PSoC 4 Lab 3 - CapSense UART - Lab Manual

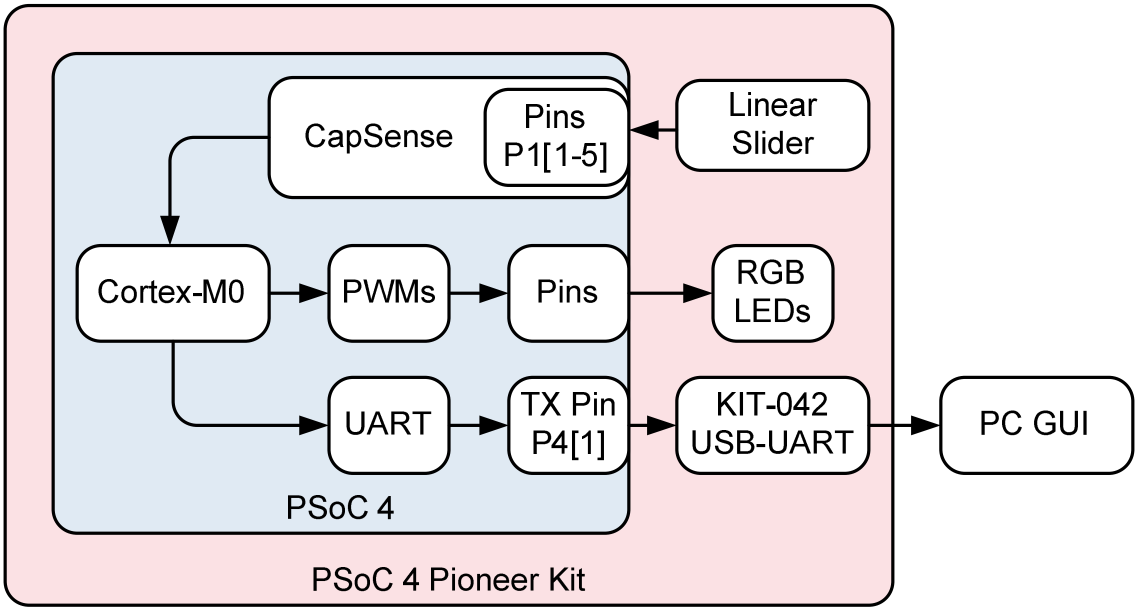
Objectives

- To learn how to use the CapSense component to detect the position of a finger on a multi-segment slider.
- To learn how to use the UART component, the USB-UART bridge on the Pioneer Kit, and a PC GUI to debug designs.

Requirements	Details
Hardware	CY8CKIT-042 PSoC 4 Pioneer Kit
Software	PSoC Creator 2.2 SP1, Bridge Control Panel 1.8
Firmware	Lab 3 Template
Components used	CapSense, UART

Block Diagram

Figure 1. Lab 3 Block Diagram



Theory

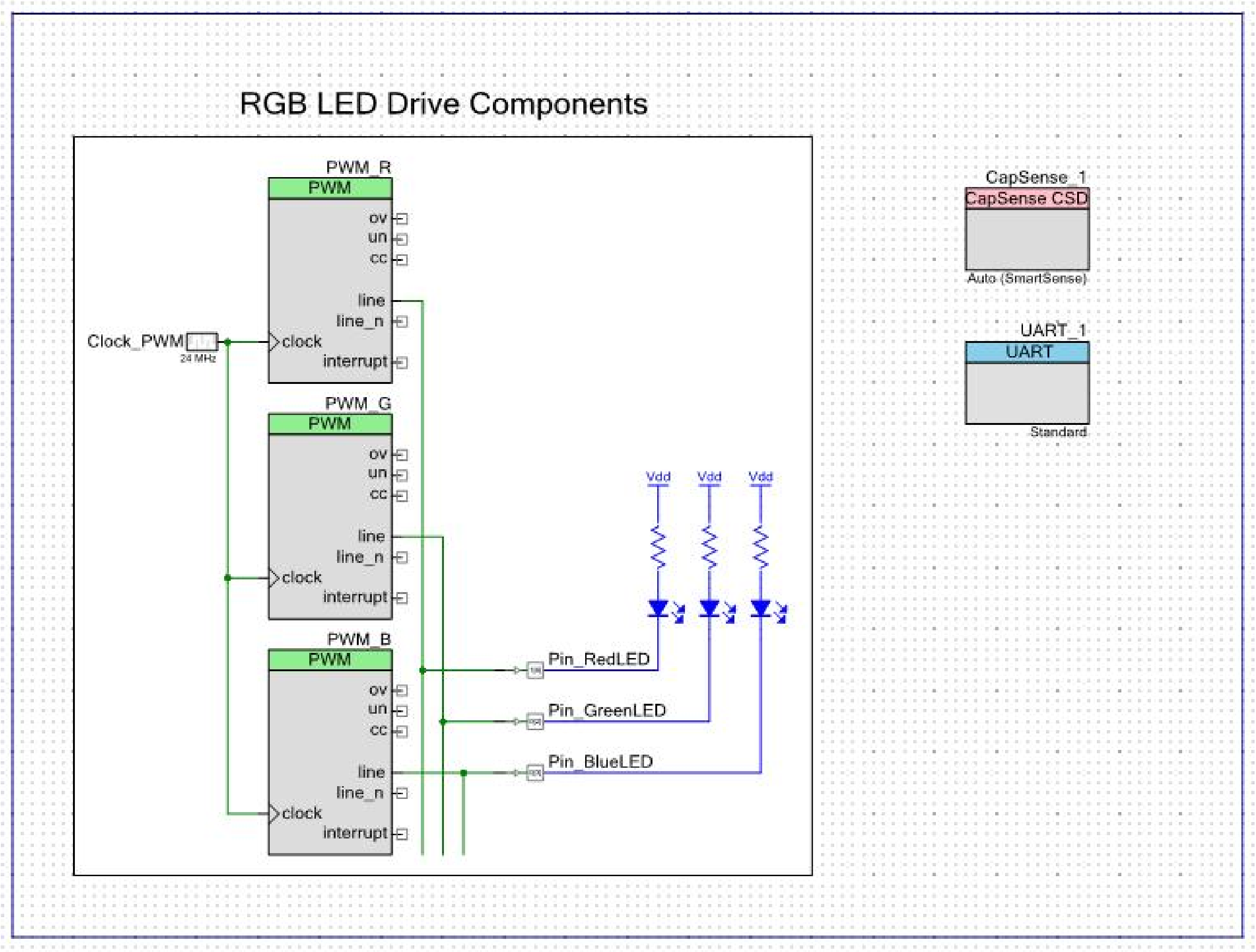
The goal of this lab is to learn how to use PSoC 4's CapSense component to sense the position of a finger on a linear slider, and use the LEDs to display that position. PSoC 4's dedicated CapSense hardware measures finger presence by measuring the change in capacitance when a finger is introduced to a capacitive sensor, normally a small copper shape on a PCB.

The lab will also demonstrate how to use a UART to transmit the position to a PC by using the USB-UART functionality of the Pioneer kit, and the Bridge Control Panel software that ships with PSoC Programmer. PSoC 4 contains 2 Serial Communication Blocks (SCBs) that can implement UART, SPI, or I2C traffic. We will use the UART component to implement UART functionality.

Procedure – Firmware

1) Open the PSoC 4 Intro Lab Templates workspace, if it's not already open. You will be presented with a workspace in PSoC Creator containing one template project for each of the labs after Lab 1.
2) Set the Lab 3 template project as the active project by right-clicking on it in the Workspace Explorer, and clicking on the "Set As Active Project" option. In its initial state, this project will cycle through mixed RGB colors at a rate of 1 Hz.
3) Open the project's schematic by double-clicking on the "TopDesign.cysch" file in the Workspace Explorer. Note that in this schematic, we've included three PWMs and pins, along with the "LED_RGB" module to allow for easy driving of tri-colored LEDs.
4) In the component catalog, under the "Capsense" category, select the "Capsense CSD" component, and drag it into the schematic.
5) In the component catalog, under the "Communications" category, select the "UART (SCB mode)" component, and drag it into the schematic. The schematic should look like the one shown in Figure 2.

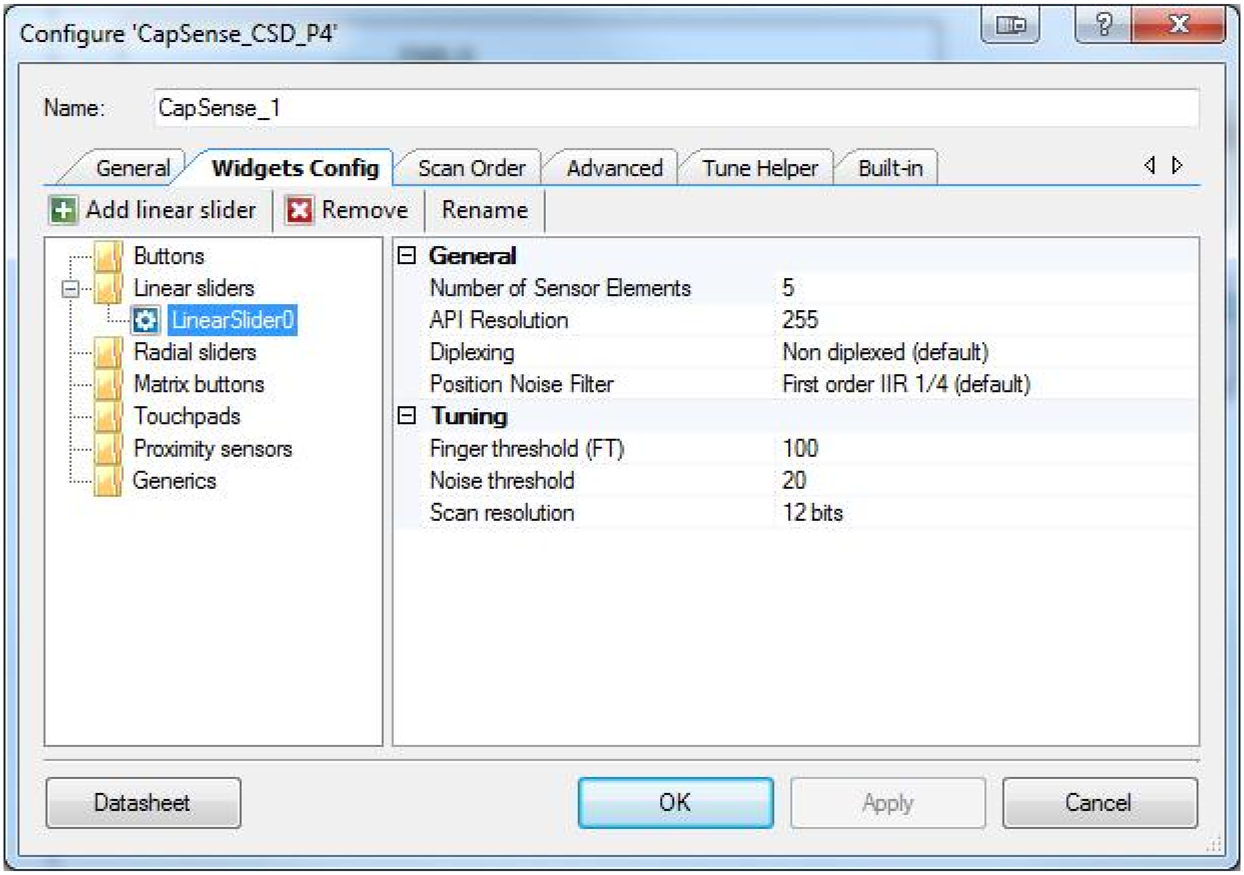
Figure 2. Schematic With CapSense and UART Placed



6) Open the CapSense component customizer by double-clicking on the component. Click "Load settings" to automatically configure the component for the kit's CapSense slider. Select the file "CY8CKIT-042 CapSense Slider.xml" in the lab template workspace directory and press "Open".

7) Navigate to the "Widgets Config" tab, and click on the "LinearSlider0" widget to examine its settings. These settings are used to configure the CapSense sensors, and are described in detail in the CapSense component datasheet. Press the "OK" button when you are done looking at the settings. This window is shown in Figure 3.

Figure 3. CapSense Widget Configuration Tab

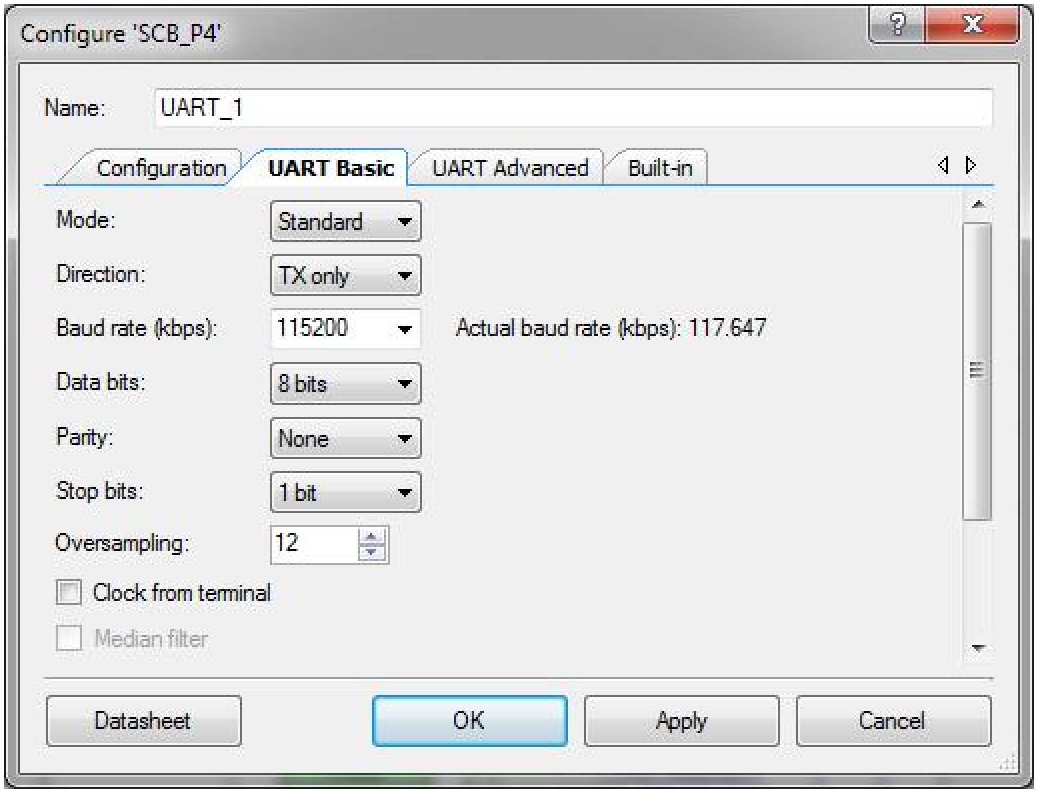


Name: CapSense_1

General	
Number of Sensor Elements	5
API Resolution	255
Diplexing	Non diplexed (default)
Position Noise Filter	First order IIR 1/4 (default)
Tuning	
Finger threshold (FT)	100
Noise threshold	20
Scan resolution	12 bits

8) Open the UART component customizer by double-clicking on the component. Navigate to the “UART Basic” tab to configure the UART settings. Change the value of the “Direction” dropdown to “TX only” to set the UART up for uni-directional communication. Press “OK” to close the GUI and apply changes. The GUI correctly configured is shown in Figure 4.

Figure 4. UART Configuration GUI



Configure 'SCB_P4'

Name: UART_1

Mode: Standard

Direction: TX only

Baud rate (kbps): 115200

Actual baud rate (kbps): 117.647

Data bits: 8 bits

Parity: None

Stop bits: 1 bit

Oversampling: 12

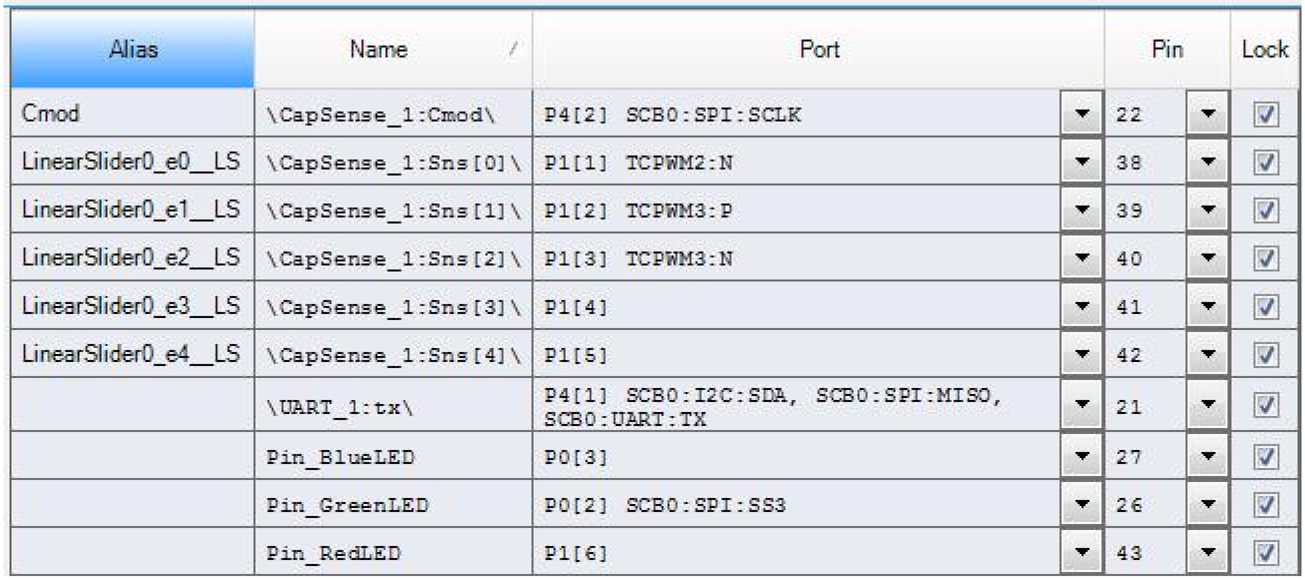
☐ Clock from terminal

☐ Median filter

9) Double click on the “Lab 3 CapSense UART.cydwr” file to open the design wide resources interface.

10) On the “Pins” tab, assign the “UART_1:tx” pin to P4[1], the “CapSense_1:Cmod” pin to P4[2], and the “CapSense_1:Sns[0-4]” pins to P1[1] through P1[5], in either ascending or descending order. The completed pin assignment should look something like that shown in Figure 5.

Figure 5. Capsense and UART Pin Mapping



Alias	Name	Port	Pin	Lock
Cmod	\CapSense_1:Cmod\	P4[2] SCB0:SPI:SCLK	22	☑
LinearSlider0_e0__LS	\CapSense_1:Sns[0]\	P1[1] TCPWM2:N	38	☑
LinearSlider0_e1__LS	\CapSense_1:Sns[1]\	P1[2] TCPWM3:P	39	☑
LinearSlider0_e2__LS	\CapSense_1:Sns[2]\	P1[3] TCPWM3:N	40	☑
LinearSlider0_e3__LS	\CapSense_1:Sns[3]\	P1[4]	41	☑
LinearSlider0_e4__LS	\CapSense_1:Sns[4]\	P1[5]	42	☑
	\UART_1:tx\	P4[1] SCB0:I2C:SDA, SCB0:SPI:MISO, SCB0:UART:TX	21	☑
	Pin_BlueLED	P0[3]	27	☑
	Pin_GreenLED	P0[2] SCB0:SPI:SS3	26	☑
	Pin_RedLED	P1[6]	43	☑

11) In the “Workspace Explorer”, double-click the “main.c” file to open it in the code editor.

12) Replace the “Change1” line with the initialization code, shown in Code 1.

Code 1. Lab 3 “Change1” Initialization Code

```
    CapSense_1_Start();
    CapSense_1_InitializeAllBaselines();
    UART_1_Start();
```

13) Replace the “Change2” line with the Capsense slider position read API, shown in Code 2.

Code 2. Lab 3 “Change2” Capsense Slider Position Read API

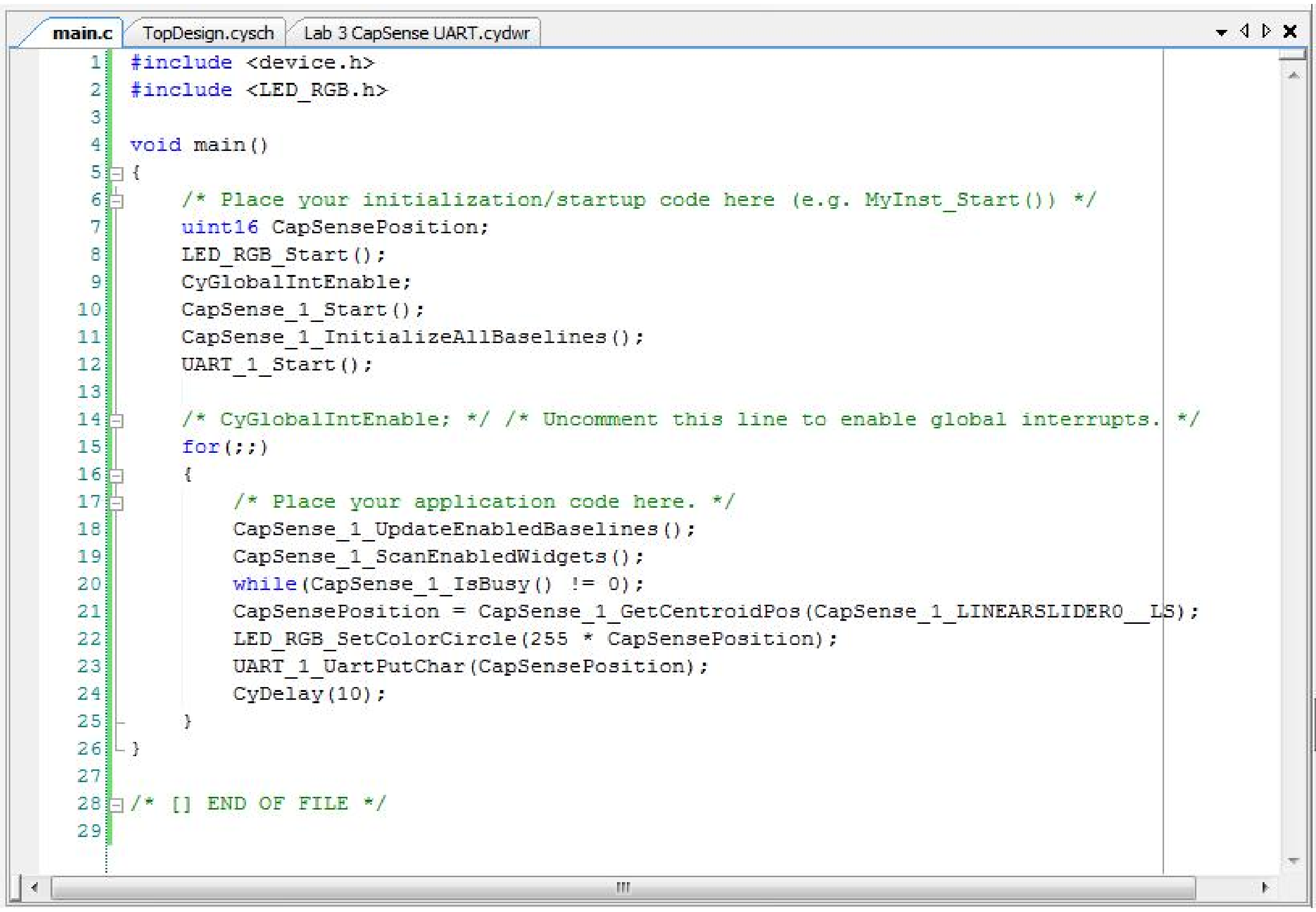
```
        CapSensePosition = CapSense_1_GetCentroidPos(
CapSense_1_LINEARSLIDER0__LS);
```

14) Replace the “Change3” line with the Capsense slider position read API, shown in Code 3. The entire main.c should look like that shown in Figure 6.

Code 3. Lab 3 “Change3” UART Put Char API

```
        UART_1_UartPutChar(CapSensePosition);
```

Figure 6. Lab 3 Solution main.c

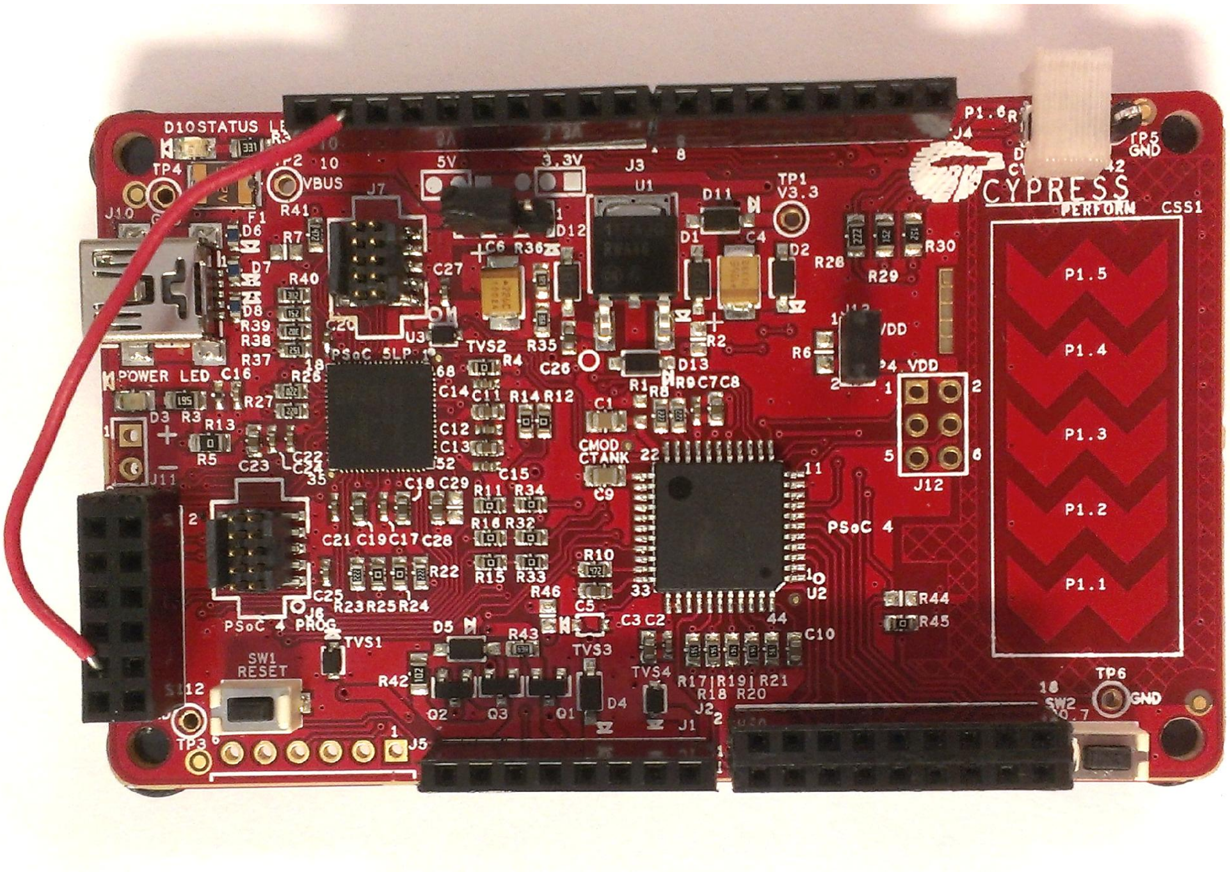


```c
#include <device.h>
#include <LED_RGB.h>

void main()
{
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    uint16 CapSensePosition;
    LED_RGB_Start();
    CyGlobalIntEnable;
    CapSense_1_Start();
    CapSense_1_InitializeAllBaselines();
    UART_1_Start();

    /* CyGlobalIntEnable; */ /* Uncomment this line to enable global interrupts. */
    for(;;)
    {
        /* Place your application code here. */
        CapSense_1_UpdateEnabledBaselines();
        CapSense_1_ScanEnabledWidgets();
        while(CapSense_1_IsBusy() != 0);
        CapSensePosition = CapSense_1_GetCentroidPos(CapSense_1_LINEARSLIDER0__LS);
        LED_RGB_SetColorCircle(255 * CapSensePosition);
        UART_1_UartPutChar(CapSensePosition);
        CyDelay(10);
    }
}

/* [] END OF FILE */
```

15) Press the “Program” button on the PSoC Creator toolbar to build the project and program your kit. After programming, you should be able to change the color of the tri-color LED by moving your finger up and down along the CapSense Slider.

16) Using one of the wires shipped with the kit, connect PSoC 4 pin P4[1] (connector J3 pin 10) with PSoC 5LP pin P12[6] (connector J8 pin 9). This is shown in Figure 7.

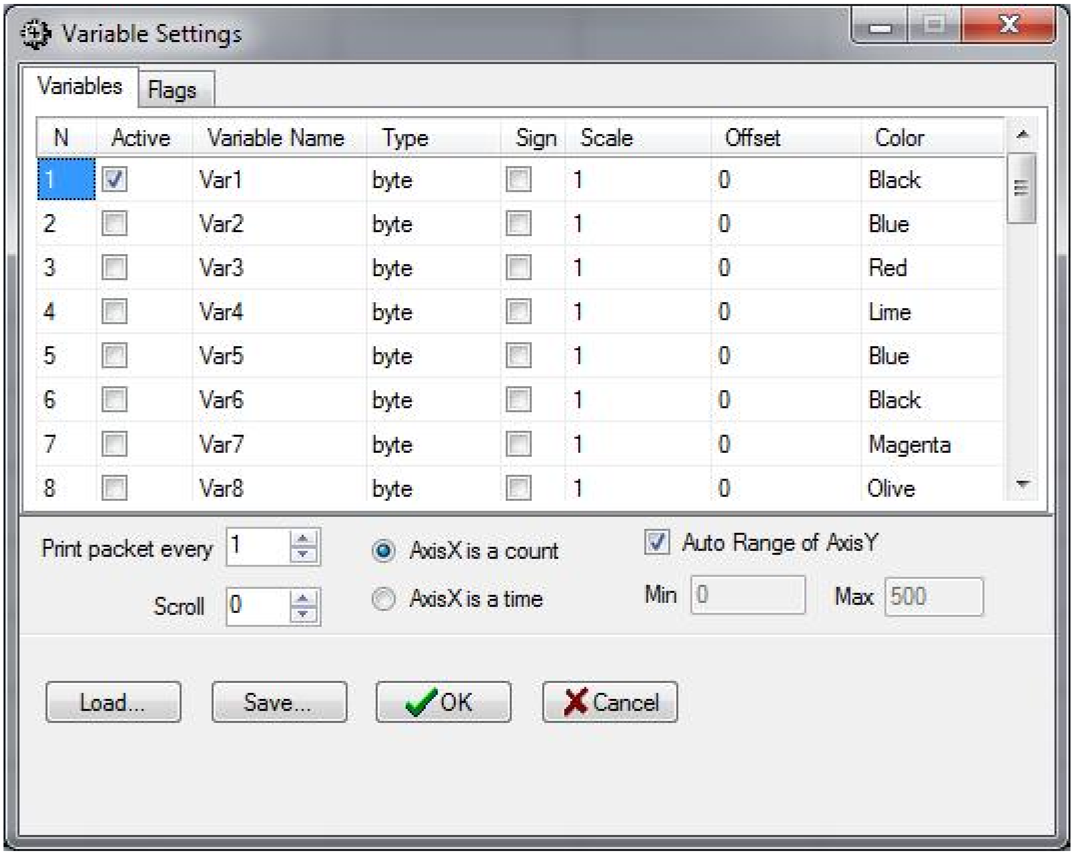
Figure 7. PSoC 4 Pioneer Kit UART TX Connection



17) Open the Bridge Control Panel software, which is in your Start menu under “Cypress -> Bridge Control Panel”.

18) Click on the “Chart” menu, and select “Variable Settings”. Ensure that the first variable is named “Var1”, it’s “Active” checkbox is checked, the “Type” is “byte” and the “Sign” checkbox is unchecked. Press “OK” to close the window. This is shown in Figure 8.

Figure 8. Bridge Control Panel Variable Settings Configuration



Variable Settings

N	Active	Variable Name	Type	Sign	Scale	Offset	Color
1	☑	Var1	byte	□	1	0	Black
2	□	Var2	byte	□	1	0	Blue
3	□	Var3	byte	□	1	0	Red
4	□	Var4	byte	□	1	0	Lime
5	□	Var5	byte	□	1	0	Blue
6	□	Var6	byte	□	1	0	Black
7	□	Var7	byte	□	1	0	Magenta
8	□	Var8	byte	□	1	0	Olive

Print packet every 1

Scroll 0

● AxisX is a count

○ AxisX is a time

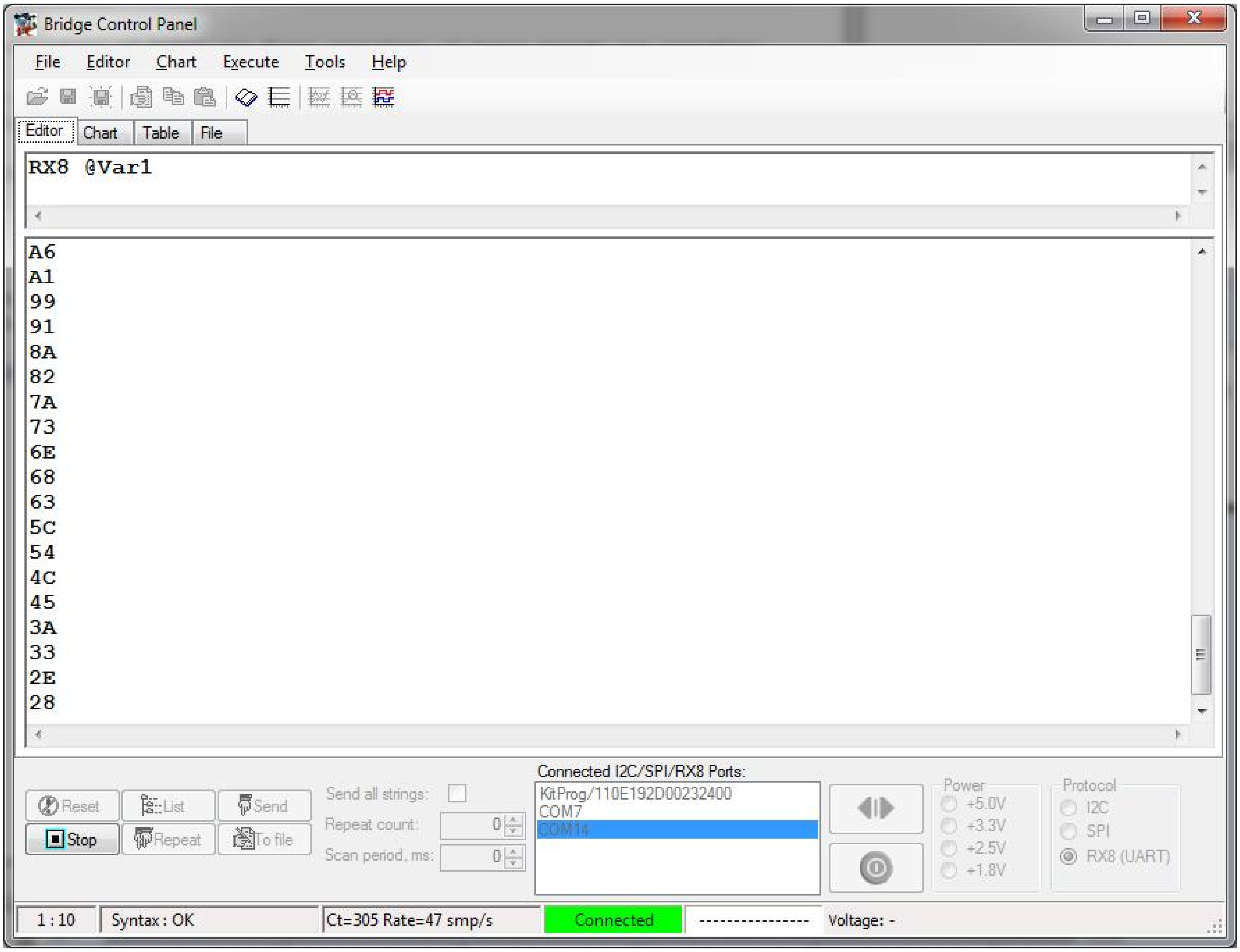
☑ Auto Range of AxisY

Min 0 Max 500

19) In the editor tab, select the KIT-042 COM port in the “Connected I2C/SPI/RX8 Ports:” list at the bottom of the window. It should be the highest numbered COM port in the list. The box below the ports list should turn green and say “Connected”.

20) In the editor text box, enter the RX8 packet instruction “`RX8 @Var1`”. With the cursor still on the line that you entered, press the “Repeat” button. Hex bytes should begin to appear in the console below the editor. Move your finger across the CapSense slider and watch the values change. It should look like the image shown in Figure 9.

Figure 9. Bridge Control Panel Data Transmission



```
RX8 @Var1
```

```
A6
A1
99
91
8A
82
7A
73
6E
68
63
5C
54
4C
45
3A
33
2E
28
```

Send all strings: □
Repeat count: 0
Scan period, ms: 0

Connected I2C/SPI/RX8 Ports:
KitProg/110E192D00232400
COM7
COM14

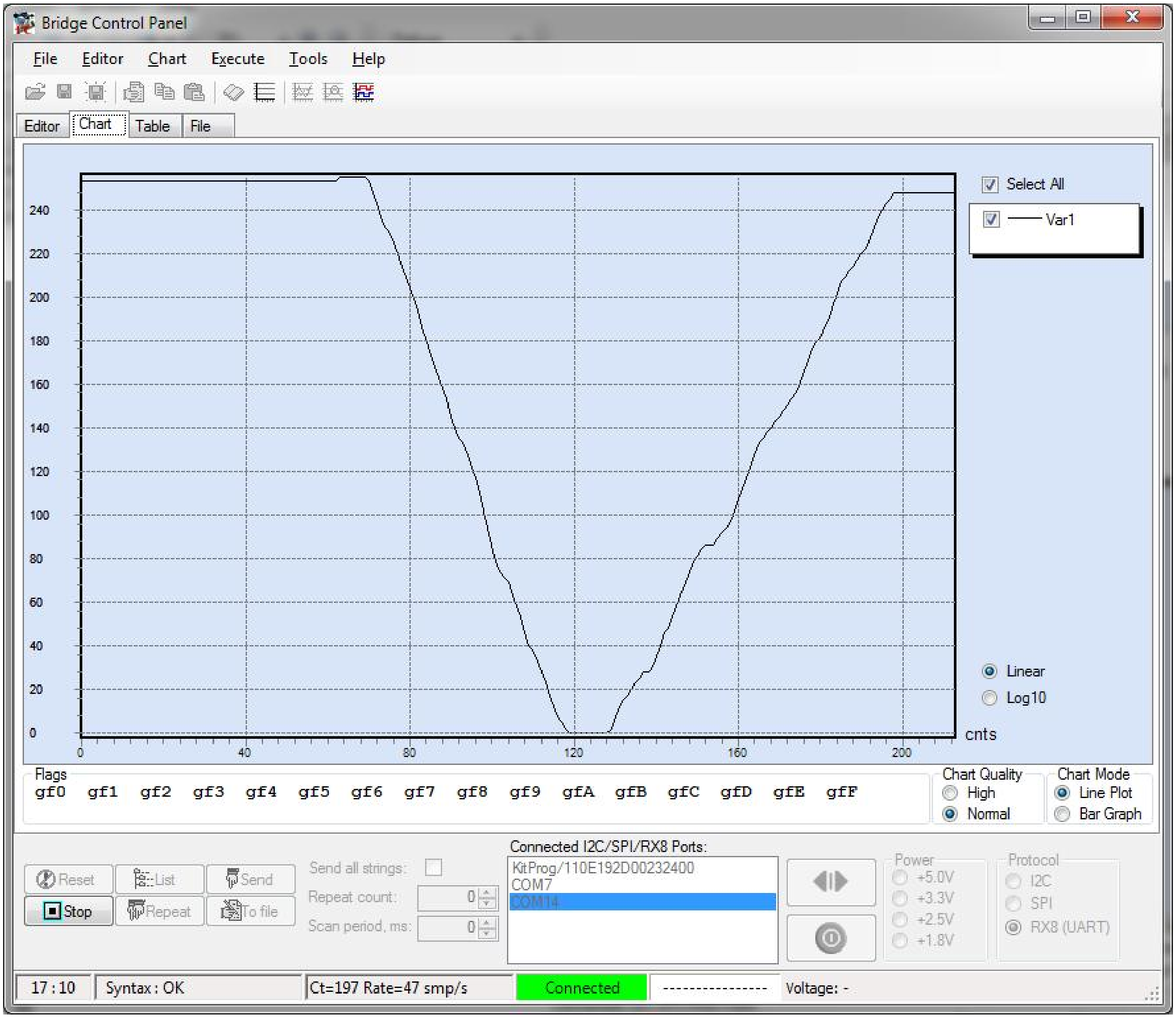
Power
○ +5.0V
○ +3.3V
○ +2.5V
○ +1.8V

Protocol
○ I2C
○ SPI
● RX8 (UART)

1:10	Syntax: OK	Ct=305 Rate=47 smp/s	Connected	----------------	Voltage: -

21) Click on the “Chart” tab, and observe the values being charted over time. If you want to clear the graph and start over, Press the “Stop” button followed by the “Repeat” button. When sweeping your finger across the slider, the chart should like that shown in Figure 10.

Figure 10. Bridge Control Panel Chart Interface



☑ Select All
☑ Var1
● Linear ○ Log10
Chart Quality: ○ High ● Normal
Chart Mode: ● Line Plot ○ Bar Graph
Protocol: ○ I2C ○ SPI ● RX8 (UART)
Repeat count: 0
Scan period, ms: 0

Connected I2C/SPI/RX8 Ports:
KitProg/110E192D00232400
COM7
COM14

17 : 10 | Syntax : OK | Ct=197 Rate=47 smp/s | Connected | ---------------- | Voltage: -

Conclusion

- You have successfully implemented a CapSense slider and UART communication on your PSoC 4 Pioneer Kit. You have also learned how to use the Bridge Control Panel to communicate with the Pioneer Kit, and graph the communicated data.

Stretch Goals

- Retain the last color when the finger is removed from the CapSense slider.
    - In this lab, we don't examine the CapSense slider position to ensure that it isn't returning a "no finger" code, so the slider turns red when that code is returned.
    - The "`CapSense_1_GetCentroidPos()`" API returns `0xffff` when no finger is detected. If this value is detected by firmware, the LED color can be left alone, and the last detected position's color will remain on the LEDs, even when the finger is removed.
- Transmit the whole 16-bit variable "color".
    - In this lab, we shifted the 16 bit variable down by 8 bits and transmitted only the most significant byte (MSB).
    - We could transmit both bytes by masking them individually, and using a control character to indicate coherency.
    - In Bridge Control Panel you will need to configure your variable to be an "int" instead of a "byte", receive both bytes of data, and interpret the packet including the control character. An RX8 command for this would look something like "`RX8 [h=43] @1intColor @0intColor`" if the control character was hex 0x43 (character 'C').
- Implement bidirectional communications
    - In this lab, we only implemented single direction communication from PSoC 4 Pioneer Kit to the PC.
    - Try implementing communication from the PC to the Pioneer Kit, perhaps setting the LED color using Bridge Control Panel.

Document Revision History

Revision	By	Description
01	MAXK	First Release