

CapSense® Proximity Design

1.10

Features

- Sensing elements: 5-segment Linear Slider:

Features

- CapSense® Proximity Sensing
- Pseudo Random Sequence Modulator based LED brightness control
- Software UART interface for viewing CapSense data

General Description

This example is a PSoC Creator starter design. The project demonstrates a CapSense based proximity sensing design to control brightness of a LED. It will help user learn how to design a proximity sensor in their design using a PSoC 4 (4000, 4100, 4200, 4200-BL, 4100M, 4200M, 4200L families) device and see how an approaching hand controls the intensity of a LED. It will employ the CapSense auto tuning ability, SmartSense®, to tune the proximity sensor of any wire/trace length. The project will also help user understand and design a simple Sleep-scan routine using the proximity sensor where the device will enter a periodic scan mode at a configurable rate and sleep once the sensor scan is complete to save power.

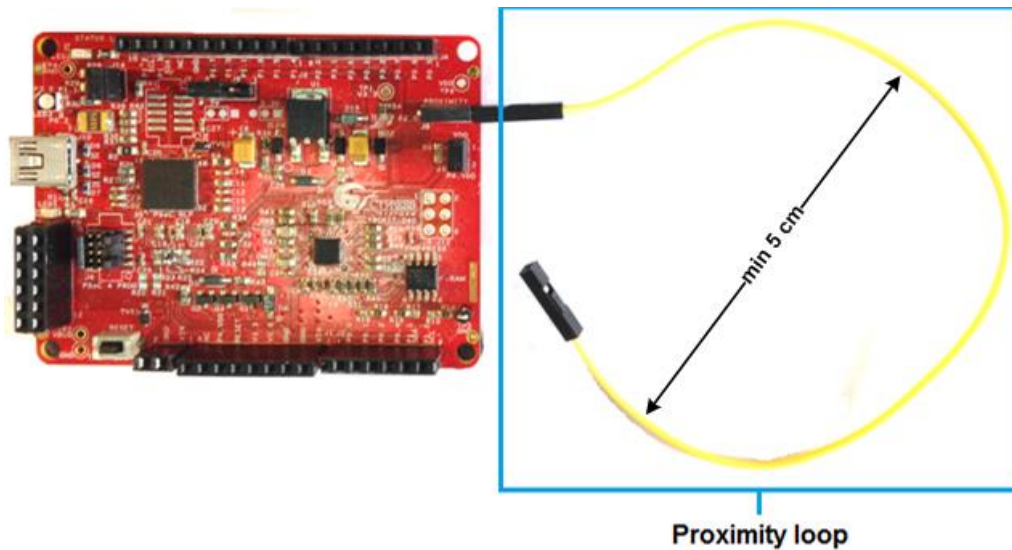
If you are new to CapSense, see [PSoC 4 CapSense Design Guide](#).

Development Kit Configuration

The following configuration instructions provide a guideline to test this design in CY8CKIT-040. Though the below instructions describe a stepwise procedure to be followed while testing this design with the CY8CKIT-040 (PSoC 4000) Kit. This example, however, can be validated on any other PSoC 4000 development platforms. For details, please refer to the [Schematic and Pin Mapping](#) section at the end of the document.

1. Use J1 on the CY8CKIT-040 to select 5.0 V.
2. Plug the Proximity wire loop into J5 on the base board and create a loop as shown in figure.

Figure 1. Proximity demo using CY8CKIT-040 Setup



3. Make sure Jumper J14 and J13 are present.
4. Plug the USB cable into J10.
5. Build the project in creator and program the device.
6. Now move your hand towards the loop and notice the LED on the board starts turning ON and its brightness will increase as the hand approaches the loop.

The project requires configuration settings changes in order to run on the CY8CKIT-042 or CY8CKIT-044 PSoC 4 Pioneer Kits from Cypress Semiconductor. A full description of the CY8CKIT-042 kit, along with more example programs and ordering information, can be found at <http://www.cypress.com/go/cy8ckit-042>.

In order to switch from the CY8CKIT-040 to the CY8CKIT-042/CY8CKIT-042-BLE/ CY8CKIT-044 KITS following steps should be performed:

1. Select the appropriate device with a Device Selector called from the project's context menu:
 - CY8C4245AXI-483 for CY8CKIT-042 KIT
 - CY8C4014LQI-422 for CY8CKIT-040 KIT
 - CY8C4247LQI-BL483 for CY8CKIT-042-BLE
 - CY8C4247AZI-M485 for CY8CKIT-044 PSoC 4 Pioneer Kit
 - CY8C4248BZI-L489 for CY8CKIT-046 PSoC 4 Pioneer Kit
2. Change assignment of the pin components to physical pins. In the Workspace Explorer window, double-click the project's design-wide resource file and assign the pins for Capsense and Pin_LED accordingly to Table 1.

Table 1. Pin assignment of CapSense_CSD_P4_Proximity_Design project

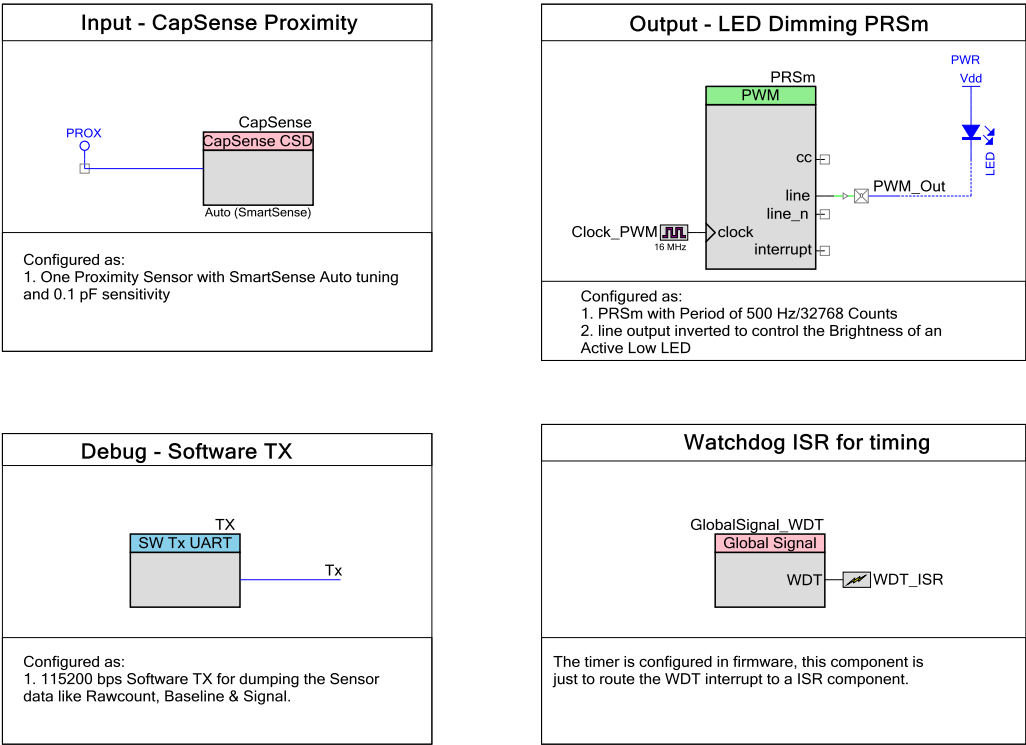
Pin Name	Development Kit				
	CY8CKIT-042	CY8CKIT-042 BLE	CY8CKIT-040	CY8CKIT-044	CY8CKIT-046
\Capsense:Cmod\	P4[2]	P4[0]	P0[4]	P4[2]	P5[0]
\Capsense:Sns\	P1[0]	P2[0]	P2[0]	P3[6]	P8[1]
Pin_LED	P1[6]	P3[6]	P3[2]	P0[6]	P5[3]
\TX:tx\	P4[1] ⁽¹⁾	P1[5]	P3[0]	P7[1]	P3[1]

The PSoC 4 device does not have the direct connection of the tx pin to PSoC 5LP Programmer and Debugger in the CY8CKIT-042 Kit. The TX line from PSoC 5LP is brought to the P12[6] (J9) pin of header J8 (PSoC 5LP GPIO Extension Header). Refer to the "3.3 USB-UART Bridge" section of the CY8CKIT-042 PSoC 4 Pioneer Kit Guide for details.

Project Configuration

This example project uses CapSense component for proximity, TCPWM for LED brightness and SW Tx UART component for data logging over PC. Figure shows the Top Design schematic of this project.

Figure 2. Topdesign Schematic of the Design
CapSense Proximity Design



Component Configuration:**CapSense:**

CapSense Component is configured in SmartSense Auto tuning mode with one proximity sensor for the design with the parameters shown in Table 1.

Table 2. CapSense Component Parameters

Parameter	Tab present	Value	Rationale
Tuning Method	General	Auto(SmartSense)	To automatically adjust sensitivity for different system environments
Threshold Mode		Automatic	To enable run-time threshold calculation for 5:1 SNR
Raw data noise filter		First Order IIR 1/4	Filter out noise/unwanted spikes in raw count. This setting can be tweaked based on requirement.
Immunity level		Low	Based on system needs can be set to high/low. Defaults to 'Low'.
ProximitySensor0	Widgets Config	-	Need to add a Proximity sensor by clicking on 'Proximity Sensors' and then clicking 'Add Proximity sensor'. The only parameter that is available to modify in this tab is debounce. This can be set/adjusted based on system requirement
Analog switch divider source	Advanced	PRS-12b	For reduced EMI emission and enhanced EMC immunity.
Sensor Auto reset		Disabled	Not required in the design. Can be added if required by the application.
Low Baseline Reset		5	System dependent number. Can be configured as per the user needs
Inactive sensor Connection		Ground	To make the proximity loop doesn't pick up any charge when not scanned
Shield		Disabled	Not used in the design
Guard sensor		Disabled	Not used in the design
Cmod precharge		By Vref	Vref is enough for precharging here, as there is only one sensor, Cmod voltage will not drop too low for a fast GPIO precharge
Sensitivity	Scan Order	1	To obtain the maximum possible sensitivity using SmartSense. The parameter controls the scan time, so for a lower number of sensitivity setting, scan rate will be higher. The parameter can be tweaked depending on the response rate and desired proximity range needed.

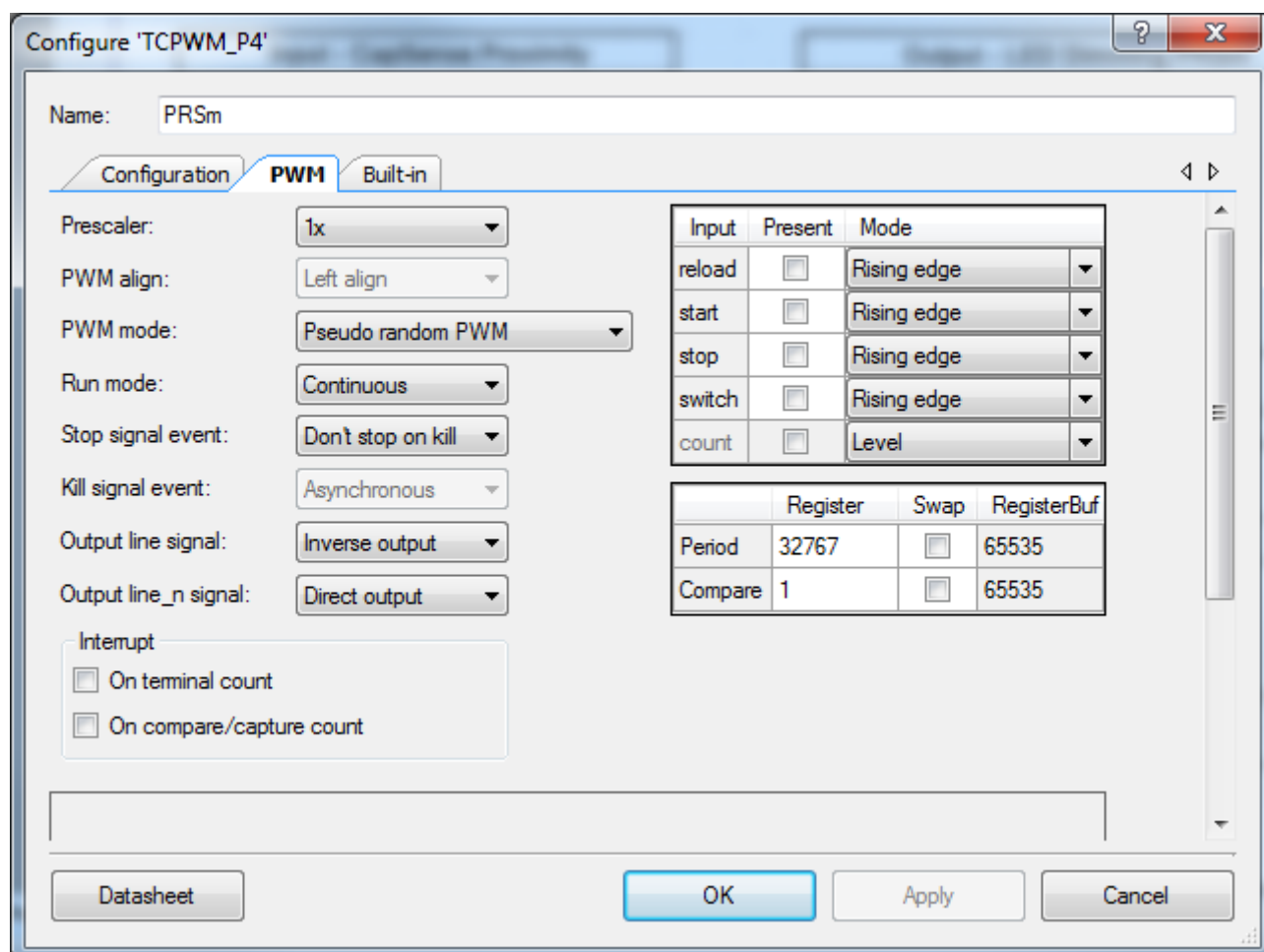
Tune helper	Tune helper	No tuner used	No tuner used
-------------	-------------	---------------	---------------

PWM (TCPWM mode)

TCPWM component is used for controlling the brightness of the LED. The CapSense Proximity sensor's signal output is used for deducing the LED brightness. The parameters for TCPWM component are as in Figure 3. The TCPWM block is configured as a PWM and in Pseudo Random Sequence modulator (PRSm) mode with a resolution of 15-bit (fixed by TCPWM block architecture). This 15-bit resolution of the PRSm along with a 16 MHz input clock generates a period of 500 Hz (PRS repeat period). The output line is inverted to drive the Active Low LED. A period of 32768 is set in the component to generate proper period macro for the 15-bit PRSm. Though the output of PRSm has a variable frequency with a max frequency of 8 MHz (16 MHz/2), the repeat rate of PRSm is considered as the period in this context.

Note: The Compare value should be minimum '1', '0' will leave the LED ON.

Figure 3. TCPWM component configuration - PWM tab

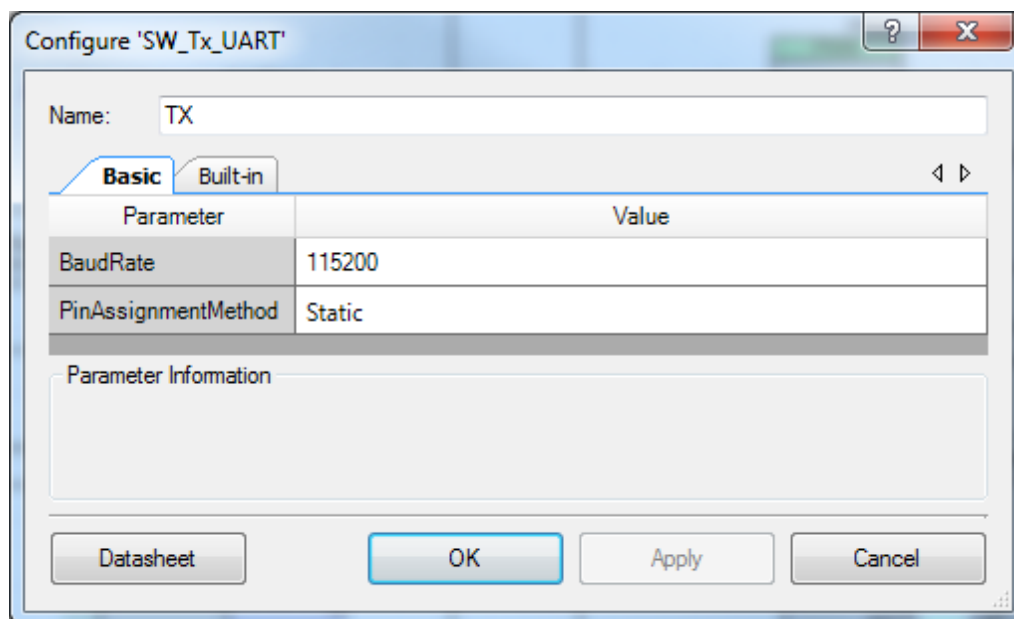


TX (SW Tx UART)

The software transmit Tx is used to send out proximity sensor related data for debugging. The configuration for the component is as in Figure 4. The SW TX can be sent over to PC using either a RS232 connector (with a translator in between) or through USB-UART bridge available

in CY8CKIT-040 PSoC 5 LP UART Bridge or CY3240 bridge configured as UART bridge as documented in [AN2397](#).

Figure 4. Software UART Tx Component parameters



PWM_Out (Digital Output Pin)

For driving the PWM output to the LED. It is a standard Strong drive output pin.

Clock_PWM (Clock)

It provides the clock that drives the PWM block. The clock is configured to be the max possible/allowed (16 MHz), so that repeat rate of the PRSm is as high as possible for reduced LED flickers.

Project Description

A capacitive proximity sensor controlling the brightness of a LED is implemented in the project. The project configures the sensor as a CapSense proximity widget with SmartSense auto tuning. The firmware flow is presented in Figure 5. The CapSense/Input initialization part tunes the CapSense system parameters using SmartSense. The output initialization part configures the PWM and the software UART Tx output. The infinite loop code is divided into two phases – Input process and Output process.

The input process phase scans the proximity sensor, processes the sensor signal, such as applying filter, calculating baseline and filtered signal. The output processing phase is also split into two phases – data calculation and data output sub-phases. In the data calculation sub-phase, the proximity signal is compared against a minimum and maximum threshold defined for an approaching hand. The LED brightness is then calculated based on the sensor's signal value relative to the thresholds. Minimum threshold generates the lowest LED brightness and maximum limit generates the highest LED brightness.

The data output phase, updates the PWM compare value with the calculated brightness. The system data such as sensor raw data, baseline, signal and calculated LED brightness are sent

over UART Tx line. The device monitors the activity on the proximity sensor and if there is no activity i.e. if the hand is out of range of the proximity, then the device enters a Sleep-scan mode. The time for which the device checks for a no activity on the sensor before entering Sleep-scan mode is set to one second and is configurable in the project (ENTER_SLEEP_COUNTS macro in main.h). In the Sleep-scan mode, the device wakes up every 100 ms and checks for any activity on the proximity sensor. This wakeup rate is configurable by modifying the 'WATCHDOG_TIMER_COUNT' macro in the 'main.h' file.

Note: By default debug is disabled and UART TX line is enabled on SWD IO line (P3[0]) in the project. If debug is desired, then the Debug Select parameter should be set to SWD (serial wire debug) value as shown in Figure 6. If debug is enabled, TX should be disabled.

The periodic interrupts from WDT may complicate debugging of project due to frequent jumps of the instruction pointer into the interrupt handler. To make debugging more comfortable, interrupts can be disabled by using option “Disable global interrupts” from menu “Debug”.

Figure 5. Firmware flowchart

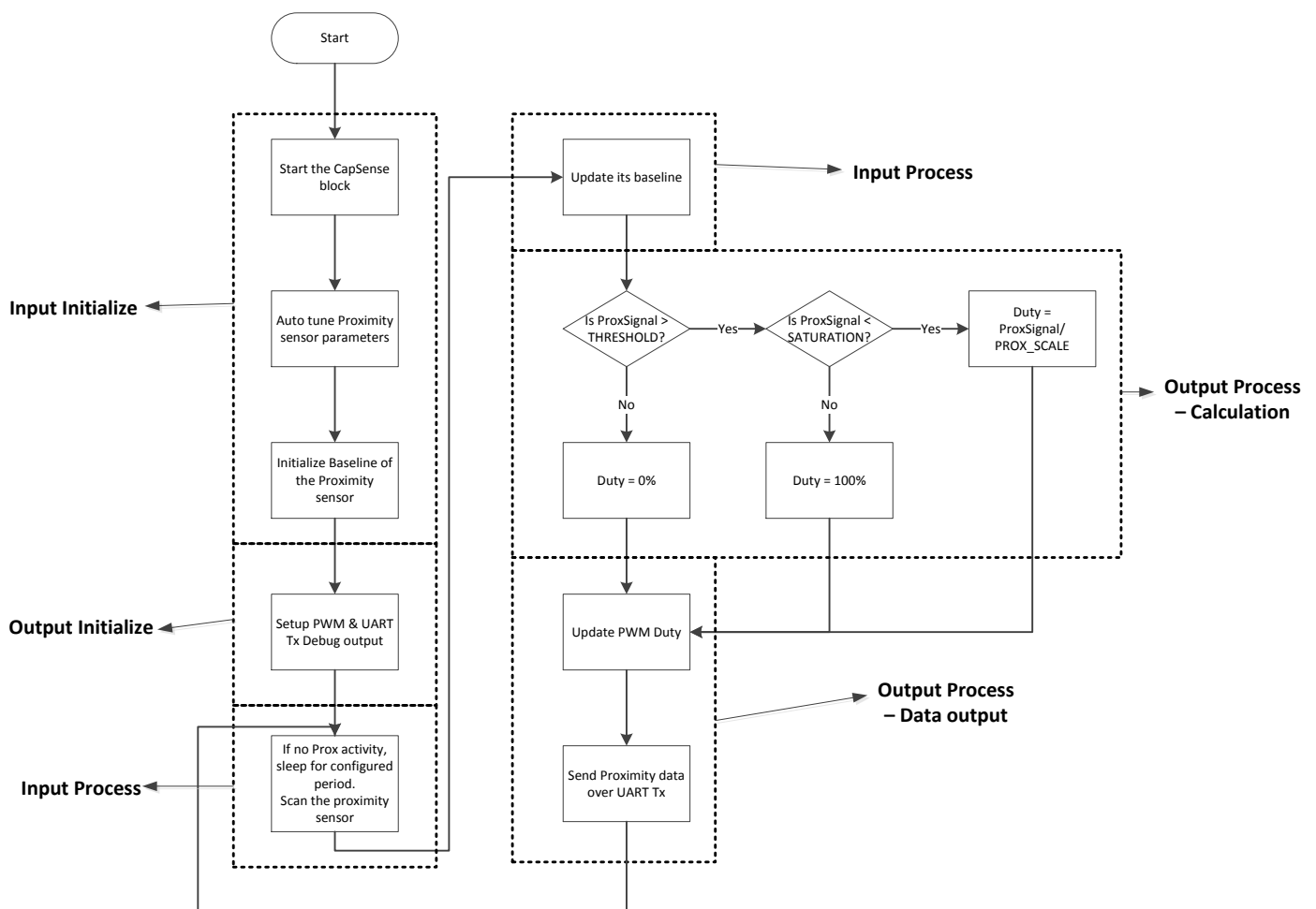
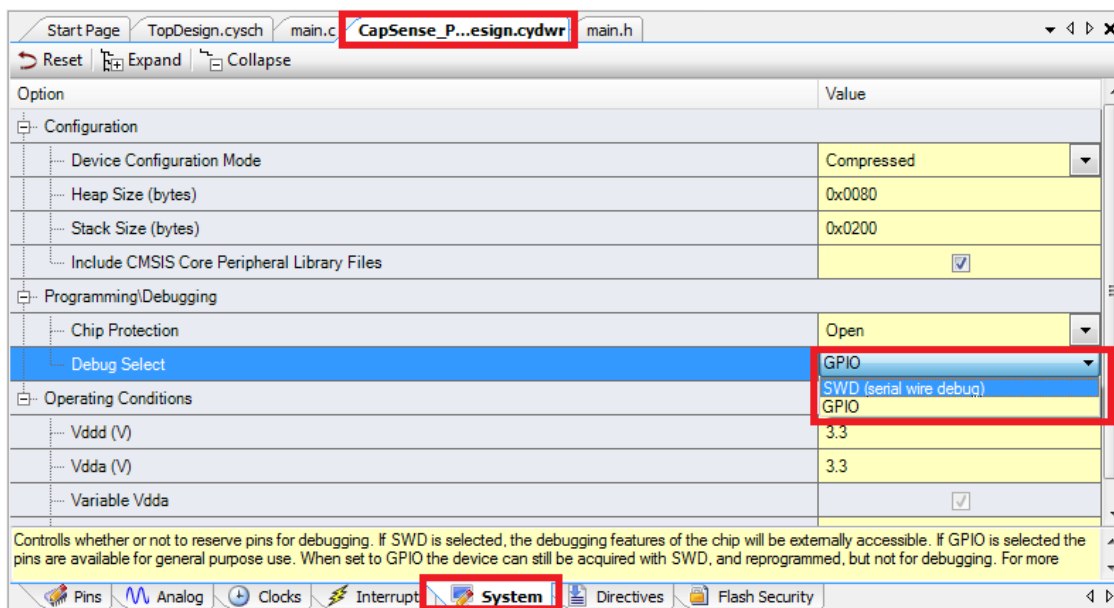


Figure 6. Enabling Debug in the project

Debug Interface (UART)

For viewing the debug data transmitted over UART Tx line, Bridge Control Panel software is used. One UART packet size is 13 bytes which includes 8 bytes of data, 2 bytes of header and 3 bytes of footer. The two byte header precedes the data bytes and in the design it is 0x0D and 0x0A. The three byte footer follows the data bytes and in this design consists of 0x00, 0xFF and 0xFF. The data itself consists of proximity sensor raw counts (RC), baseline (BL) and signal (SIG) along with the calculated PWM duty (DUTY). The UART Tx data packet structure is as shown below –

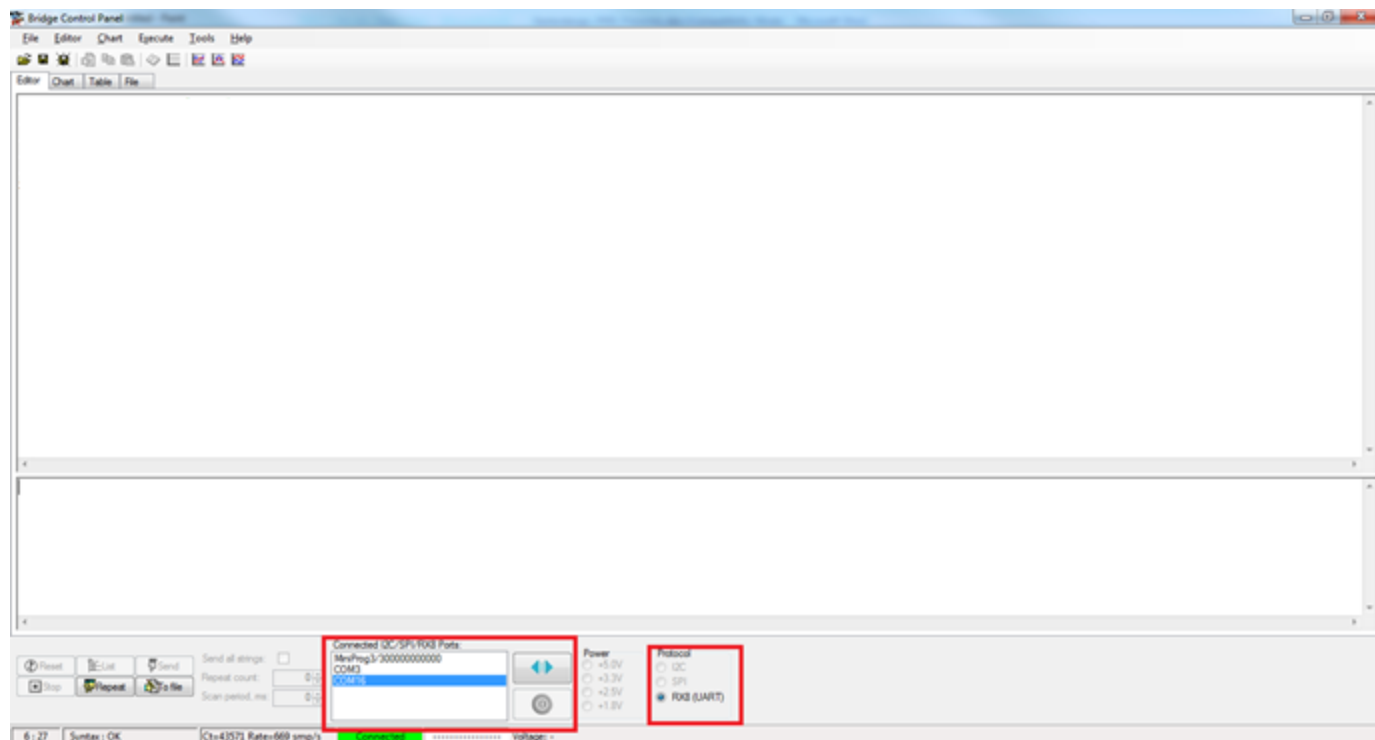
Table 3. UART Tx Packet Structure

Header		Data				
BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6
0x0D	0x0A	RC_MSB	RC_LSB	BL_MSB	BL_LSB	SIG_MSB
Data		Footer				
BYTE 7	BYTE 8	BYTE 9	BYTE 10	BYTE 11	BYTE 12	
SIG_LSB	DUTY_MSB	DUTY_LSB	0x00	0xFF	0xFF	

Follow the below steps to setup Bridge Control Panel (BCP) for viewing the data with CY8CKIT-040's USB-UART bridge,

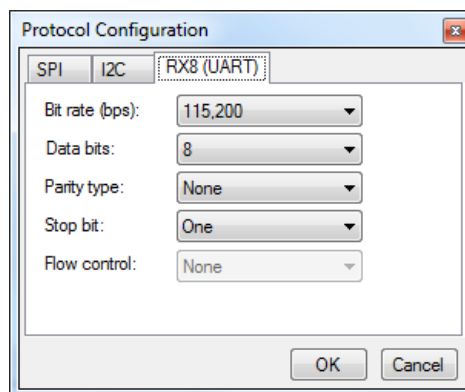
1. Open Bridge control Panel software available under 'All programs -> Cypress -> Bridge Control Panel -> Bridge Control Panel.
2. Route the Tx pin of the device to any available Rx which can connect to the PC Com port. CY3240 (refer [AN2397](#)) or KitProg in CY8CKIT-040 can be used for this purpose. In CY8CKIT-040, Pin 3[0] (Tx line) is directly connected to RX line (P12[6]) of PSoC 5LP bridge (Zero Ohm R57 in the board needs to be populated).
3. In the Bridge Control Panel software, click on the COM port to which you have connected the data. In our case, it is KitProg's COM (can be found in Device manager).
4. Select RX8 as protocol, as shown in Figure 7

Figure 7. Bridge Control Panel - COM Port and Protocol Selection



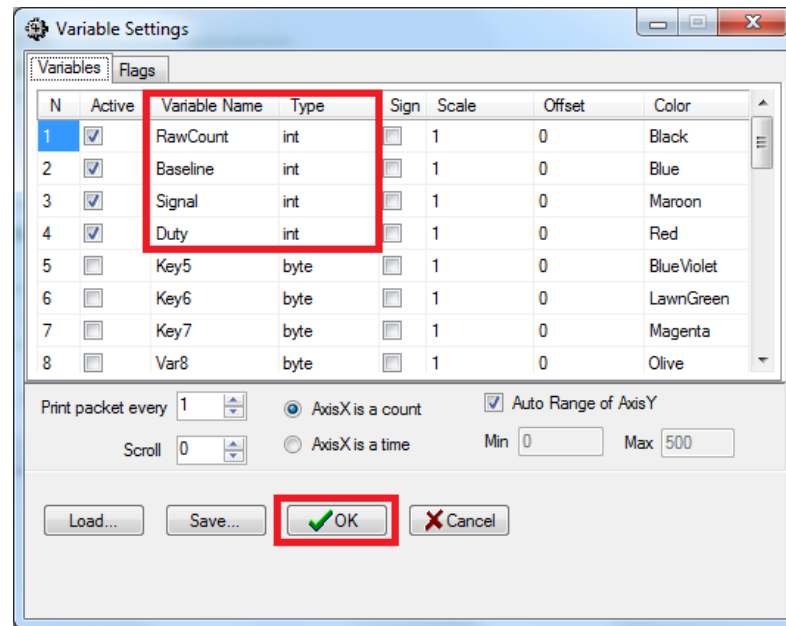
5. Click on 'Tools -> Protocol Configuration' or Press 'F7' and configure the RX8 protocol parameters as shown below.

Figure 8. RX8 Protocol Configuration



6. Click on 'Chart -> Variable Settings' and Set the variable names and types as shown in Figure 9 and then press 'OK'

Figure 9. Bridge Control Panel - Variable settings



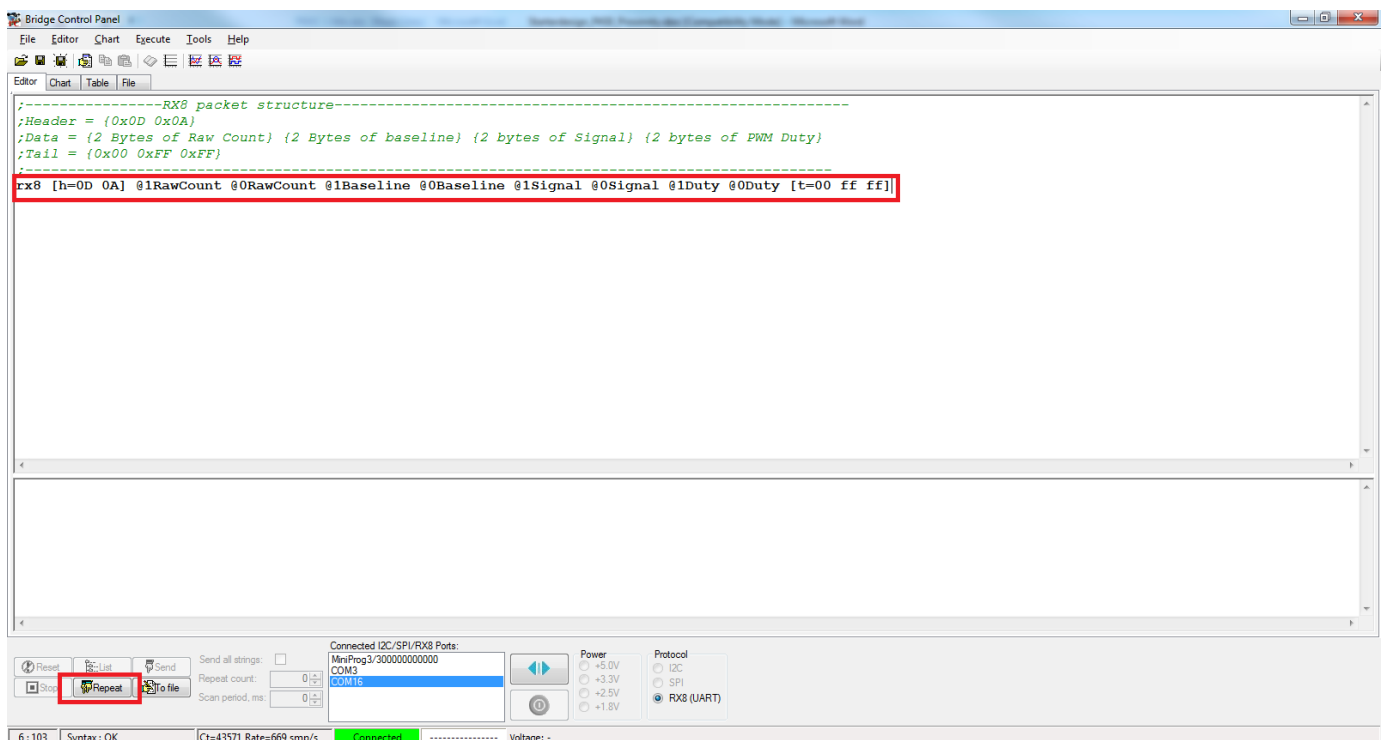
- Go to editor and Type or copy the command as shown below

Command:

```
rx8 [h=0D 0A] @1RawCount @0RawCount @1Baseline @0Baseline @1Signal @0Signal @1Duty @0Duty [t=00 ff ff]
```

- Click 'Repeat' as shown below to start receiving the packets (make sure you have powered the device and programmed with the project's firmware and the Tx is connected to Rx line of the COM and the COM port is selected in BCP)

Figure 10. Bridge Control panel - Protocol execution



9. You should start receiving data, Click on ‘Chart’ tab to view the graph.

Expected Results

Following table provides the expected behavior for a wire loop of approximately 5 cm diameter.

Table 4. Expected Results

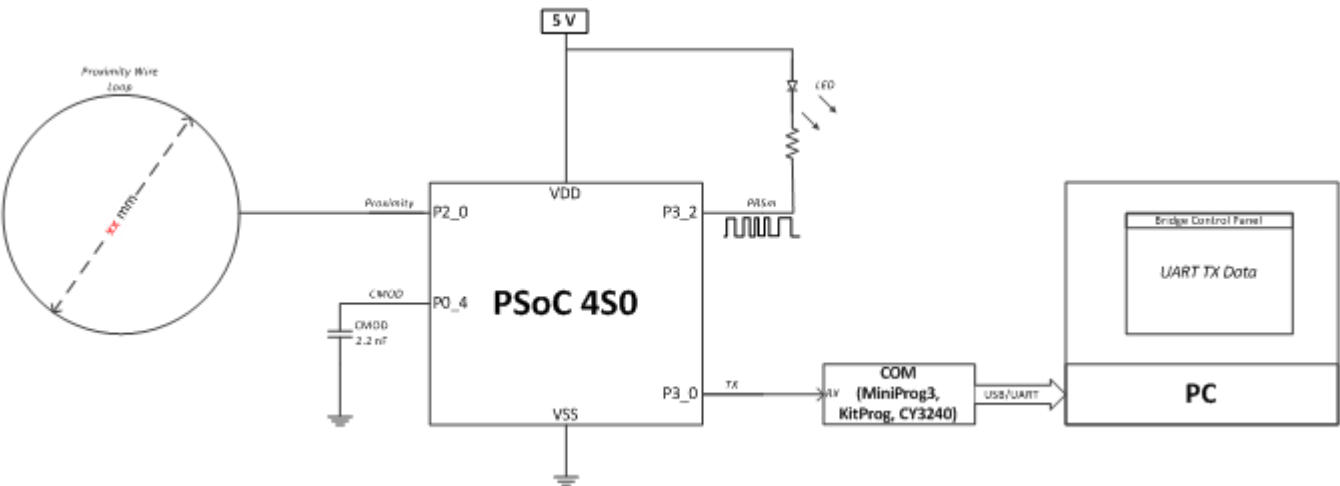
Hand Distance(in cm)	Signal observed in BCP GUI	LED state
5-6 ¹⁾	40-50	Just turns ON with lowest brightness
1	250	LED with full brightness

¹⁾ The detected distance strongly depends on the wire loop diameter and form.

Schematic and Pin Mapping

Schematic:

Figure 11. Schematic of the Project



Pin mapping table:

Table 5. Pin Mapping and Alternate pin options

Function	Pin	Alternate Pins
Proximity Sensor	P2[0]	Any CapSense supported GPIO. Refer datasheet for the supported GPIOs
Modulator Capacitor	P0[4]	Fixed to P0[4]
LED	P3[2]	P1[1]. Alternately P1[6] can be used as well for an inverted output.
UART Tx	P3[0]	Any available GPIO.



Cypress Semiconductor Phone : 408-943-2600
198 Champion Court Fax : 408-943-4730
San Jose, CA 95134-1709 Website : www.cypress.com

© Cypress Semiconductor Corporation, 2016. 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 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, 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.