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.

CE54363 - CSA with I2CHW Slave on CY8C20xx6

Objective

This code example demonstrates how to use CapSense Successive Approximation (CSA) User Module (UM) in CY8C20xx6 device to scan CapSense button sensors and send the sensor 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 My I²CRegs. This structure is used by the I²CHW module to send the data to master when required.

This code example can be tested with CY3280-20x66 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:

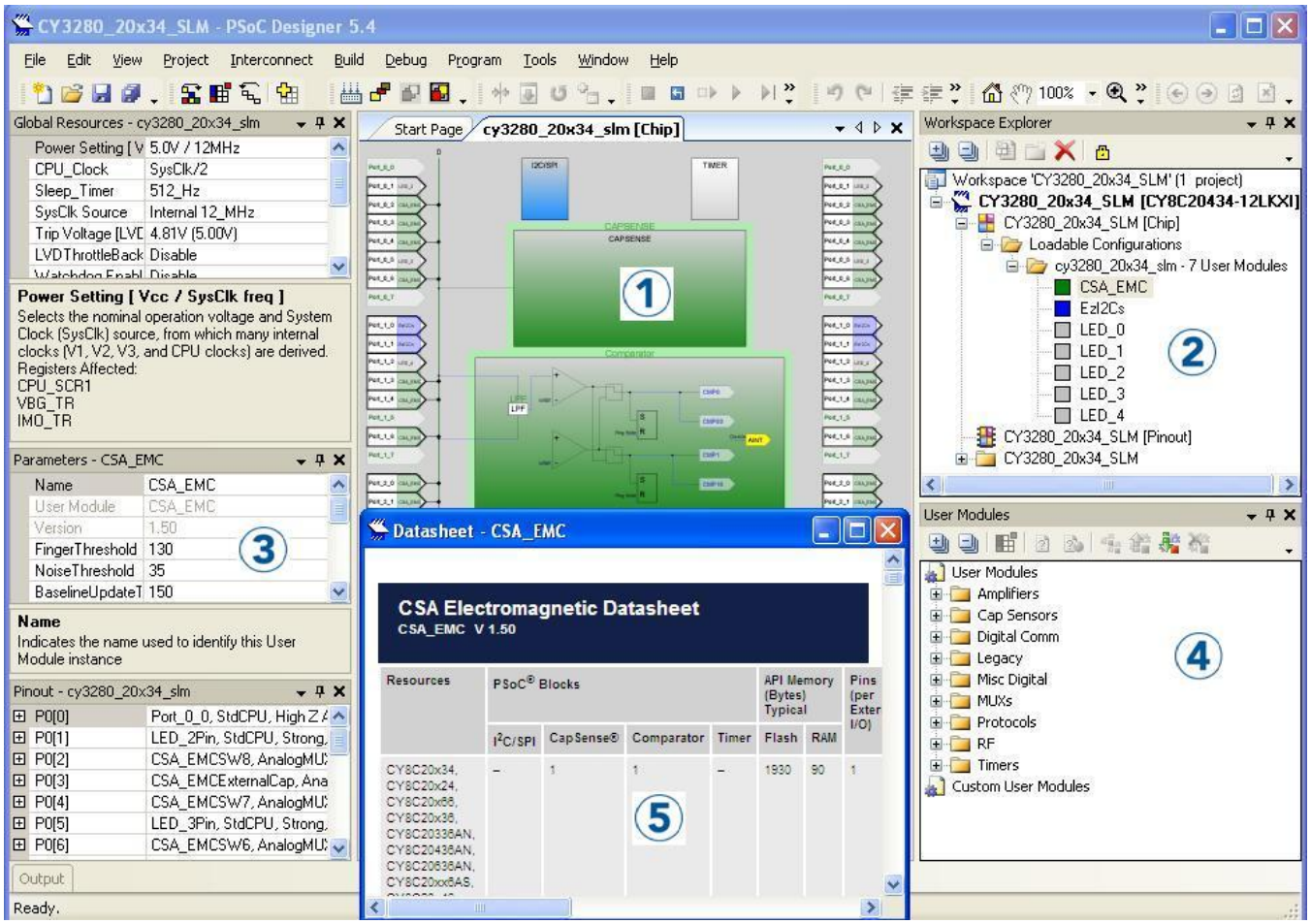
- **Overview:** CapSense Portfolio, CapSense Roadmap
- **Product Selectors:** CapSense, CapSense Plus, CapSense Express, PSoC 3 with CapSense, PSoC 5 with CapSense, PSoC 4. In addition, PSoC Designer offers a device selection tool at the time of creating a new project.
- **Datasheets**: Describe and 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:
  - AN64846 – Getting Started With CapSense.
  - AN65973 - CY8C20xx6/A/H/AS 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 each of the CapSense, PSoC 1, PSoC 3, PSoC 4 and PSoC 5LP device families.
- **Development Kits:**
  - CY3280-20xx6 Universal CapSense Controller Board features a predefined control circuitry and plug-in hardware to make prototyping and debugging easy. Programming and CY3280-I2USB 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 CY3280 Universal CapSense Controller Board, including CY3280-21x34 kit.
  - CY3280-BBM Universal CapSense Prototyping Module Kit consists of eight LEDs as well as eight CapSense sensors organized in a 4x4 matrix format to form 16 physical buttons.
- **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 PSoC 1 and CapSense device; see Figure 1. 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
3. Configure User Module
4. Explore the library of user modules
5. Review user module datasheets

Figure 1. PSoC Designer Features



Requirements

Tool: PSoC Designer 5.4 SP1, Bridge Control Panel

Programming Language: C (Imagecraft and Imagecraft Pro Compilers)

Associated Parts: CY8C20xx6A/H/AS device series

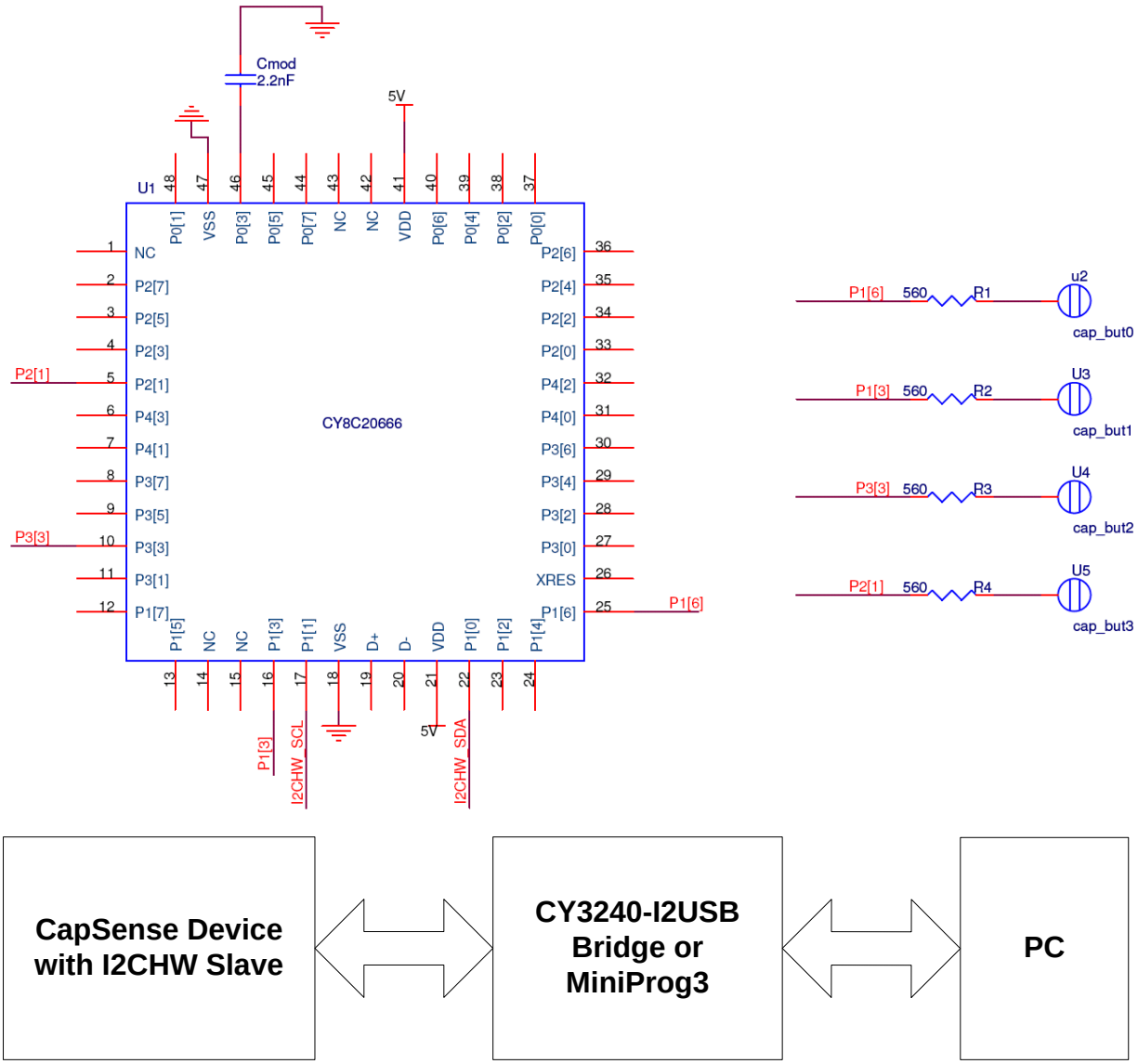
Related Hardware: CY3280-20x66 Universal CapSense Controller (UCC) Board.

Design

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

- CapSense CSA user module to detect finger touches on the CapSense button sensor.
  - The CSA user module is configured to scan four CapSense button sensors
  - All the CSA parameters are manually set to achieve optimum touch sensing performance
- I2CHW user module to send the CapSense button sensor data to I^2C master

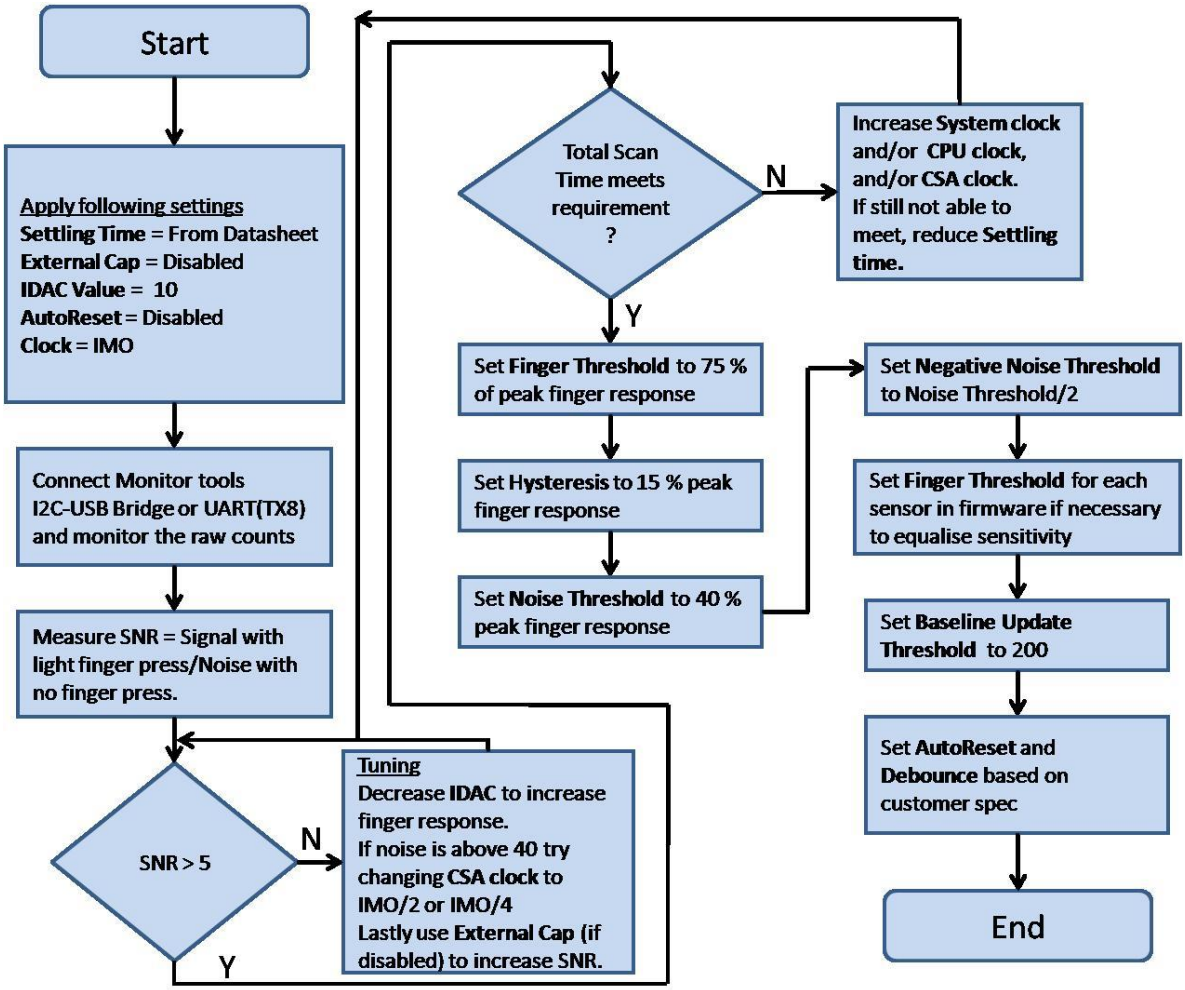
Figure 2. CapSense CSA Design with CY8C20xx6



Design Considerations

For optimum performance, the 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 CY3240-I2USB Bridge or MiniProg3 or UART and the actual hardware and overlay, capture the raw counts, baseline, and difference counts for the sensors.
3. **Coarse Tuning**. Check if signal-to-noise ratio (SNR) is greater than 5. If SNR is less than 5, increase SNR by following recommended PCB guidelines, decreasing the IDAC value, changing CSA clock, and using external capacitor. For PCB guidelines refer to section 3.7 of Getting Started with CapSense. For details about SNR and how to measure SNR, refer to section 4.1.1 of CY8C20xx6A/H CapSense® Design Guide.
4. Check if total scan time for all the sensors meets requirement. If it does not, increase 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 project-to-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 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

For the basic kit board setup, see the corresponding Kit User Guide.

CY3280-20x66 Universal CapSense Controller board along with CY3280-SLM Universal CapSense 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 slave device. Because P1[0] and P1[1] are used as I^2C lines, CY3240-I2USB Bridge or MiniProg3 can be connected to the ISSP header itself. CY3240-I2USB Bridge or MiniProg3 program 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]

Software Setup

The code example works on CY3280-20xx6 kit with CY3280-SLM board without any modification. To test the code example on a custom board, the sensor pins should be reassigned and the CSA parameters should be retuned as per flowchart in **Error! Reference source not found.**. To reassign the sensor pins, refer to "CSA Wizard" section in CapSense® CSA user module datasheet.

User Modules

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

Table 1. List of PSoC Designer User Modules

User Module	Placement
CSA	CapSense and comparator
I^2CHW (normal slave operation)	I^2C/SPI block

Parameter Settings

Table 2 to Table 4 show the parameter settings for each of the user modules used in the code example.

Table 2. CSA User Module Settings

CSA		
Parameter	**Value**	**Comments**
Finger Threshold	100	After the difference count crosses 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 and 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 setting controls a software delay that is the voltage on the Cmod and Cbus capacitance to stabilize and is dependent on the clock setting.
IDAC Setting	20	This parameter declares the amount of current that the source is going to pump in the second phase of CSA that is during the Cmod trying to reach Vref from Vstart.
ExternalCap	P0[3]	Cmod is connected to the specified pin.
Hysteresis	10	It takes care of 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 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 baseline and the difference count is more than this threshold, the baseline does not update.
Low BaseLine Reset	50	If the raw count is below baseline and the difference count is more than negative noise threshold for number of samples given by this parameter, the baseline resets to the new raw count.
Sensors Autoreset	Disabled	When the parameter is set to disabled, the baseline is updated only when raw count and baseline difference is below the noise threshold parameter.
High Level API	Enabled	Setting this parameter to Enabled 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 default IMO setting. Setting a larger divider of IMO increases the effective resistance of the sensor, compensating for the high capacitance.

Table 3. I2CHW User Module Settings

I^2CHW		
Parameter	**Value**	**Comments**
Slave Address	10	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 RAM data buffer used
Communication_Service_Type	Interrupt	See Notes below
I^2C Clock	100k Standard	It decides the maximum clock speed at which the slave can operate.
I^2C Pin	P1[0]-P1[1]	Indicates which pins are going to be used as 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 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, data is moved in and out of buffer quickly in the background using an ISR.
- The I²C clock is dependent on the SysClk. The I²C clock setting in the user module is based on a SysClk of 24 MHz. In devices which 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

Table 4. Global Resources Settings

Important Global Resources		
Parameter	**Value**	**Comments**
IMO Setting	12 MHz	Selects 12 MHz SysClk
CPU_Clock	SysClk/1	Selects 12 MHz as CPU clock.

Note Other parameters are left at their default value.

Operation

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

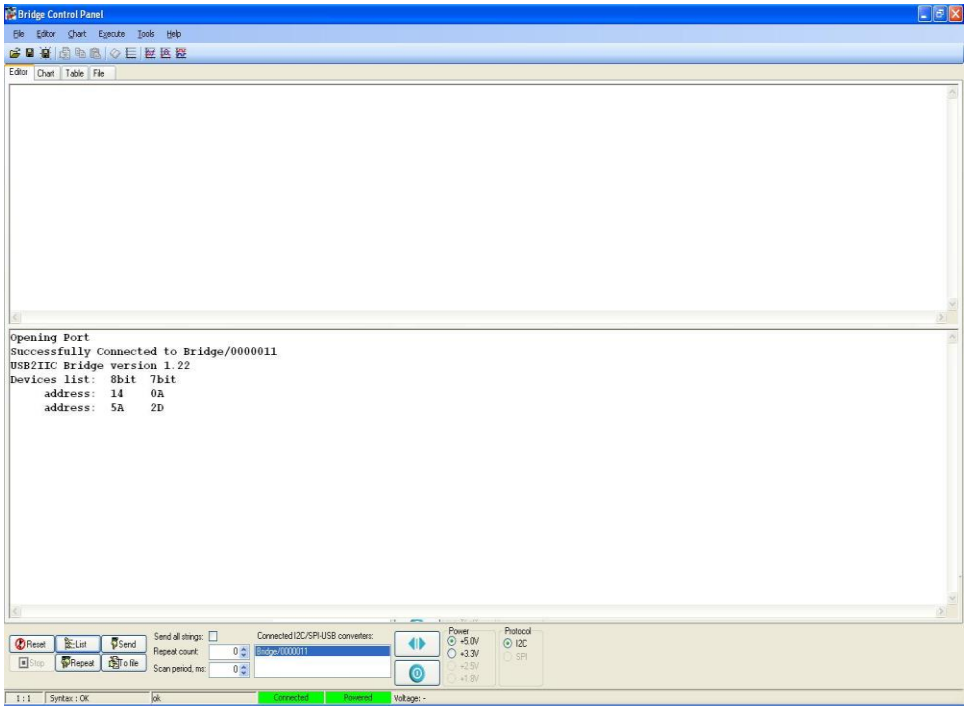
The following operations are performed by the firmware:

- 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.
- I²CHW user module is started. Structure “sI²CRegs” is set as the I²C Read buffer and BYTE variable “bButtonNumber” is set as the I²C write buffer.
- To get information of a button, I²C master must write the button number into the I²C write buffer (bButtonNumber). I²C master can get that button information by reading the I²C read buffer (sI²CRegs).
- In an infinite while loop, all the CapSense buttons are scanned, and then all baselines are updated; and sI²CRegs is updated with the raw count, difference count, baseline, and status of the requested CapSense button.
- Whenever I²C master writes (or reads) into the buffer, the Write flag (Read flag) must be reset and write (Read) buffer must be set again. This operation is also done in the while loop.

Instructions to Use Bridge Control Panel Software for Monitoring CapSense Data

1. Open the Bridge Control Panel Software and connect the CY3240-I2USB Bridge or MiniProg3 to USB port.
2. Click on **Tools** and select **IIC Speed** – 100 K.
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.

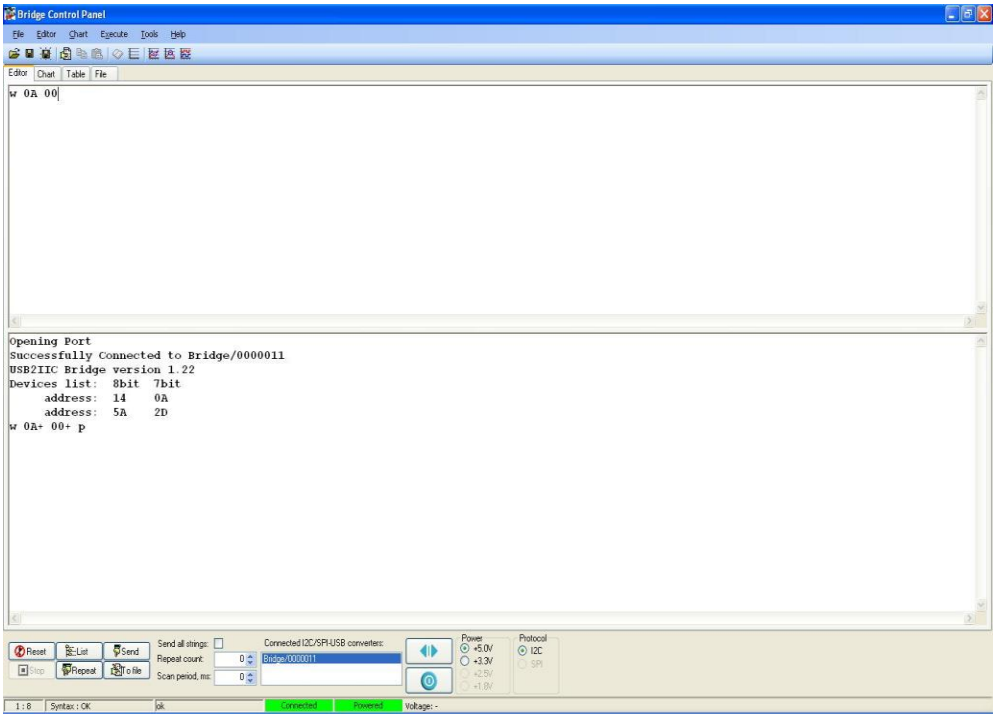
Figure 4. Display of Slave Address in Status Window



5. To monitor the CapSense data of button 0, write the following command in the command window and press <enter>.

```
W 0A 00
```

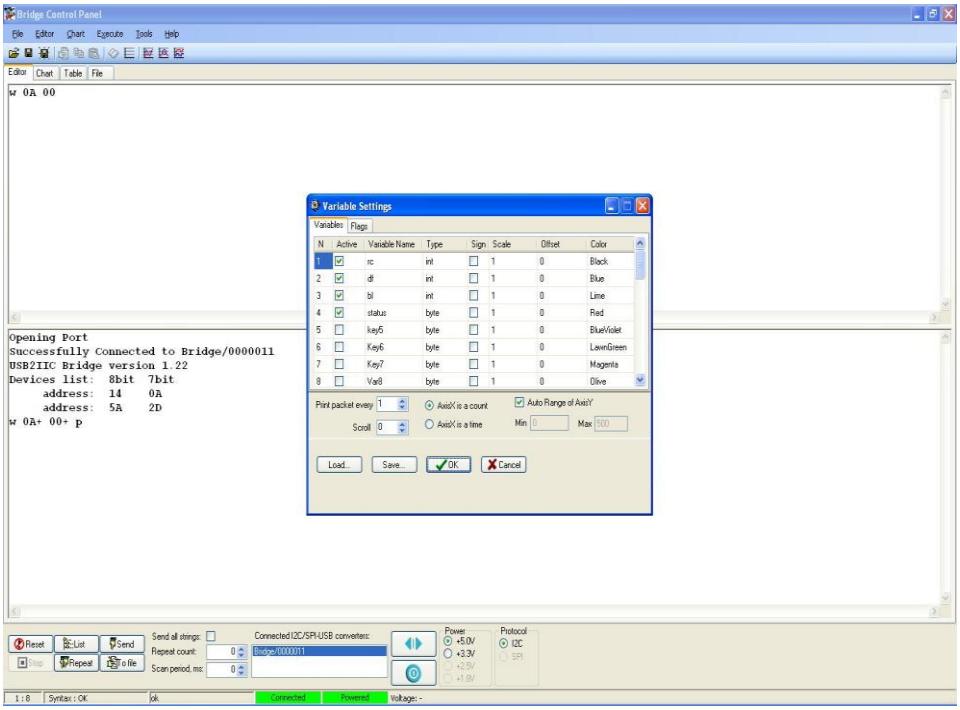
Figure 5. Write Command Execution



6. To view CapSense data as a graph, click **Chart** and select **Variable Settings**.

7. The **Variable 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**.

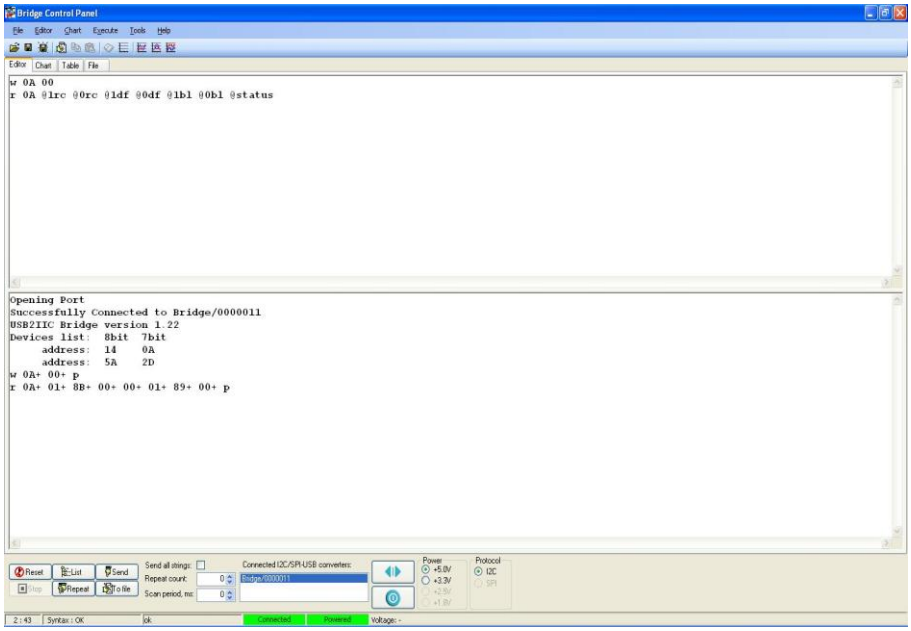
Figure 6. Variable Settings Window



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

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

Figure 7. Read Command Execution



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

Figure 8. Raw Counts Chart without Button Press

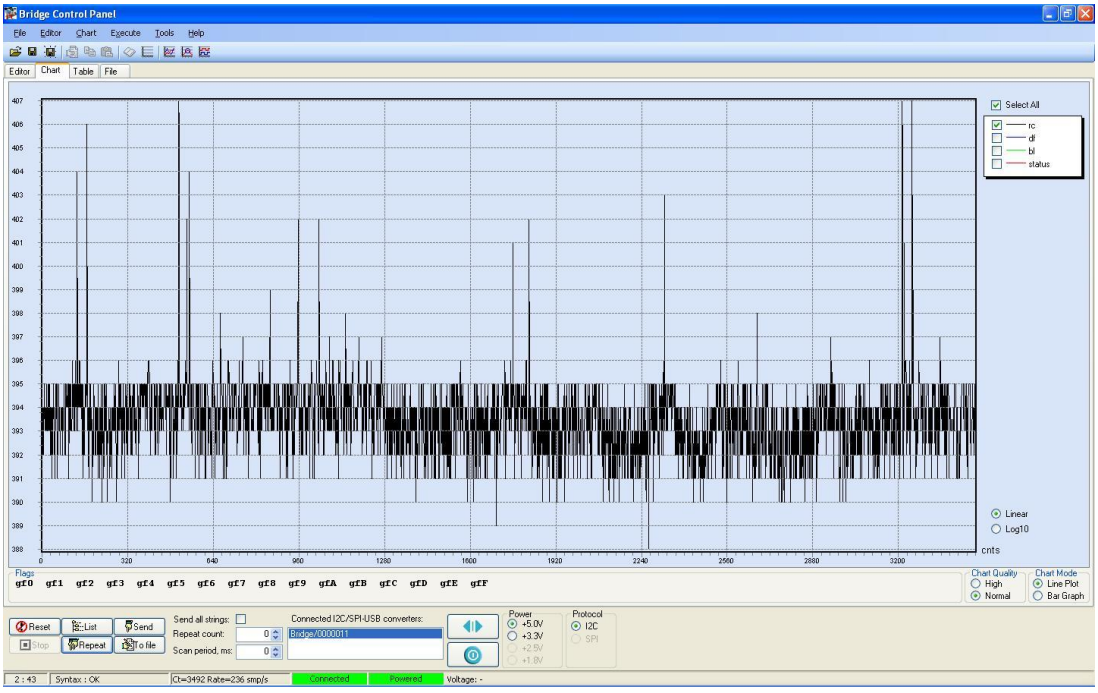
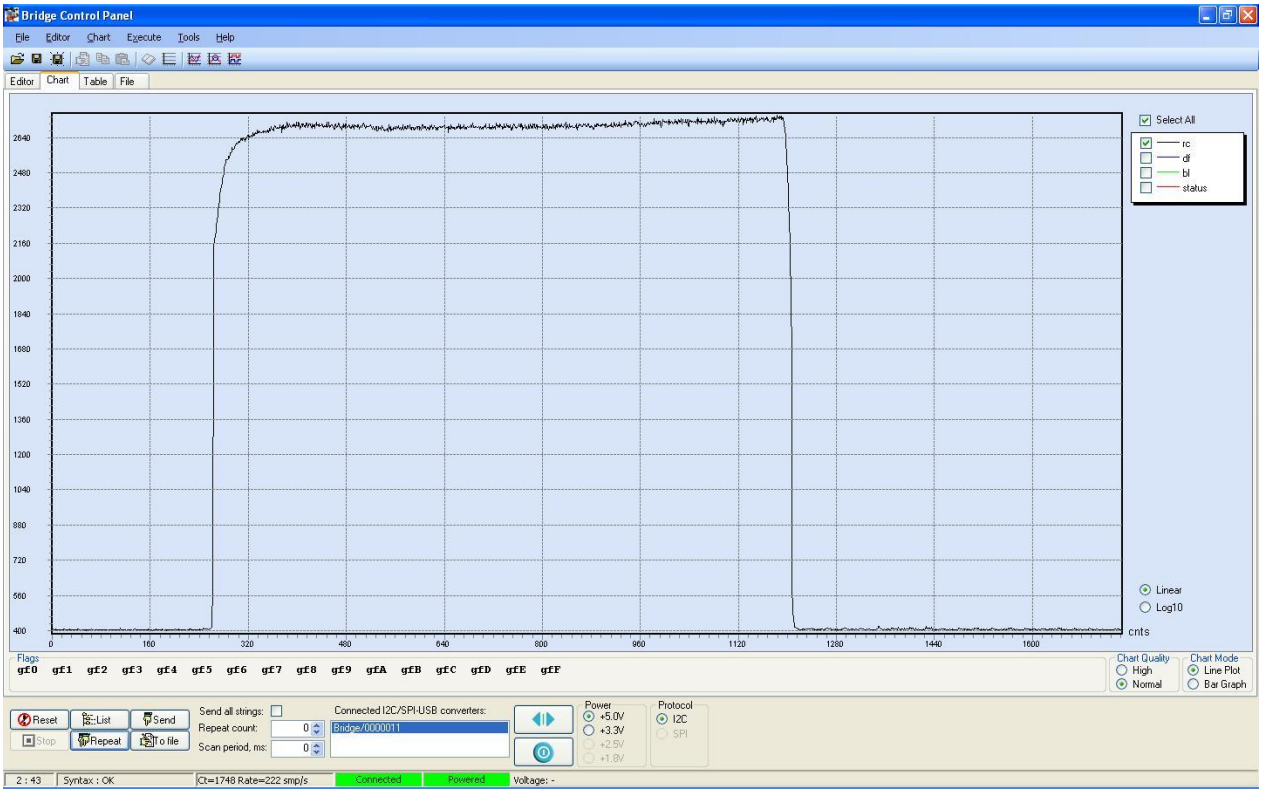


Figure 9. Raw Counts Chart (Button Pressed)



Related Documents

Table 5 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and user module datasheets.

Table 5. Related Documents

Application Notes		
AN75320	Getting Started with PSoC® 1	In this application note, you will learn about the PSoC 1 architecture and the development tools that is required to design an embedded system with PSoC 1 device family.
AN50987	Getting Started with I2C in PSoC® 1	In this application note you will learn how I^2C communication works and how to implement I^2C communication in PSoC 1.
AN2397	PSoC® 1and CapSense® Controllers – CapSense Data Monitoring Tools	In this application note you will learn how to use I2C communication to tune and debug a CapSense system.
Design Guide		
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.
AN65973 - CY8C20xx6/A/H/AS CapSense® Design Guide		This document provides design guidance for building CapSense applications with the CY8C20xx6 family of devices.

Code Examples	
CapSense® Controller Code Examples	This code example 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	
CapSense CSA	Controls CapSense CSA block and detects change in capacitance in applications such as touch sense buttons, sliders, touchpad, and proximity detection.
I2CHW	I^2C Hardware User Module implements an I^2C device in firmware. The I2CHW User Module supports the standard mode with speeds up to 400 kbps.
Device Documentation	
CY8C20xx6A/S Datasheet	
CY8C20xx6A/AS/L Family Technical Reference Manual	
Development Kit(DVK) Documentation	
CY3280-20x66 Universal CapSense Controller Kit	

Document History

Document Title: CE54363 - CSA with I2CHW Slave on CY8C20xx6

Document Number: 001-54363

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2727566	ARVM	07/01/2009	New example project
*A	2973490	ARVM	07/08/2010	Updated the software version to "PD5.0 SP6 Build 1127" Added "Related Hardware" and "Author" fields on page 1, below the title.
*B	3197599	SSHH	03/16/2011	Replaced ‘example project’ with ‘code example’, and updated the title. Changed the default compiler to Image craft from Hi-tech in the PSoC Designer setting.
*C	3319209	UDYG / ARVM	07/19/2011	Updated the software version to “PD5.1 SP2 Build 2306”.
*D	3689951	SSHH	07/23/2012	No technical updates. Completing Sunset Review.
*E	4593734	SLAN	01/13/2015	Replaced “I2CtoUSB bridge”, “USBtoI2C bridge”, “USB-I2C bridge” and “I2C-USB bridge” with “CY3240-I2USB Bridge or MiniProg3” in all instances across the document. Replaced “USB-I^2C tool software” with “Bridge Control Panel Software” in all instances across the document. Updated Instructions to Use Bridge Control Panel Software for Monitoring CapSense Data: Updated Figure 4, Figure 5, Figure 6, Figure 7, Figure 8 and Figure 9.
*F	4874381	SSHH	08/06/2015	Updated to new template
*G	6291547	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.