9. In the **Type** column, select int for rc, df, and bl. Select byte for status. Click **OK**.

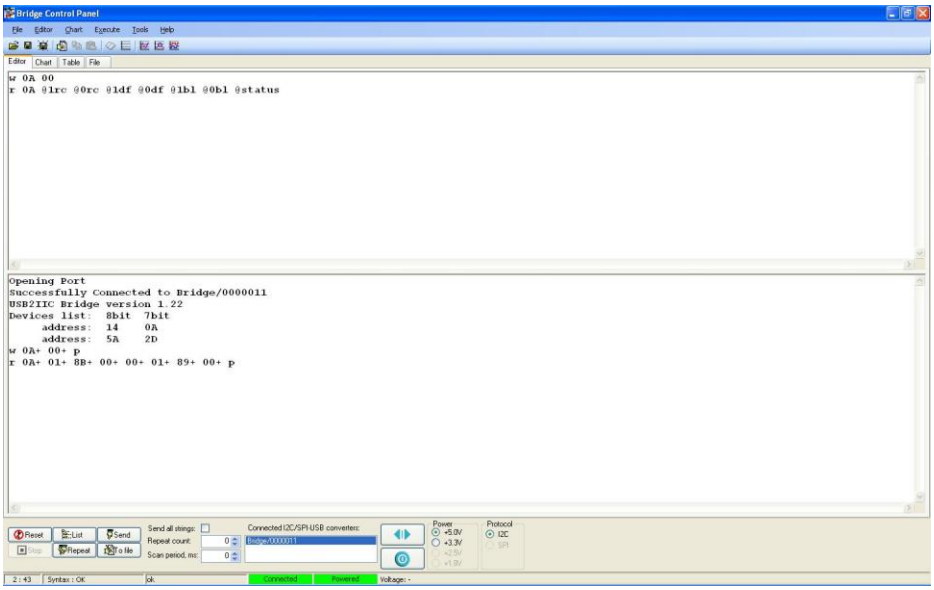
Figure 7. Variable Settings Window



10. Press **Ctrl** + **Enter** to move the cursor to the next line.
11. Write the following command in the command window:

    r 0A @1rc @0rc @1df @0df @1bl @0bl @status

Figure 8. Read Command Execution



12. Click the **Chart** tab and then click the **Repeat** button. Now, CapSense data is displayed.
13. To monitor only raw counts, uncheck df, bl, and status. This is shown in the following figure.

Figure 9. Raw Counts Chart without Button Press

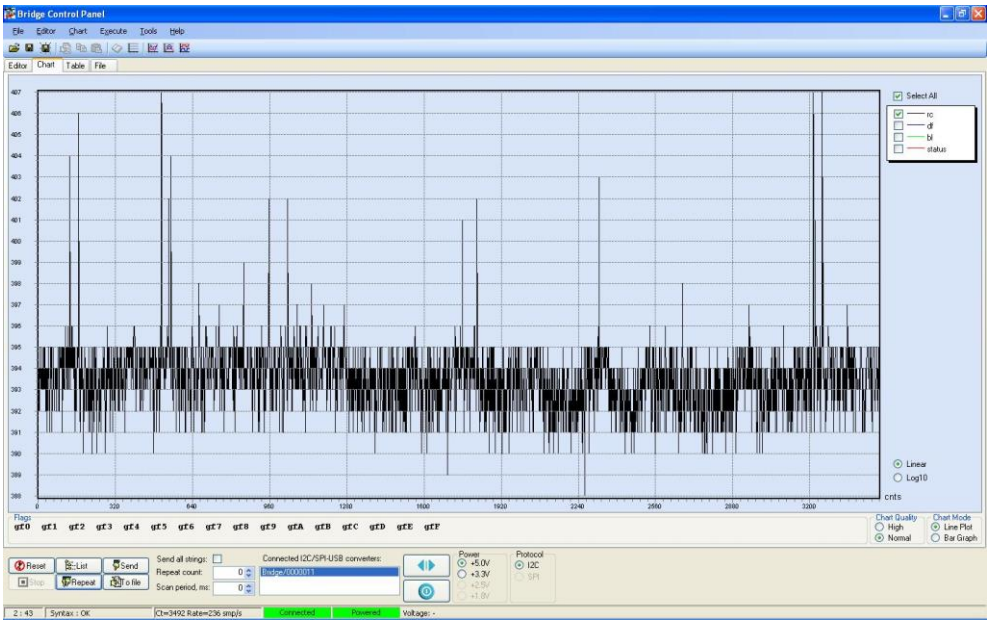
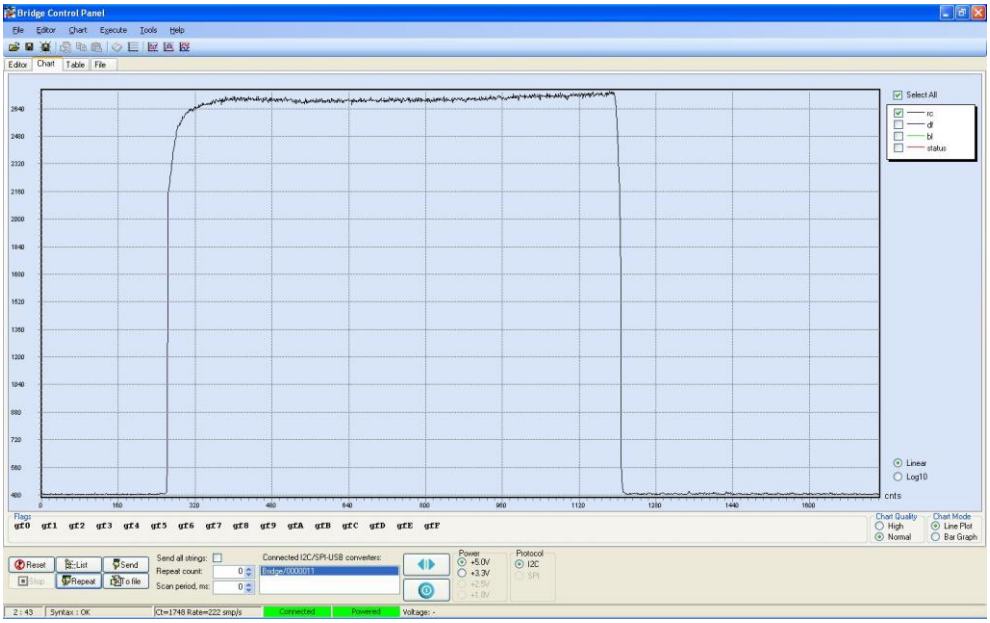


Figure 10. Raw Counts Chart (Button Pressed)



Document History

Document Title: CE54362 - CSA with I²CHW Slave on CY8C20x34

Document Number: 001-54362

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 Software Version to read as "PD5.0 SP6 Build 1127" in page 1. Added "Related Hardware" and "Author" fields in page 1, below the title.
*B	3124139	ARVM	12/30/2010	Updated attached project (to use 3280-20x34 UCC and SLM boards).
*C	3323188	UDYG	07/21/2011	Updated Software Version to read as “PD5.1 SP2 build 2306” in page 1.
*D	4490969	DIMA	09/02/2014	Updated Software Version to read as “PSoC Designer 5.4”. Updated attached project to (PSoC Designer 5.4). Completing Sunset Review.
*E	4740155	SSHH	04/24/2015	Updated Related Hardware to read as “CY3280-20x34 UCC board, CY3280-SLM board, CY3240-I2USB bridge or MiniProg3”. Replaced “I²CtoUSB Bridge” with “CY3240-I2USB bridge or MiniProg3” in all instances across the document. Updated to new template.
*F	4883434	DIMA	08/22/2015	Added “PSoC Resources” section. Updated to new template. Completing Sunset Review.
*G	6322470	SGAN	09/26/2018	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 | PSoC 6 MCU

Cypress Developer Community

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

Technical Support

cypress.com/support

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corporation.

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.