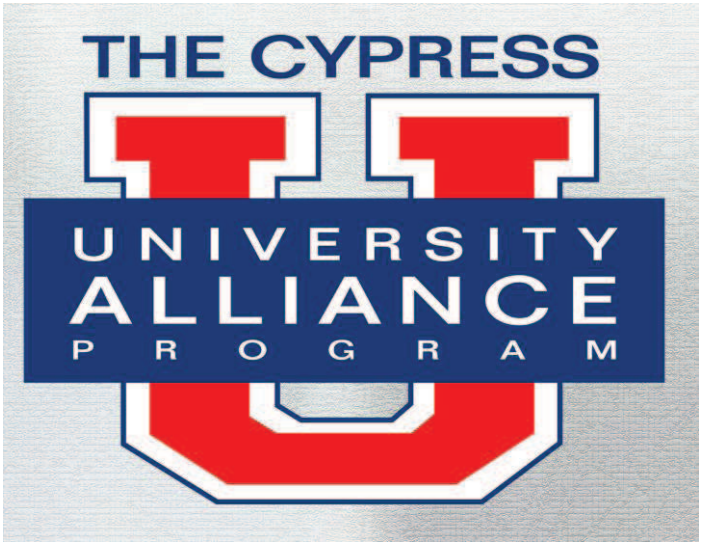
THE PSoC 3 LABBOOK



The experiments in this lab book were designed, implemented, tested and documented by

ADITYA YADAV bearing ID No-2006B2A8648G

And

ASHWATH KRISHNAN K bearing ID No-2007A3PS061G

Of

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE,
PILANI (KK BIRLA GOA CAMPUS)

ACKNOWLEDGEMENTS

This lab book in its present version would have been impossible without several people.

We would like to thank Dr K R Anupama, Group Leader Electrical, Electronics and Instrumentation Department at BITS Pilani, KK Birla Goa Campus, for helping us cross every administrative barrier as well as for providing valuable suggestions.

We also thank Mr Patrick Kane and Cypress University Alliance for signing an MoU with BITS Pilani, and funding the PSoC 3 lab at the campus. The MoU laid the foundation for this manual and the lab provided us the necessary resources for our research.

Mr. Karthikeyan Mahalingam from Cypress, Bangalore, was our focal point of contact throughout this project and provided us with critical feedback that helped us improve this manual.

Mr. Ganesh Raaja, Mr. Kannan Sadasivam, Mr Sai Prashanth, all from Cypress provided crucial suggestions to improve the documentation of this manual. We are indebted to them.

Mr. Angad Singh Gill at Cypress San Jose, Mr.Rohan Gandhi at Purdue, Indiana and Mr. Rahul Parsani, at National University of Singapore provided the impetus for this project. We are grateful to them for their assistance and willingness to help at all times.

Our thanks also to Jamie Allan, Director, Sales & Marketing Artaflex, Emil Pop Program Manager Artaflex and Sheldon Benjamin, Technical Sales. Artaflex for helping us complete the experiments which employ the wireless module by Artaflex.

Finally, we would like to acknowledge the efforts of Akash Raman, Abhinav Asthana, Debanshu Singh, Rishabh Ginotra, Karandeep Singh, Manil Gupta, Mihir Shete, Venkat Sanjay and Utkarsh Sinha for helping us at various stages of our experiments.

Aditya Yadav (adityayadav88@gmail.com)
Ashwath Krishnan K (ashwath.krishnan7@gmail.com)

August 15 2011

PREFACE

The Programmable System on Chip (PSoC) has evolved over the last few years. However, no change has been as significant as the transition from PSoC 1 to PSoC 3. Moving from a simple M8C core to a more powerful and efficient 8051 core, the PSoC is set to take embedded systems to a new high.

Although fundamentally the process of designing any application is the same, there are several new aspects to the PSoC 3. Additional flexibility for hardware configurations and the ever expanding set of peripherals make the PSoC 3 ideal for powerful applications on chip.

The introduction of PSoC Creator over the earlier used PSoC Designer/ PSoC Express brings an additional dimension in terms of challenges for a novice seeking to explore and master the PSoC 3.

Though Cypress Semiconductors has come out with an extensive and thorough Technical Reference Manual for the PSoC 3, a starting point is needed for a novice to grasp this technology.

This manual was written keeping in mind the needs of a student starting from the grass root level. Attempts were made to cover every small step that would make life easier and hassle free for anyone attempting to learn the PSoC 3 with fundamental knowledge of Microprocessors, Analog and Digital Electronics.

With the advent of the PSoC 5 with a lethal ARM core not too far away, this manual serves as the perfect bridge for anyone seeking to keep pace with the changing environment of PSoC.

TABLE OF CONTENTS

GETTING STARTED

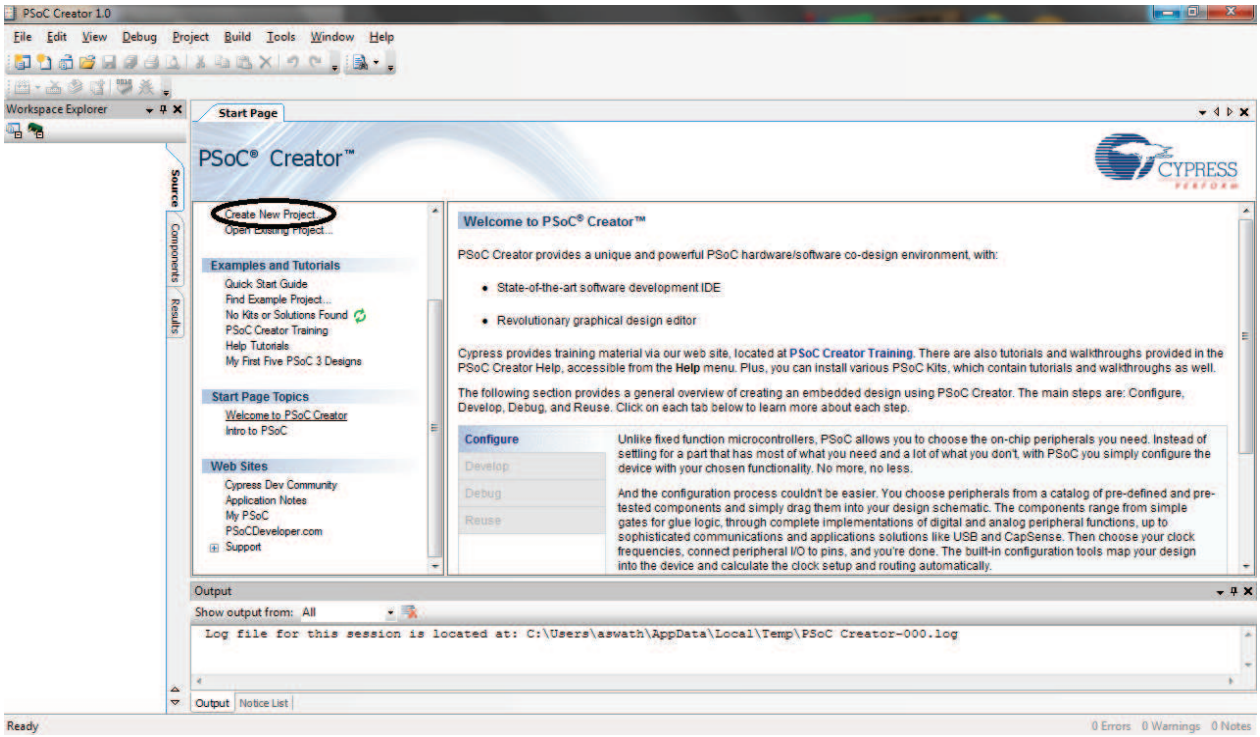
OBJECTIVE

For those of you who've used PSoC 1 and are familiar with the PSoC Express/PSoC Designer environments, the PSoC creator is not hard to grasp at all. It has the versatility of the PSoC Designer coupled with the easy usability of the PSoC Express. Additionally, interconnecting modules is far easier than before.

For all those of you getting started with PSoC, you could not have an easier software interface than the PSoC Creator.

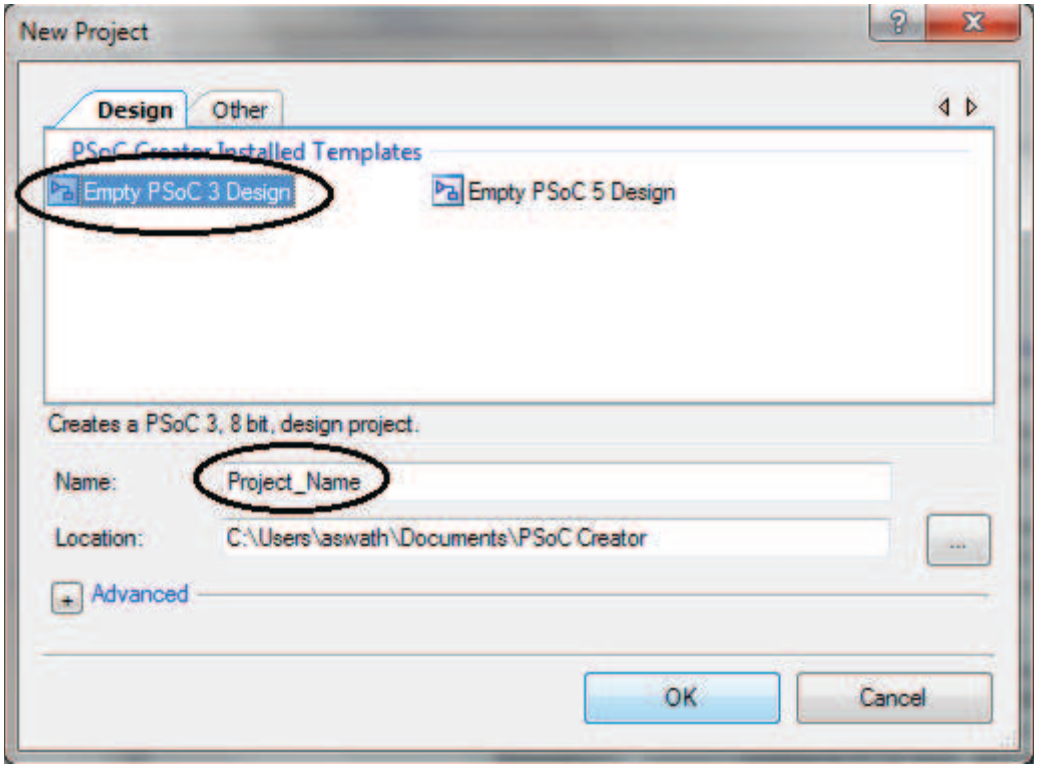
The objective of this exercise is to get the user familiar with rudimentary aspects of PSoC Creator. All the exercises in this lab book have been designed, built and tested using PSoC Creator 1, and programmed using PSoC Programmer 3.12.3.

CREATING A NEW PROJECT



- The PSoC Creator Start Page has a link that enables one to create a new project as shown in the figure above.
- Alternatively, a new project can also be created by going to File > New>Project.

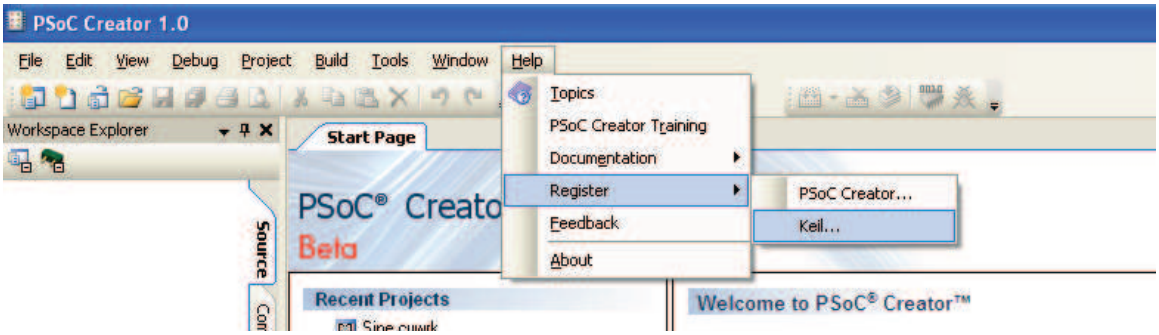
NAMING THE PROJECT



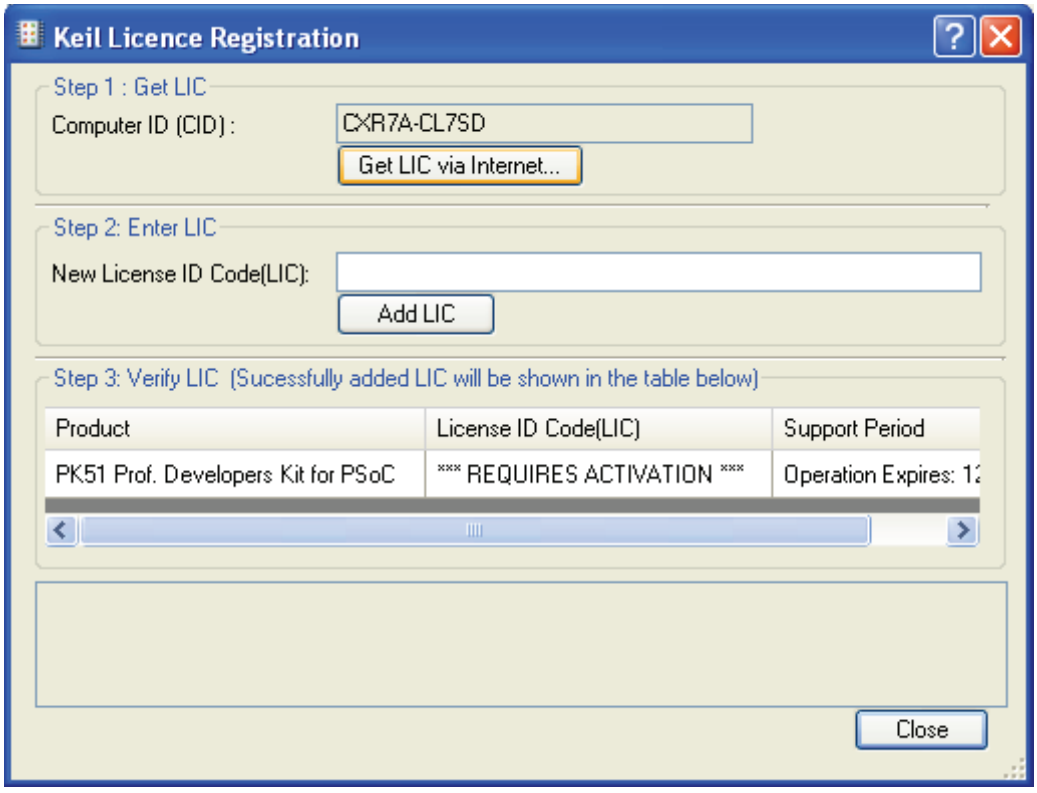
- Once the new project is created, a window appears like the one shown above.
- Select Empty PSoC 3 Design and Assign your project a name.

LICENSING KEIL

- Before beginning with the design, it is necessary to license Keil without which you will be unable to compile your project.
- Go to Help>Register>Keil as shown below

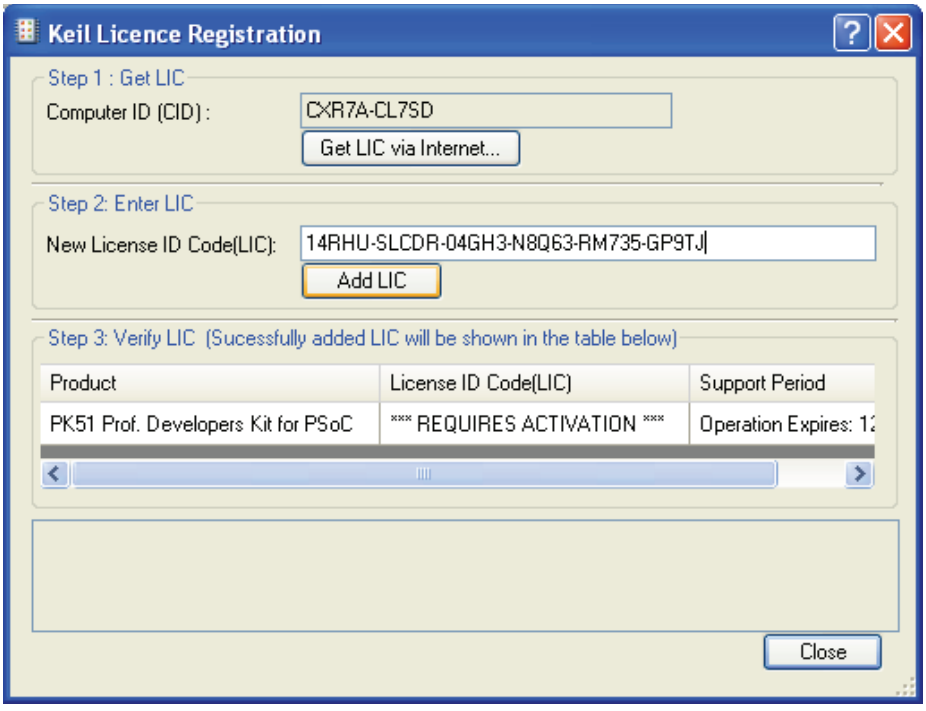


- Select "Get LIC via Internet" as shown below



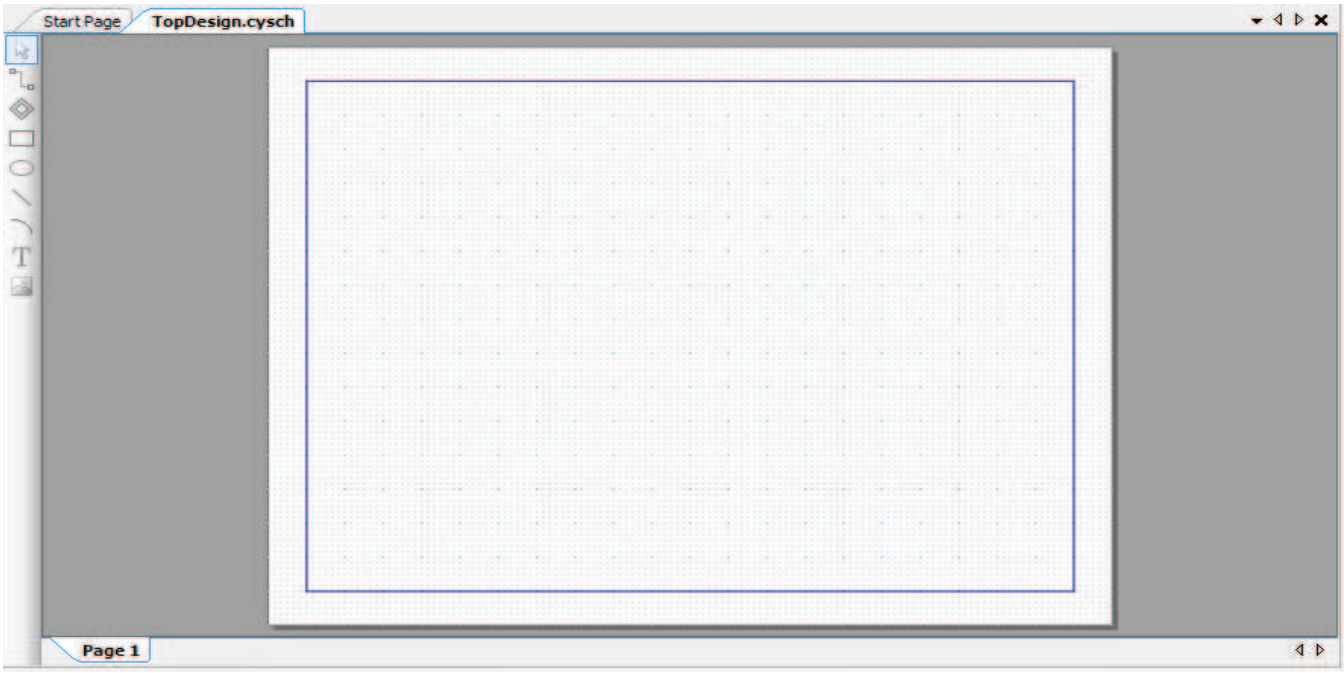
- Fill in all the necessary fields in the Product Licensing form from ARM
- You should see a confirmation page as shown below.

- Check your email for the License file. Copy the License ID Code (LIC) to the window.



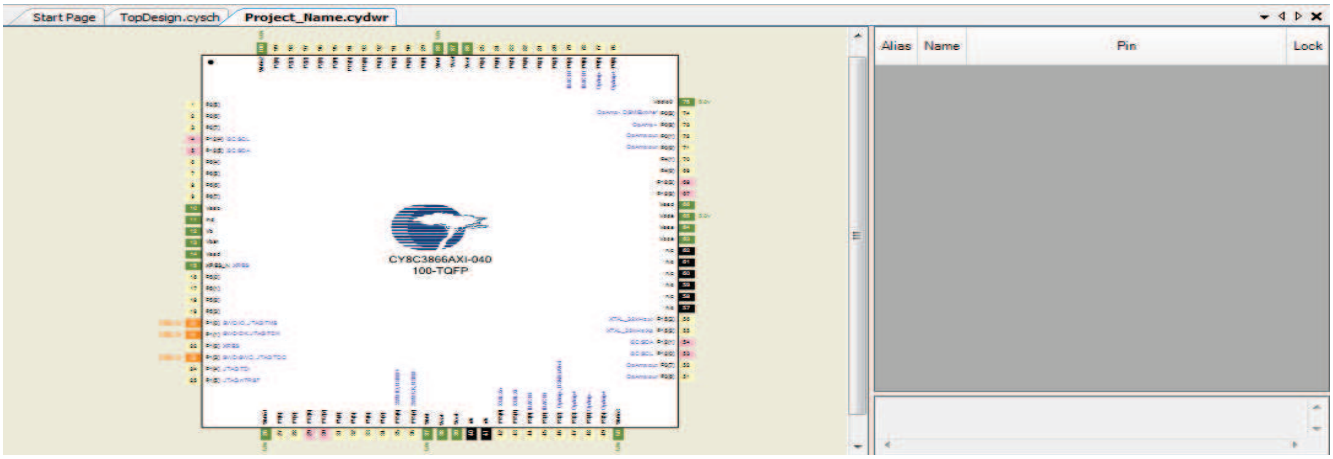
THE SCHEMATIC PAGE

- The schematic page as shown below is the first element of the project. This is where modules of the PSoC are added and interconnected just like drawing a schematic.
- This can be accessed by double clicking TopDesign.cysch in the workspace explorer.



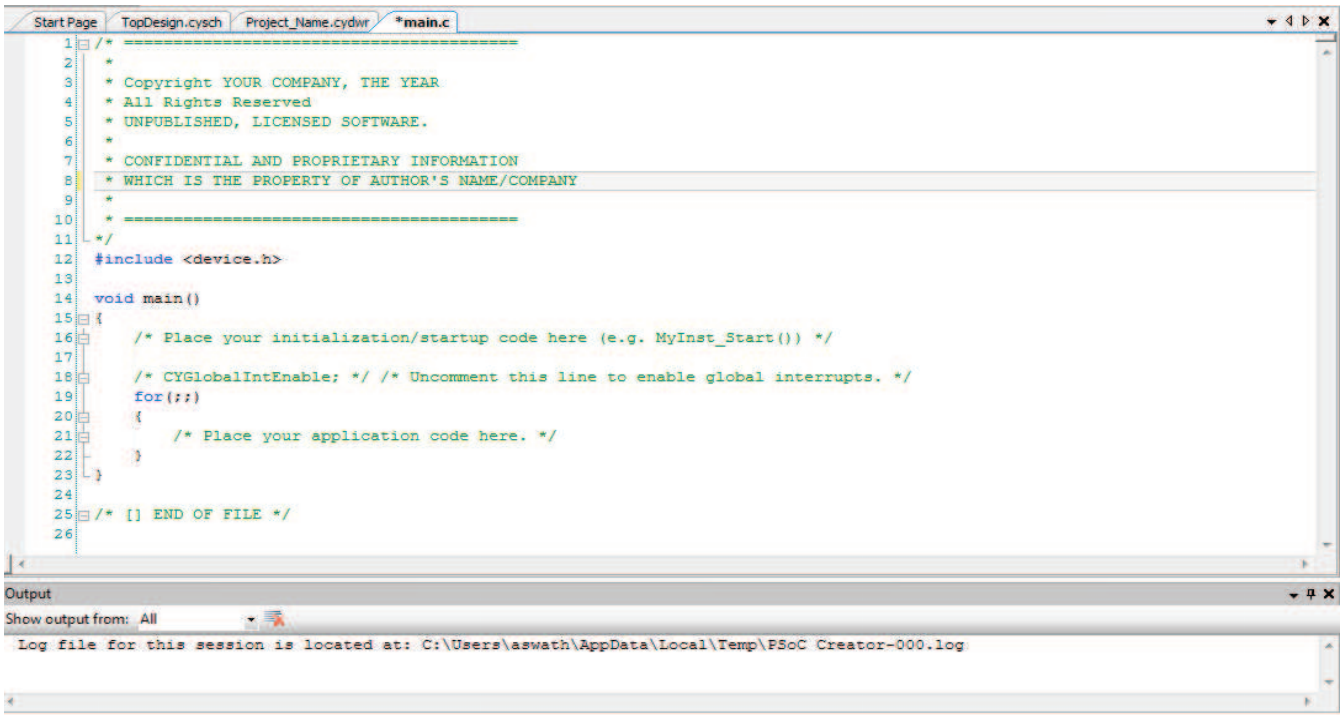
THE PIN ASSIGNMENT PAGE

- The pin assignment page as shown below enables the user to assign physical device pins to the project, configure clocks, interrupts etc.
- It can be accessed by double clicking Project_Name.cydwr



WRITING FIRMWARE

- Most PSoC 3 kits come with an inbuilt flash memory which can be used to store data.
- Writing firmware enables the user to activate modules and use a variety of pre defined functions for each module.
- This window as shown below can be accessed by double clicking the filename in the workspace for example "main.c"



At the end of this exercise you should be able to

- Create a New Project
- License Keil
- Access the schematic page, pin assignment page and the firmware page.

CPU EXERCISE

CPU EXERCISE

OBJECTIVE

Perform a CPU exercise that will help you understand the 8051 core of the PSoC 3

REQUIRED MATERIALS

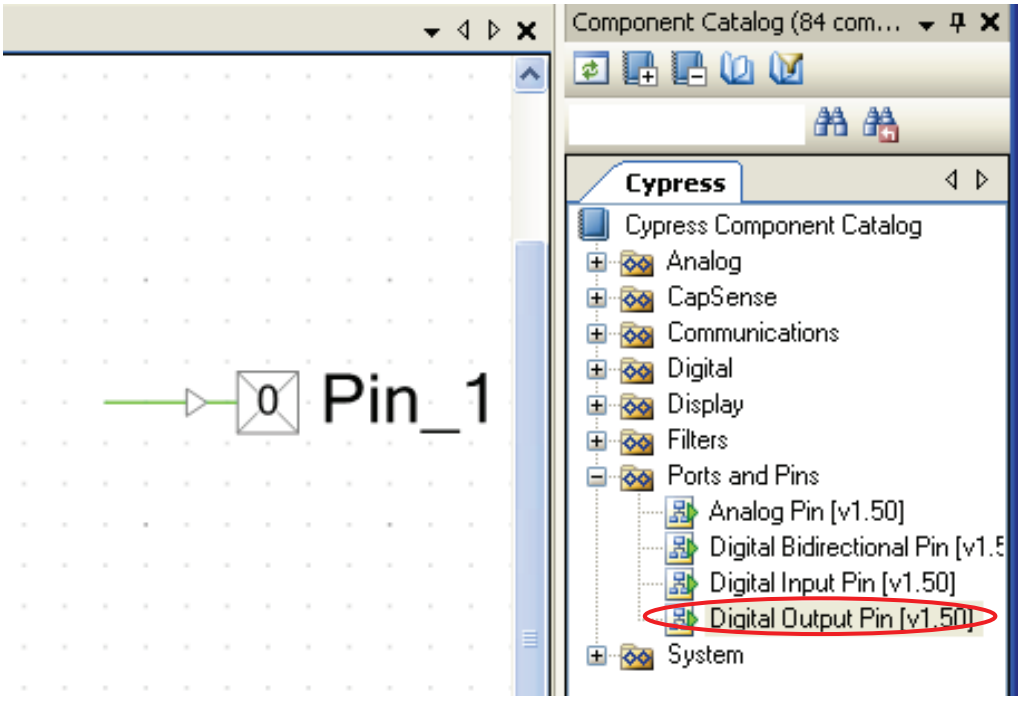
1. PSoC 3 (001) board.

OVERVIEW

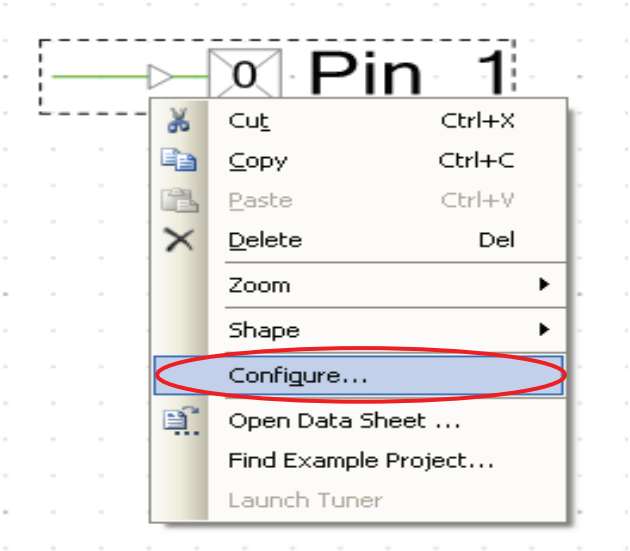
The new enhanced feature of the PSoC 3 that makes it better than the PSoC 1 is the fact that a new 8051 core is used. In order to understand the brain of the PSoC 3 we perform the following CPU exercise where we use #pragma statements.

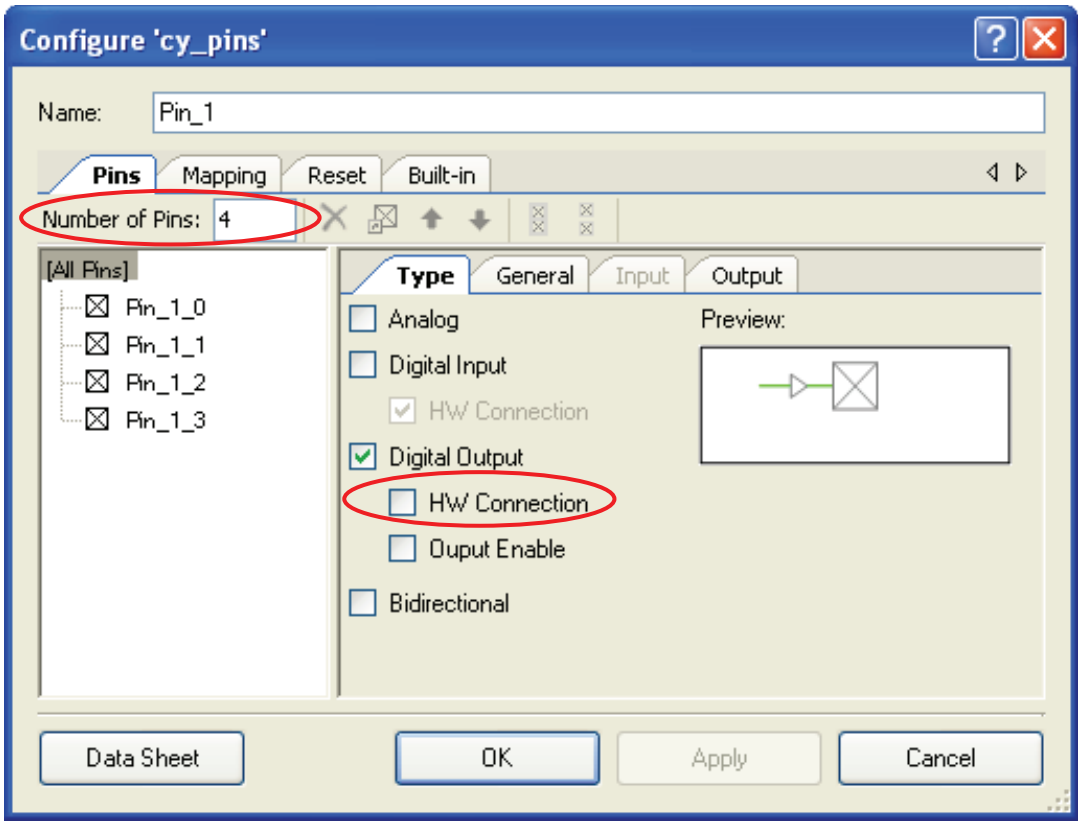
HARDWARE CONFIGURATIONS

- Open a new project and name is LAB1A
- Make sure that the window "TopDesign.cysch " is open
- From Component Catalog->Ports and Pins, select Digital Output Pin. Drag and Drop it in the grid.

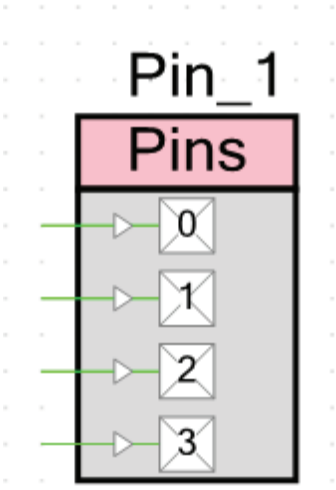


- Double Click (or right click and select Configure) on **Pin_1**. Uncheck the HW Connection Box. Select Number of Pins: **4**.



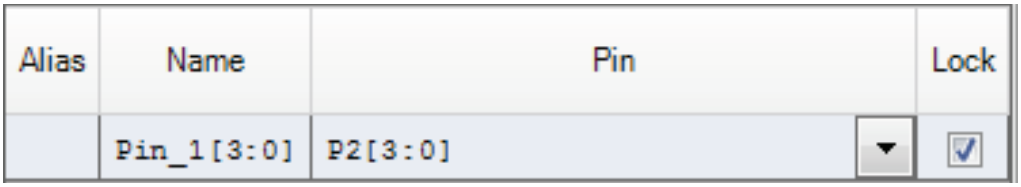


- The change in number of pins will be reflected in the schematic as shown below.



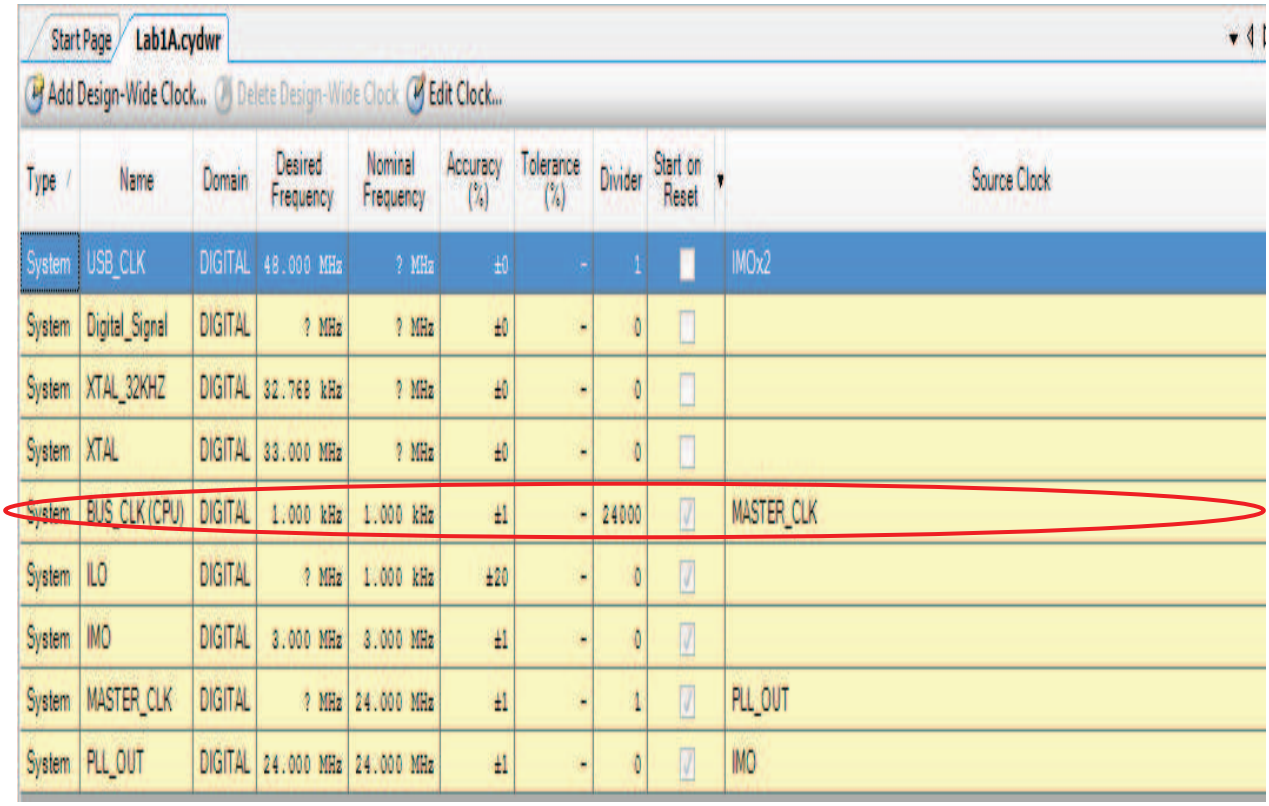
ASSIGNING PINS AND SETTING CLOCK

- Go to Lab1A.cydwr->Pins from the Workspace Explorer. On the right hand side of the window in front of Pin_1[3:0], select P2[3:0]. (P2[3:0] correspond to 4 LED's on the PSoC3-003 Kit). A screenshot of the same is shown below.



Alias	Name	Pin	Lock
	Pin_1[3:0]	P2[3:0]	☑

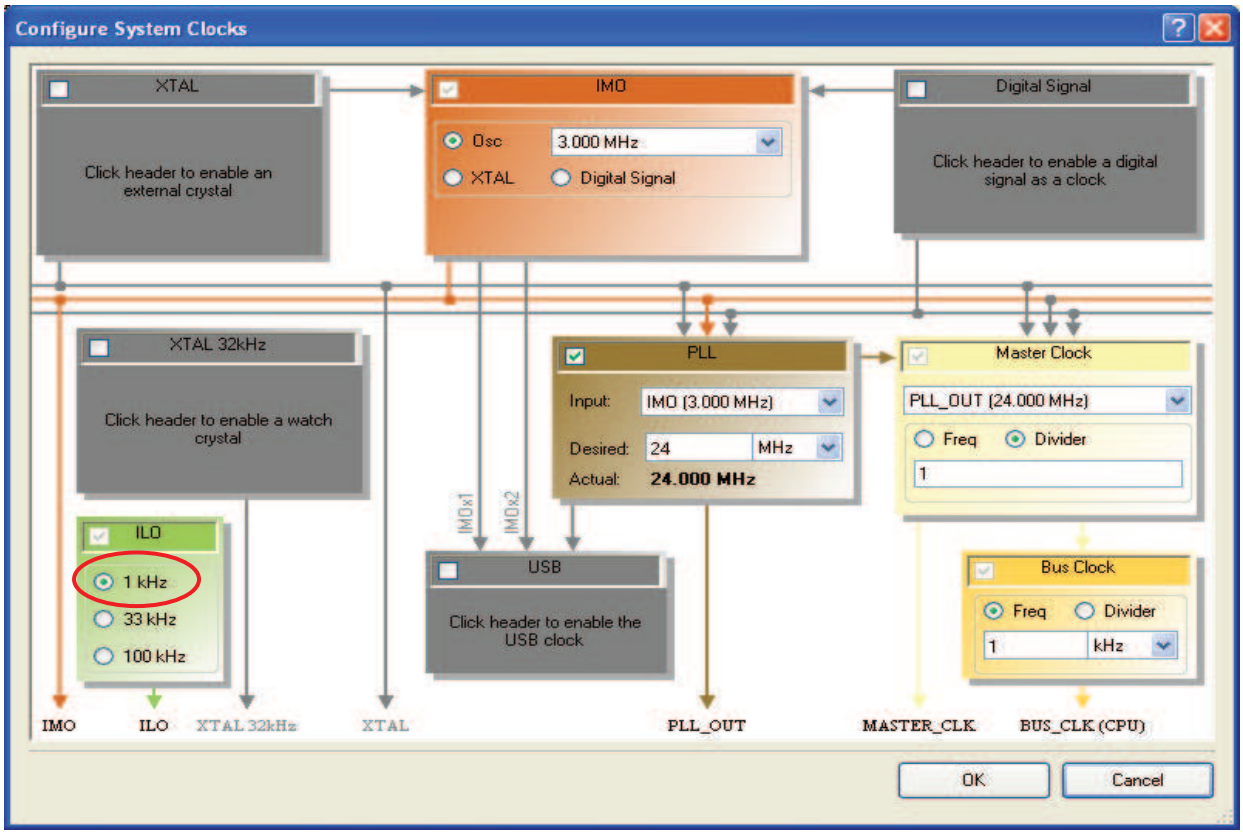
- Go to Lab1A.cydwr->Clocks. Double Click Bus_Clk. Set the Bus Clock to 1kHz as shown below. This will be the CPU_Clk.



Start Page | Lab1A.cydwr

Add Design-Wide Clock... | Delete Design-Wide Clock | Edit Clock...

Type	Name	Domain	Desired Frequency	Nominal Frequency	Accuracy (%)	Tolerance (%)	Divider	Start on Reset	Source Clock
System	USB_CLK	DIGITAL	48.000 MHz	? MHz	±0	-	1	☐	IMOx2
System	Digital_Signal	DIGITAL	? MHz	? MHz	±0	-	0	☐	
System	XTAL_32KHZ	DIGITAL	32.768 kHz	? MHz	±0	-	0	☐	
System	XTAL	DIGITAL	33.000 MHz	? MHz	±0	-	0	☐	
System	BUS_CLK (CPU)	DIGITAL	1.000 kHz	1.000 kHz	±1	-	24000	☑	MASTER_CLK
System	ILO	DIGITAL	? MHz	1.000 kHz	±20	-	0	☑	
System	IMO	DIGITAL	3.000 MHz	3.000 MHz	±1	-	0	☑	
System	MASTER_CLK	DIGITAL	? MHz	24.000 MHz	±1	-	1	☑	PLL_OUT
System	PLL_OUT	DIGITAL	24.000 MHz	24.000 MHz	±1	-	0	☑	IMO


Configure System Clocks
XTAL
Click header to enable an external crystal
IMO
Osc
3.000 MHz
XTAL
Digital Signal
Digital Signal
Click header to enable a digital signal as a clock
XTAL 32kHz
Click header to enable a watch crystal
ILO
1 kHz
33 kHz
100 kHz
IMOx1
IMOx2
USB
Click header to enable the USB clock
PLL
Input:
IMO (3.000 MHz)
Desired:
24
MHz
Actual:
24.000 MHz
Master Clock
PLL_OUT (24.000 MHz)
Freq
Divider
1
Bus Clock
Freq
Divider
1
kHz
IMO
ILO
XTAL 32kHz
XTAL
PLL_OUT
MASTER_CLK
BUS_CLK (CPU)
OK
Cancel

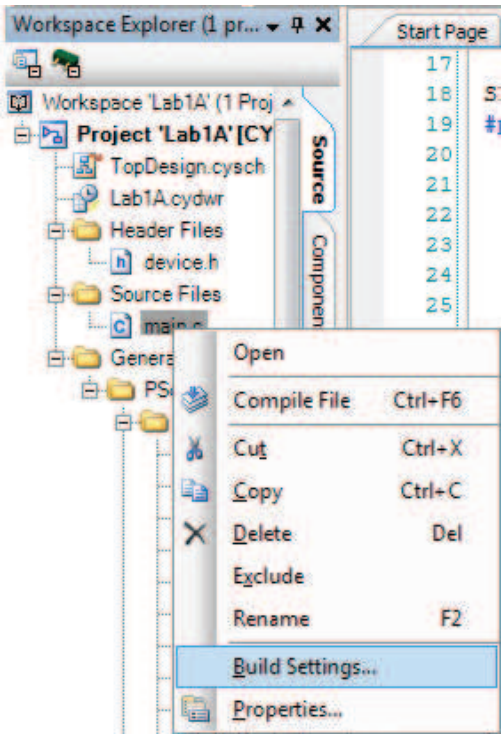
WRITING FIRMWARE

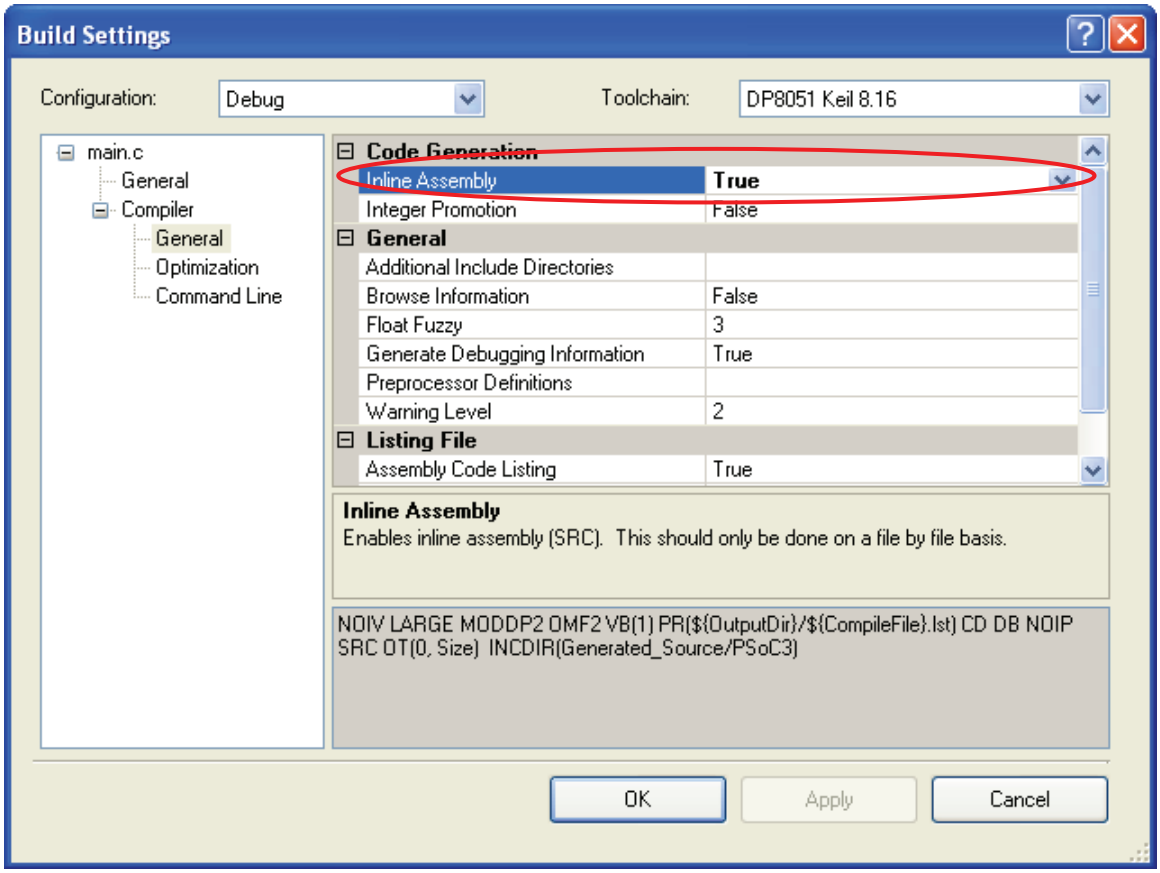
- Open main.c from Workspace Explorer
- Insert the following code in main()

```
SFRPRT2SEL |= 0xFF;                          //enables the control of the port
#pragma asm

loop:
mov A, SFRPRT2PS        ; 1 Clock Reads the value of port2
inc A                   ; 1 Clock Adds 1
mov SFRPRT2DR,A         ; 1 Clock Writes to port2
;
REPT 992            ; delay for 992 clock cycles
NOP                 ; 1*992=992 Clock Cycles
ENDM
;
jmp loop                ; 5 clock

#pragma endasm

/* the clock frequency is set to 1kHz. The above code is an infinite loop. Every
cycle takes 1000 cpu clocks to complete i.e 1sec
The initial value of PRT2 is 0. The LED's are Sinking on PSoC3-003 kit, therefore
the state of LED will be NOT of the value of the Pin.*/
```

- Allow in-line assembly code in main.c
- Right click main.c and select build settings.
- Under Compiler select Inline Assembly as **True**

The figures below illustrate the same.


Workspace Explorer (1 pr...
Start Page
Workspace 'Lab1A' (1 Proj
Project 'Lab1A' [CY
TopDesign.cysch
Lab1A.cydwr
Header Files
device.h
Source Files
Source
Open
Compile File Ctrl+F6
Cut Ctrl+X
Copy Ctrl+C
Delete Del
Exclude
Rename F2
Build Settings...
Properties...


Build Settings
Configuration: Debug
Toolchain: DP8051 Keil 8.16
main.c
General
Compiler
General
Optimization
Command Line
Code Generation
Inline Assembly True
Integer Promotion False
General
Additional Include Directories
Browse Information False
Float Fuzzy 3
Generate Debugging Information True
Preprocessor Definitions
Warning Level 2
Listing File
Assembly Code Listing True
Inline Assembly
Enables inline assembly (SRC). This should only be done on a file by file basis.
NOIV LARGE MODDP2 OMF2 VB(1) PR(${OutputDir}/${CompileFile}.lst) CD DB NOIP
SRC OT(0, Size) INCDIR(Generated_Source/PSoC3)
OK
Apply
Cancel

PROGRAMMING THE PSoC

- Connect the PSoC3-003 kit to the computer using the USB cable. Burn the code by pressing Ctrl+F5 or the Program button. 
- Reconnect the PSoC3-003 kit to the computer or connect to battery separately.

Pass	LED4(P2_3)	LED3(P2_2)	LED2(P2_1)	LED1(P2_0)
Start	0	0	0	0
1st	0	0	0	1
2nd	0	0	1	0
3rd	0	0	1	1
4th	0	1	0	0
5th	0	1	0	1
6th	0	1	1	0
7th	0	1	1	1
8th	1	0	0	0
9th	1	0	0	1
10th	1	0	1	0
11th	1	0	1	1
12th	1	1	0	0
13th	1	1	0	1
14th	1	1	1	0
15th	1	1	1	1
16th	0	0	0	0
17th	0	0	0	1
18th	0	0	1	0
19th	0	0	1	1
20th	0	1	0	0

TESTING YOUR DESIGN

- You can adjust the output frequency at pins by adjusting the CPU_Clk and the number of NOP commands.

GPIO DRIVE MODES

GPIO DRIVE MODE-ANALOG

OBJECTIVE

To get familiar with the general purpose I/O analog drive mode.

REQUIRED MATERIALS

1. PSoC 3 (001) board
2. Cathode Ray Oscilloscope (CRO).

RELATED REFERENCE MATERIAL

1. Schematic of the 001 board

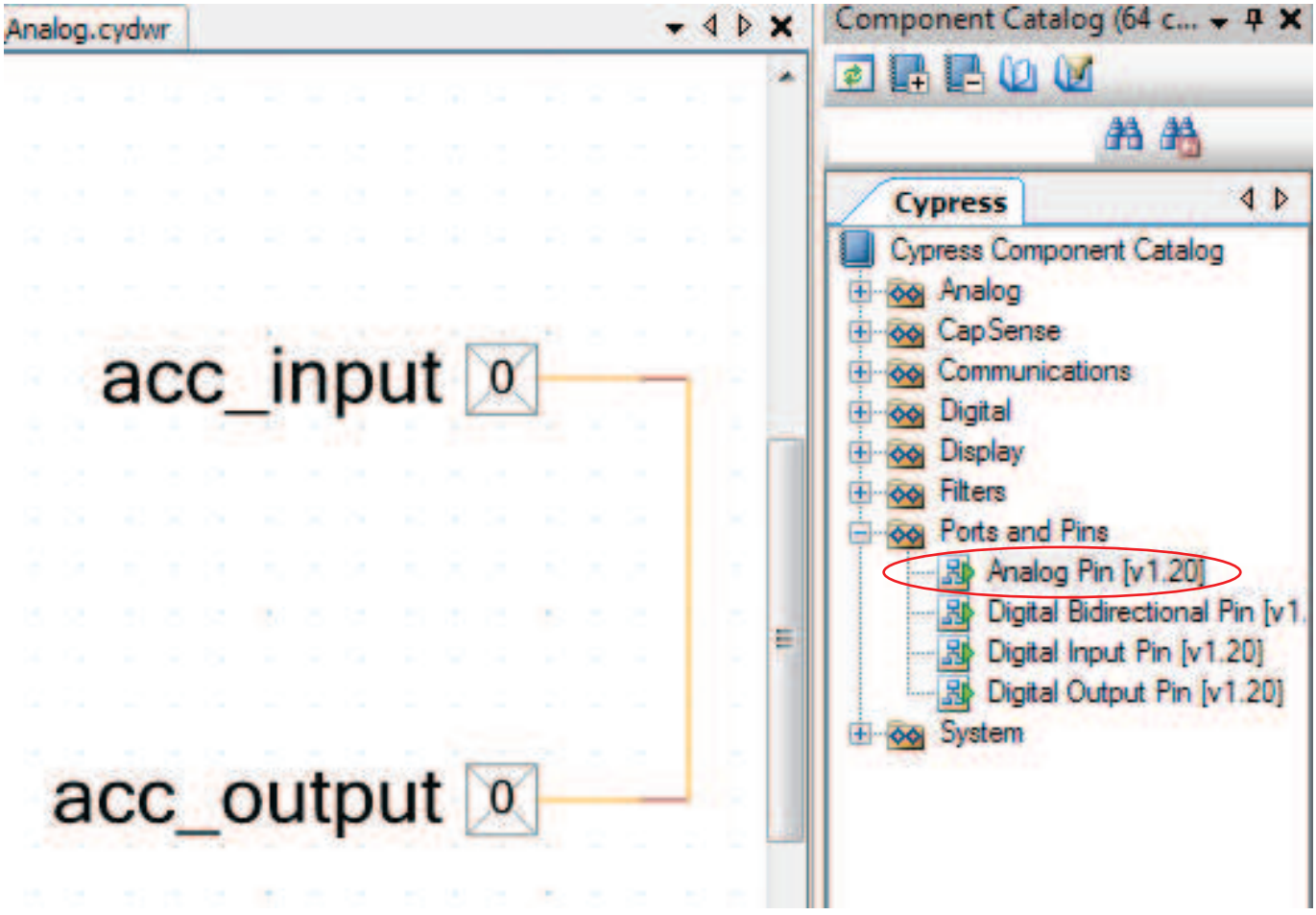
OVERVIEW

Input –output pins commonly known as I/O are the doors by which the PSoC sends and receives information from the external world. I/O can be of many types but the PSoC 3 uses General Purpose I/O or GPIO which can be used for all purposes. These GPIOs have different drive modes. In this exercise we examine the analog drive mode where we use an accelerometer to provide the analog input.

HARDWARE CONFIGURATIONS

- Open PSoC Creator and start "Create new project".

- Select "Empty PSoC3 Design". Name the Project "Lab_DM_Analog"
- Make sure that the window "TopDesign.cysch " is open.
- From Component Catalog->Ports and Pins, select Analog Pin. Drag and Drop it in the grid. Make a copy of it. Connect them using the Wire Tool. Name them as
  - acc_input
  - acc_output



ASSIGNING PINS

- Go to Lab_DM_Analog.cydwr->Pins from the Workspace Explorer. On the right hand side of the window in front of
  - acc_input, select P3[5]. (P3[5] is x-input from the accelerometer on PSoC3-003 kit)
  - acc_output, select P6[0]. (P6[0] is a free pin on PSoC3-003 kit.)

Alias	Name	Pin
	acc_input	P3[5]
	acc_output	P6[0]

- Press Shift+F6 or the Build Button to Build the hex file for the project. 

PROGRAMMING THE PSoC

- Connect the PSoC3-003 kit to the computer using the USB cable. Burn the code by pressing Ctrl+F5 or the Program button. 

TESTING YOUR DESIGN

- Reconnect the PSoC3-003 kit to the computer or connect to battery separately.
- Connect the CRO to P3[5].
- Rotate the kit along the x-axis and observe the output on the CRO.

GPIO DRIVE MODES

GPIO DRIVE MODE-STRONG

OBJECTIVE

To get familiar with the general purpose I/O strong drive mode.

REQUIRED MATERIALS

1. PSoC 3 (001) board

RELATED REFERENCE MATERIAL

1. Schematic of the 001 board

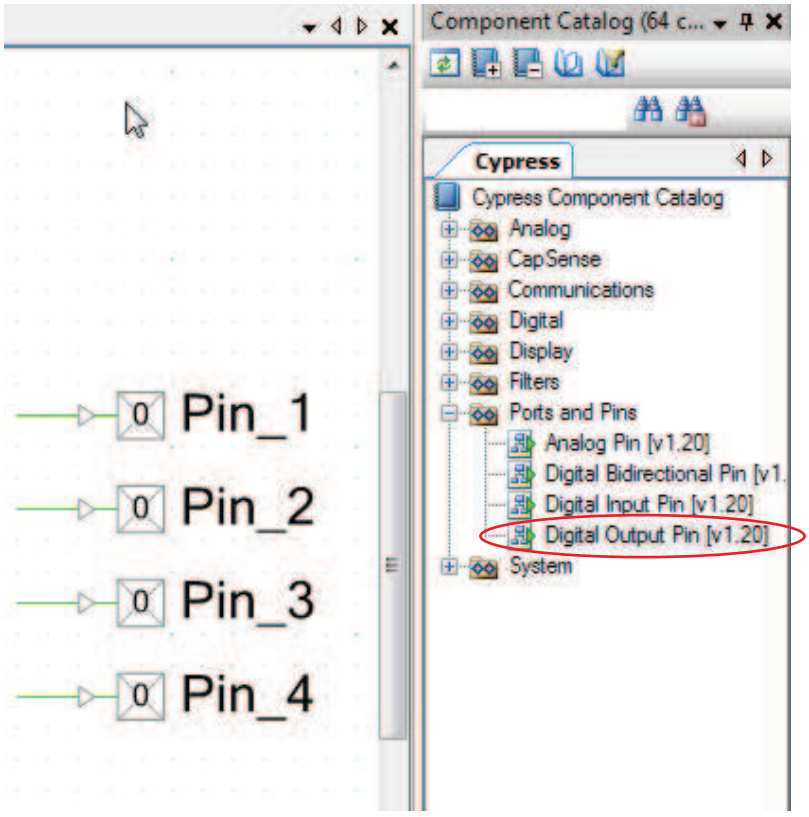
OVERVIEW

Input –output pins commonly known as I/O are the doors by which the PSoC sends and receives information from the external world. I/O can be of many types but the PSoC 3 uses General Purpose I/O or GPIO which can be used for all purposes. These GPIOs have different drive modes. In this exercise we examine the strong drive mode where we use I/O's to drive LEDs. The strong mode is asserted through software.

HARDWARE CONFIGURATIONS

- Open PSoC Creator and start "Create new project".

- Name the Project "Lab_DM_Strong"
- Make sure that the window "TopDesign.cysch " is open.
- From Component Catalog->Ports and Pins, select Digital Output Pin. Drag and Drop it in the grid. Replicate it 4 times.
- Double Click each pin unselect the HW Connection Box.



ASSIGNING PINS

- Go to Lab_DM_Strong.cydwr->Pins from the Workspace Explorer.
- Assign pins as shown in the screenshot below.

Alias	Name	Pin
	Pin_1	P2[0]
	Pin_2	P2[2]
	Pin_3	P2[3]
	Pin_4	P4[0]

WRITING FIRMWARE

- Open main.c from the workspace explorer. Paste the following code

```
/* ========================================
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/

#include <device.h>
void main()
{
//
SFRPRT4SEL |= 0x01; //enable SFR access for P4[0]

Pin_1_SetDriveMode(Pin_1_DM_STRONG);//Change DM of P2[0] to Strong
Pin_2_SetDriveMode(Pin_2_DM_STRONG);//Change DM of P2[2] to Strong

CyPins_SetPinDriveMode(Pin_3_0,PIN_DM_STRONG); //Change DM of P2[3] to
Strong//
CyPins_SetPinDriveMode(Pin_4_0,PIN_DM_STRONG); //Change DM of P4[0] to
Strong
//
for(;;)
{
/*
There are 3 ways to access GPIO's in Creator
a:Using Per-Pin API i.e. CyPins        ///Slowest///

b:Using Component APIs i.e Pin_x (Using Pin_x_DR or Pin_x_Write)

c:Using SFR register space i.e SFRPRT  ///Fastest///

In the following code each pin is toggled.
*/
CyDelay(500); //500ms delay
if (!CyPins_ReadPin(Pin_1_0)) CyPins_SetPin(Pin_1_0);
else CyPins_ClearPin(Pin_1_0);

CyDelay(500);
Pin_2_Write(!(Pin_2_Read()));
CyDelay(500);
if(Pin_3_PS==Pin_3_MASK) Pin_3_DR &= ~Pin_3_MASK;
else Pin_3_DR |= Pin_3_MASK;
CyDelay(500);
SFRPRT4DR = (!(SFRPRT4PS));

}
}
```

PROGRAMMING THE PSoC

- Press Shift+F6 or the Build Button to Build the hex file for the project. 
- Connect the PSoC3-003 kit to the computer using the USB cable. Burn the code by pressing Ctrl+F5 or the Program button. 

TESTING YOUR DESIGN

- Reconnect the PSoC3-003 kit to the computer or connect to battery separately and observe the LEDs.

INTERRUPTS

INTERRUPT EXERCISE

OBJECTIVE

To become familiar with the interrupts of PSoC 3 and learn how to use them.

REQUIRED MATERIALS

1. PSoC 3 (001) board

RELATED REFERENCE MATERIAL

1. Control Register Component Data Sheet
2. Sleep Timer Component Data Sheet.

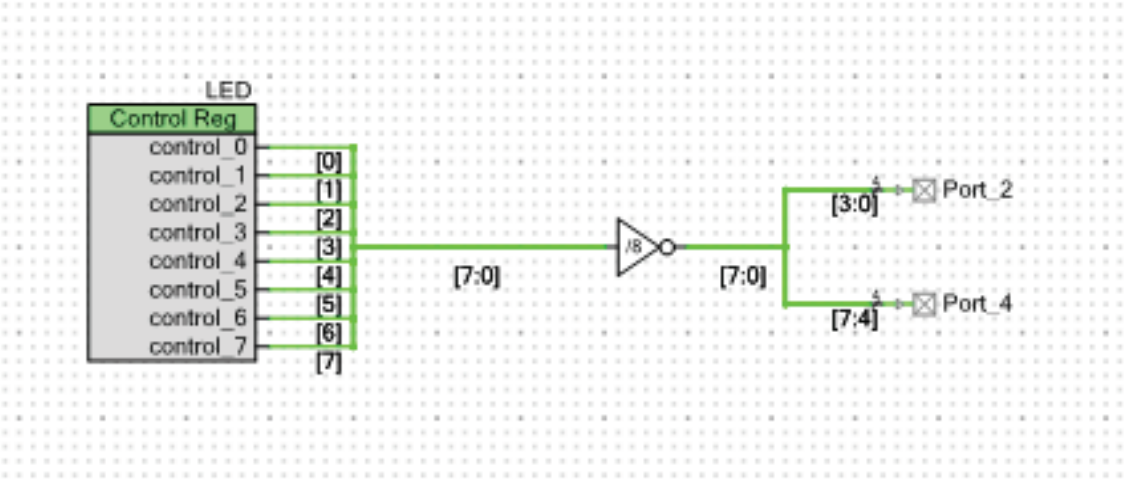
OVERVIEW

Polling may not be the most efficient technique of performing various operations on a PSoC 3. Interrupts come in handy when one doesn't know when an event is likely to occur. This exercise focuses on interrupts and how to use them on a PSoc 3. We use a sleep timer to generate the interrupts.

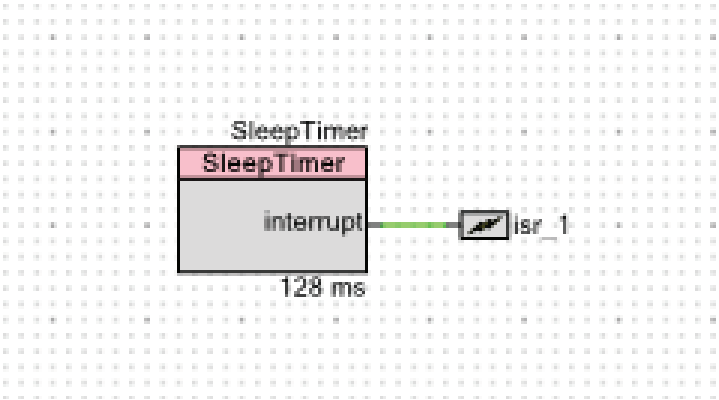
HARDWARE CONFIGURATIONS

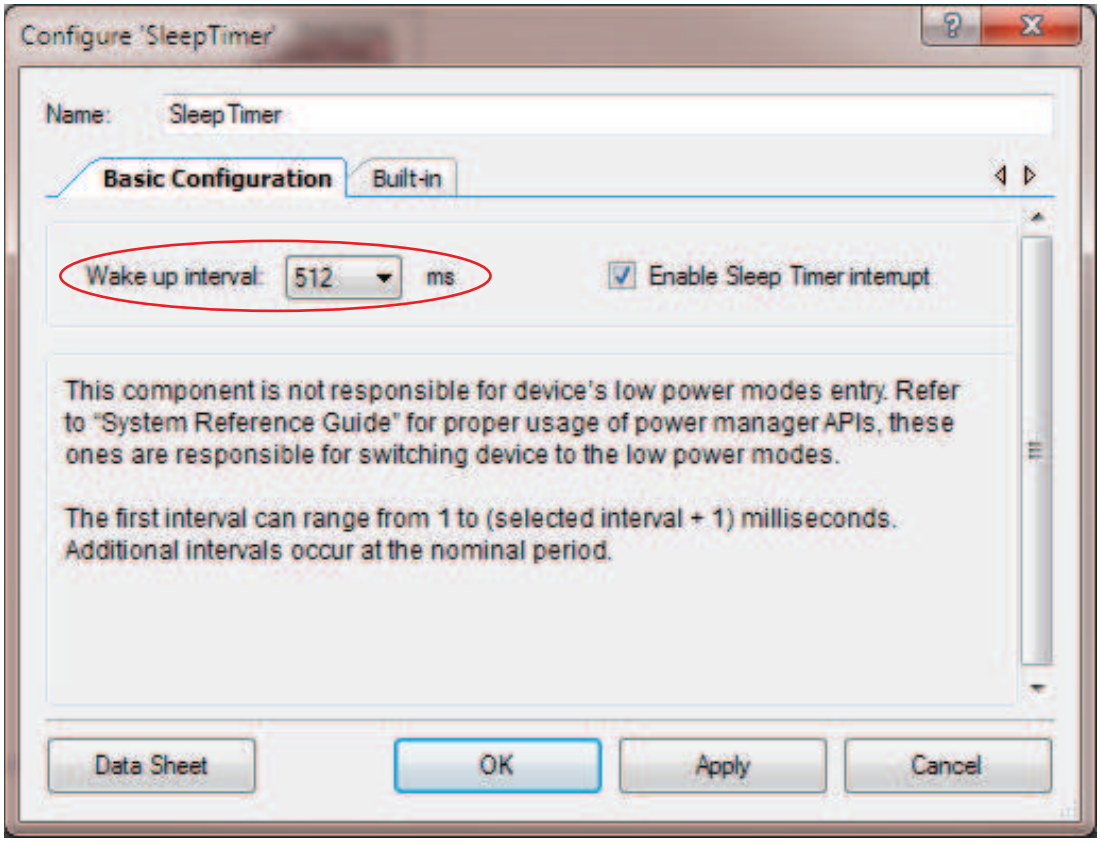
- Open PSoC Creator and start "Create new project".

- Name the Project "lab_interrupt".
- Make sure that the window "TopDesign.cysch " is open.
- From Component Catalog->Ports and Pins, select two Digital Output Pin.blocks. Drag and Drop them in the grid. Name them Port2 and Port4.
- Insert a NOT gate from Digital->Logic. Make its width 8.
- Insert a Control Register block from Digital->Register. Name it LED.
    - Step(4-7): Follow steps as given in http://www.planetpsoc.com/psoc3-videos-how-to-videos/78-connecting-a-multiple-wired-bus-in-psoc-creator.html

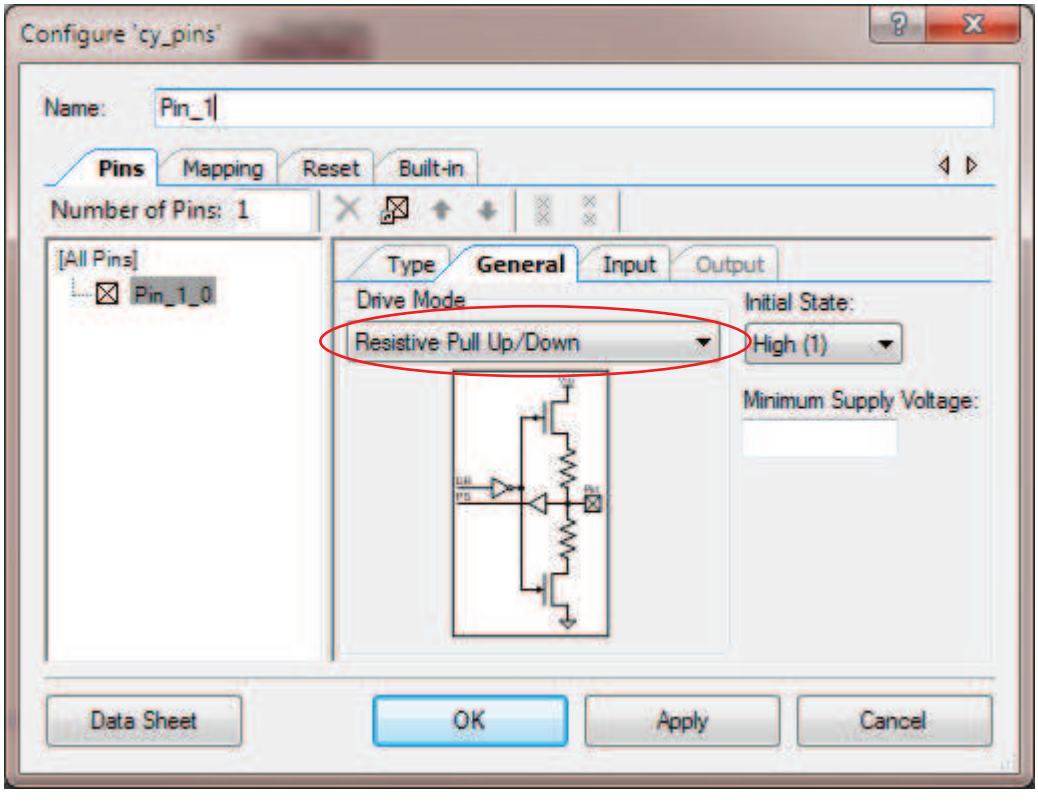


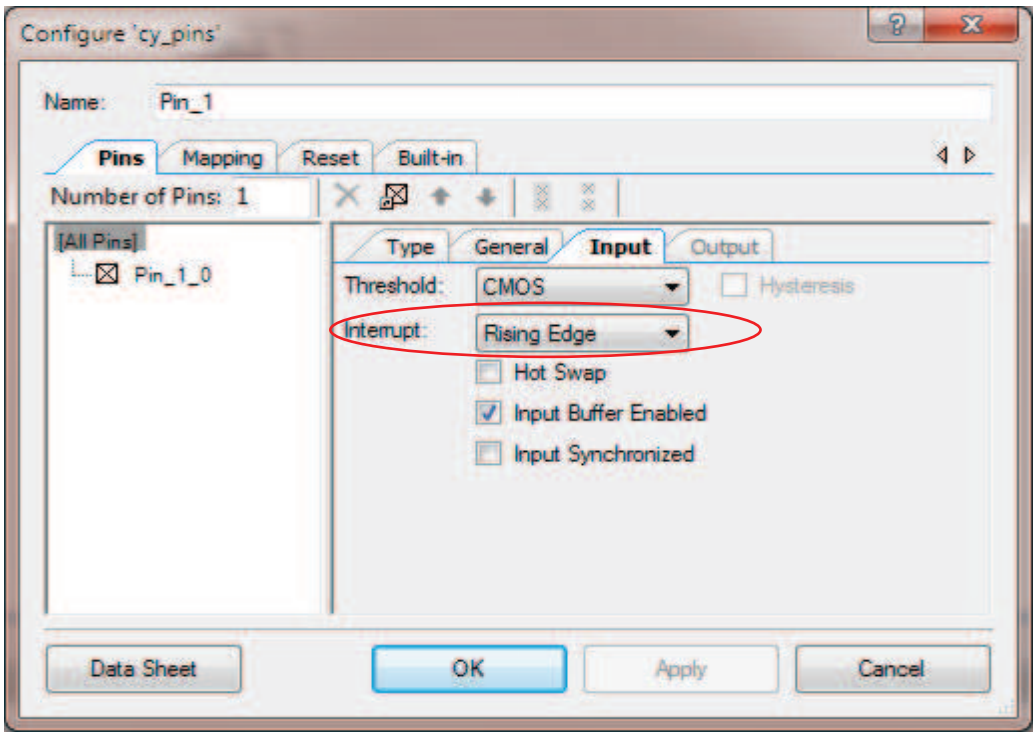
- Insert a SleepTimer from System>Sleep Timer. Name the sleep timer as "SleepTimer" and let the isr be "isr_1". Double click on the component and set the WakeUpInterval to 512ms.





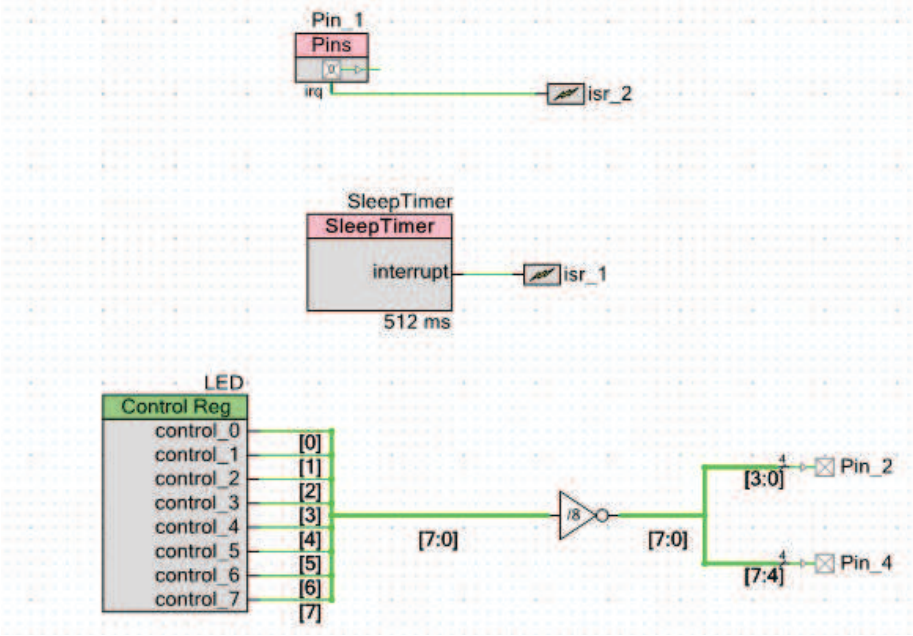
- Insert a Digital Input Pin. Set the Drive Mode as Resistive PullUp/Down. Set Interrupt as Rising Edge as shown in the two screenshots below.





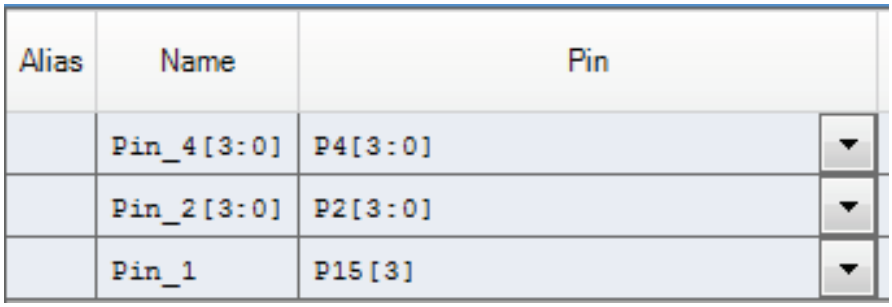
- Insert a interrupt block from System>Interrupt and name it as "isr_2". Connect it to Pin_1 as follows.

- The complete schematic should look as shown in the screenshot below.



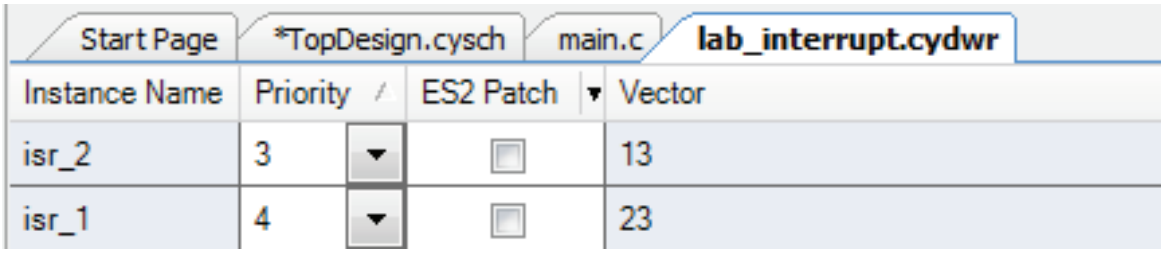
ASSIGNING PINS

- Open lab_interrupt.cydwr->Pins.
- Set the pin configurations as shown in the figure below



Alias	Name	Pin
	Pin_4[3:0]	P4[3:0]
	Pin_2[3:0]	P2[3:0]
	Pin_1	P15[3]

- Open lab_interrupt.cydwr->Interrupts.
- Set the isr priorities as shown in the figure below.



WRITING FIRMWARE

- Open main.c from Workspace Explorer
- Insert the following code in main.c

```
/* ========================================
 *
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/

#include <device.h>
uint8 i=0;

CY_ISR(sleep)
{
      LED_Write(i++);
      SleepTimer_GetStatus();

}

CY_ISR(button)
{
      i=0x05;
      LED_Write(i);
      Pin_1_ClearInterrupt();
}



void main()
{
            CYGlobalIntDisable;

            isr_1_Start();
            isr_1_SetVector(sleep);
            isr_2_Start();
            isr_2_SetVector(button);
                        SleepTimer_EnableInt();

            CYGlobalIntEnable;

            SleepTimer_Start();


}
```

PROGRAMMING THE PSoC

- Press Shift+F6 or the Build Button to Build the hex file for the project.
- Connect the PSoC3-003 kit to the computer using the USB cable. Burn the code by pressing Ctrl+F5 or the Program button. 

TESTING YOUR DESIGN

- Reset the kit. The LED's will show the current status of "I". If the button is pressed then I will be set to 0x05 and the count will begin from there.

LCD

DISPLAY USING AN LCD

OBJECTIVE

To learn interfacing an LCD with PSoC 3 and display on it.

REQUIRED MATERIALS

1. PSoC 3 (001) board, character LCD (14 pin).

RELATED REFERENCE MATERIAL

1. Character LCD Component Data Sheet

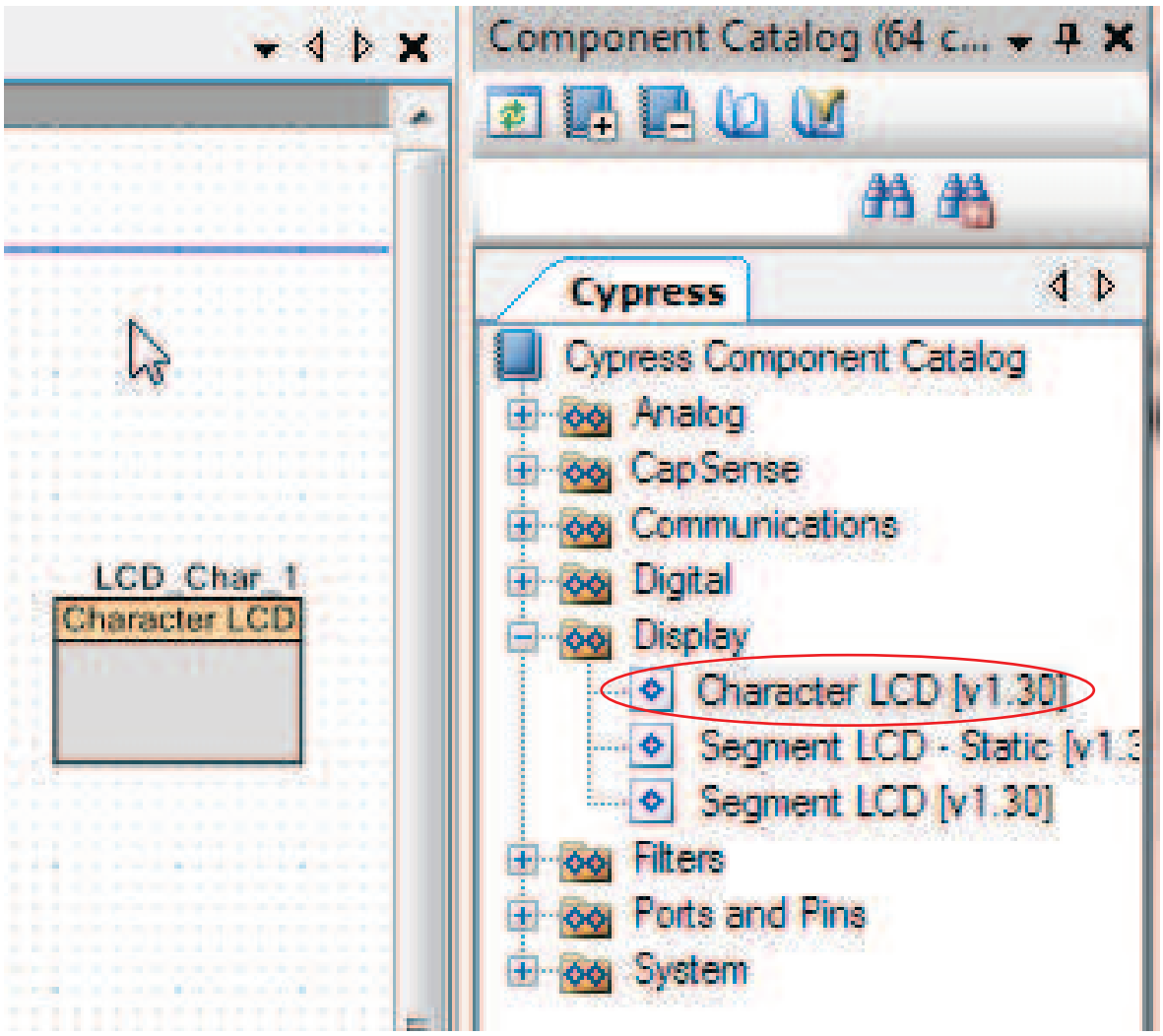
OVERVIEW

LCD's are fundamentally the easiest form of character and bar-graph display. LCDs come in many forms but the most common ones continue to be the DOT MATRIX LCDs. Here, we learn how to interface an LCD with a PSoC 3 and display text on it. The LCD we use is a 16 pin LCD from Hitachi which comes along with an eval(3210) kit or a DVK board(003).

HARDWARE CONFIGURATIONS

- Open PSoC Creator and start "Create new project".

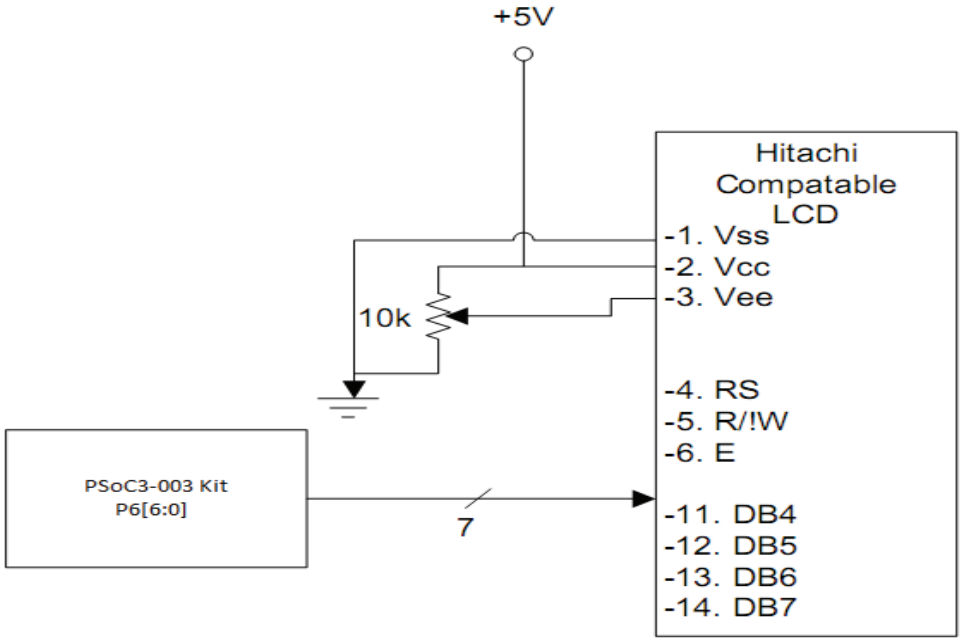
- Name the Project "LAB_ LCD".
- Make sure that the window "TopDesign.cysch " is open.
- From Component Catalog->Display, select Character LCD. Drag and Drop it in the grid.



ASSIGNING PINS

- Go to LAB_ LCD.cydwr->Pins from the Workspace Explorer. On the right hand side of the window in front of LCD_CHAR_1[6:0], select P6[6:0]. (P6[6:0] correspond to 7 GPIO pins on the PSoC3-003 Kit).
- Connect the Hitachi LCD to the kit according to the following pin out.

Port Pin	LCD Component Pin	Description
P6_0	DB4	Data Bit 0
P6_1	DB5	Data Bit 1
P6_2	DB6	Data Bit 2
P6_3	DB7	Data Bit 3
P6_4	E	LCD Enable
P6_5	RS	Register Select
P6_6	R/!W	Read/not Write



Alias	Name	Pin	Loc
	\LCD_Char_1:LCDPort\[6:0]	P6[6:0]	

WRITING FIRMWARE

- Open main.c from Workspace Explorer. Insert the following code.

```
/* ========================================
 *
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/

#include <device.h>
#include "LCD_Char_1.h"

uint8 i=0;

void main()
{
    LCD_Char_1_Start();
    LCD_Char_1_Position(0, 0);                      //Positions the cursor
    LCD_Char_1_PrintString("Hello World");      /*Prints the null terminated
                                                string Hello World at (0,0)*/
    for(;;)
    {
            CyDelay(1000);                          //1 sec Delay
            LCD_Char_1_Position(1, 0);
            LCD_Char_1_PrintNumber(i);              //Prints the value of i
            i++;
    }
}
```

PROGRAMMING THE PSoC

- Press Shift+F6 or the Build Button to Build the hex file for the project. 
- Connect the PSoC3-003 kit to the computer using the USB cable. Burn the code by pressing Ctrl+F5 or the Program button. 

TESTING YOUR DESIGN

- Reconnect the PSoC3-003 kit to the computer or connect to battery or power supply separately and observe the LCD.
- If you are using the kit at 3.3V then short the resistor R7 on the LCD.

BASIC DIGITAL

GETTING FAMILIAR WITH COUNTERS

OBJECTIVE

To familiarize with counters and use them to drive LEDs

REQUIRED MATERIALS

1. PSoC 3 (001) board.

RELATED REFERENCE MATERIAL

1. Counter Component Data Sheet
2. Clock Component Data Sheet

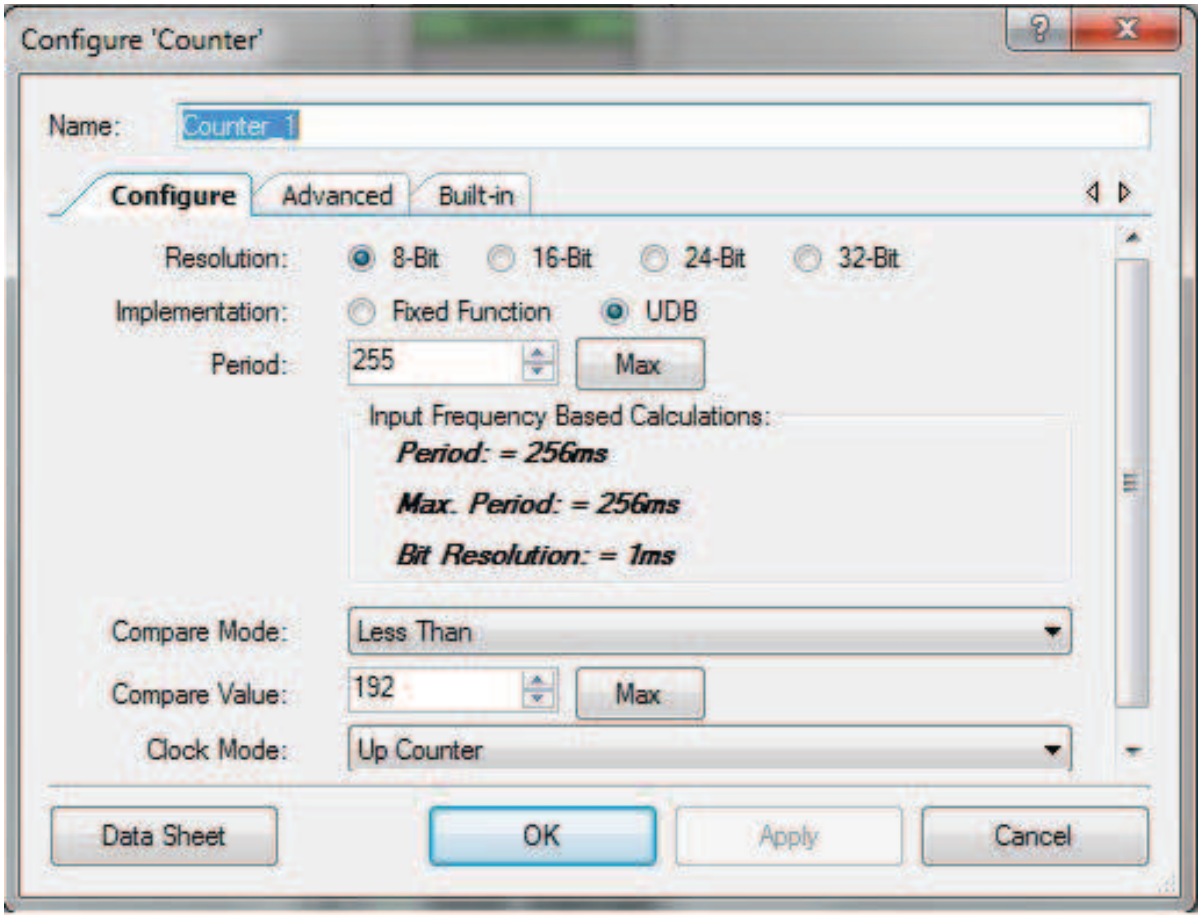
OVERVIEW

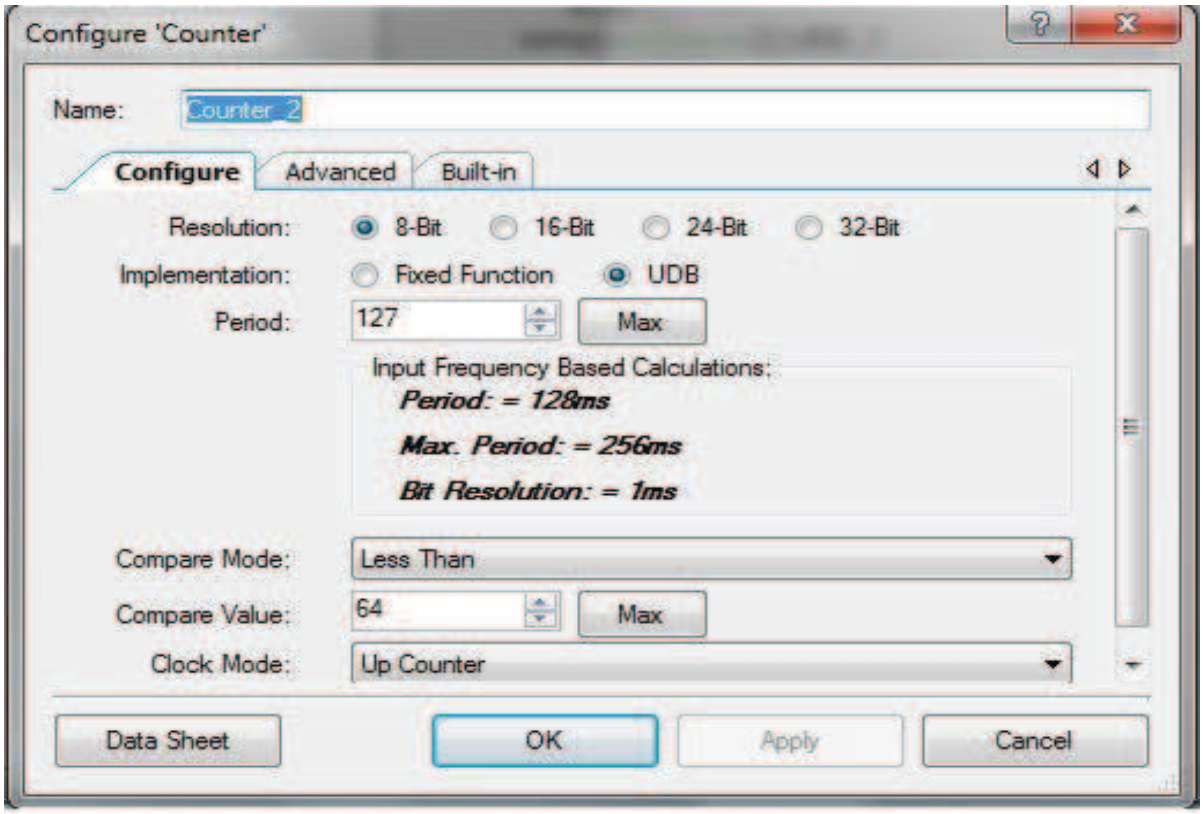
A counter is a digital device that increments/ decrements in count from an initial value specified by the user. The counter can be reset on 4 occasions as required by the user, namely capture, terminal count, on compare and on reset.

Here we use the counter to blink LEDs based on its period and its compare value.

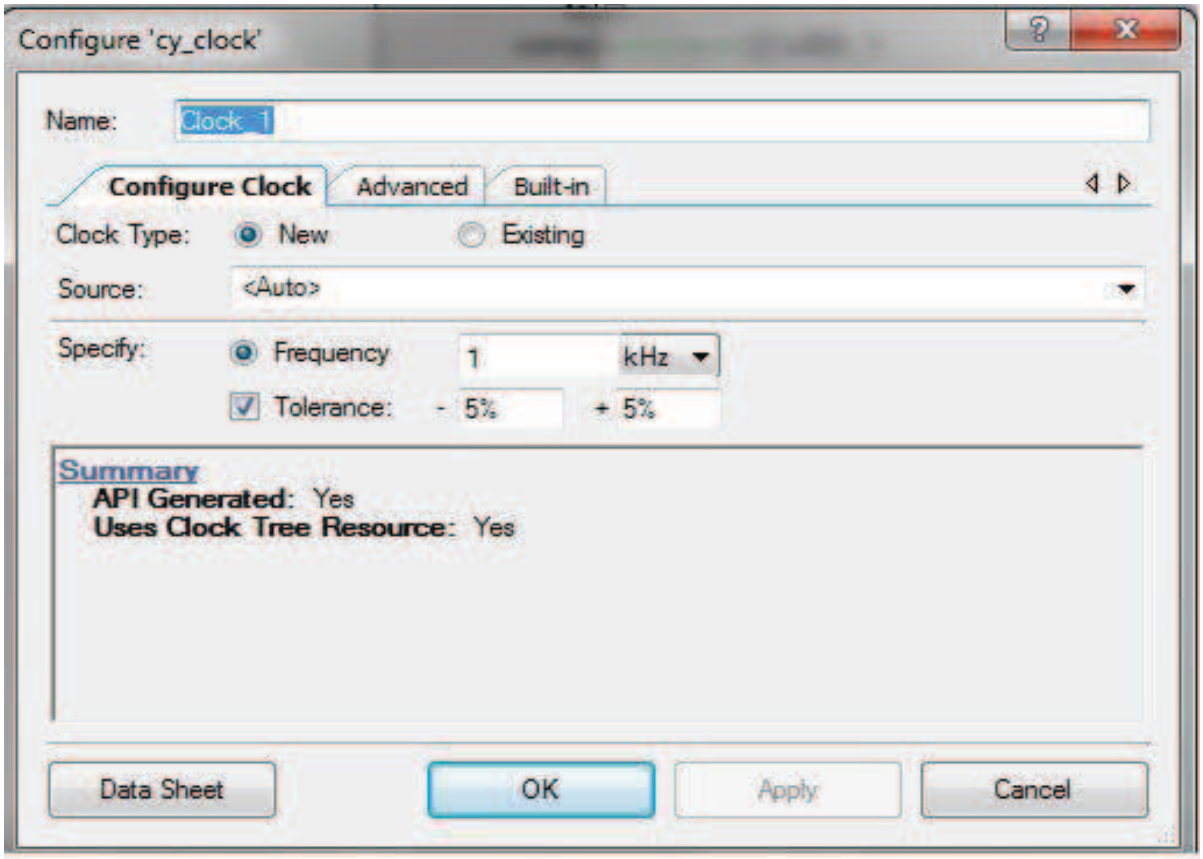
HARDWARE CONFIGURATIONS

- Open a new project is PSoC creator 3 BETA
- Select Empty PSoC 3 design and name it bdig.
- Drag and drop 2 counters. By default they are named Counter_1 and Counter_2 Respectively. Double click on them and configure them as shown in the 2 screenshots below.

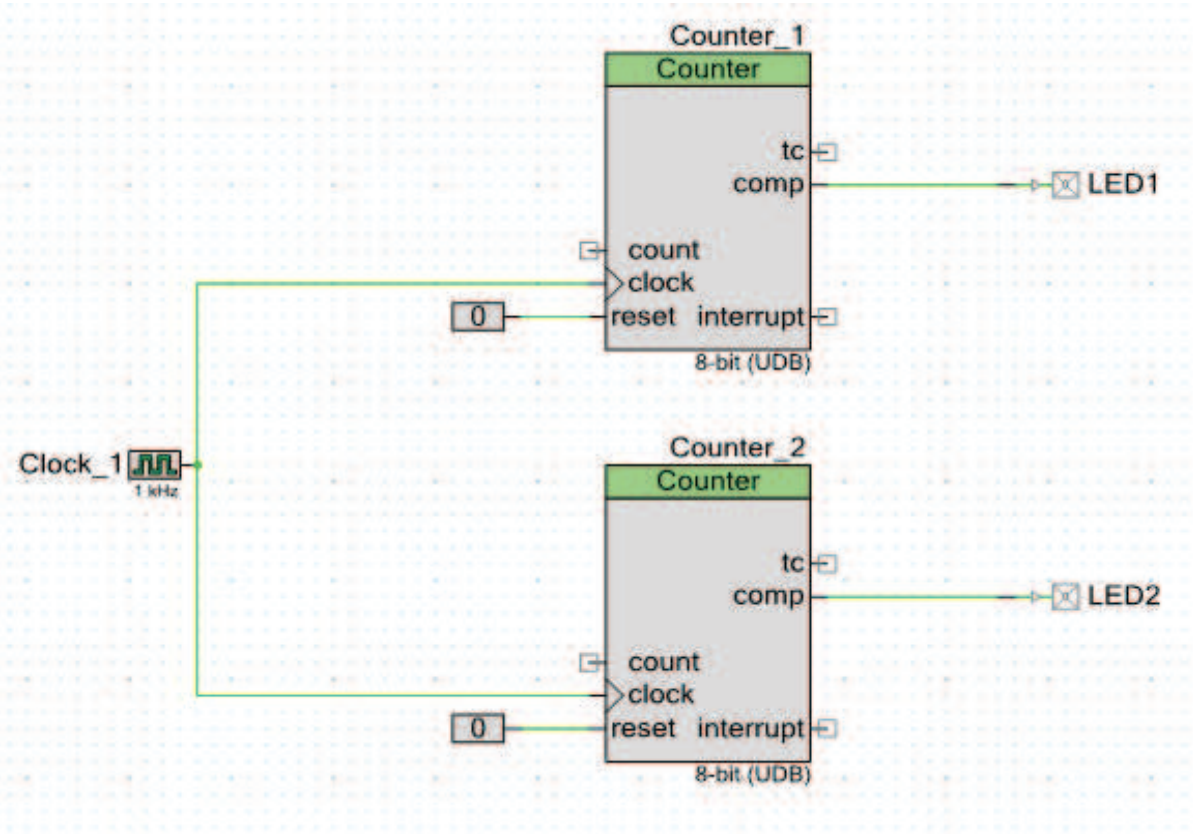




- From System>Clock, drag and drop a clock component. Double click on it and configure it to 1 kHz.



- From Ports and Pins>Digital Output Pin drag and drop 2 digital output pins and rename them as LED1 and LED2.
- Connect the Clock to the Count inputs of both the counters and the LEDs to the comp output.
- NEVER LEAVE THE RESET OF A COUNTER FLOATING. From the Digital Logic >Logic 0 drag and drop 2 logic 0 components and tie them to the reset pins.
- Your final schematic should look something like this.



ASSIGNING PINS

- Double click on basic_digital.cydwr and assign pins as shown in the screenshot below.

Alias	Name	Pin	Lock
	LED1	P2[0]	☑
	LED2	P2[2]	☑

WRITING YOUR FIRMWARE

Once this is done double click on main.c and write the following code.

```
/* ========================================
 *
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/
#include <device.h>

void main()
{
Clock_1_Start ();
Counter_1_Start ();
Counter_2_Start ();
}
```

PROGRAMMING THE PSOC

- Build your project and ensure it is error free.
- Plug in the USB Connector and click on the program option. 

TESTING YOUR DESIGN

- After programming, remove the cable and reconnect it or connect a battery SEPERATELY.
- The 2 LEDs will blink at different frequencies.
- Change the Period and Compare Value of the Counter in order to change the rate of blinking.

BASIC ANALOG

OPAMPS, ADCs and DACs

OBJECTIVE

Use Data converters to convert analog data to digital and vice versa

REQUIRED MATERIALS

1. PSoC 3 (001) board, LCD.
2. Cathode Ray Oscilloscope (CRO)

RELATED REFERENCE MATERIAL

1. VDAC8 Component Data Sheet
2. Delta Sigma ADC Component Data Sheet
3. Opamp Data Sheet

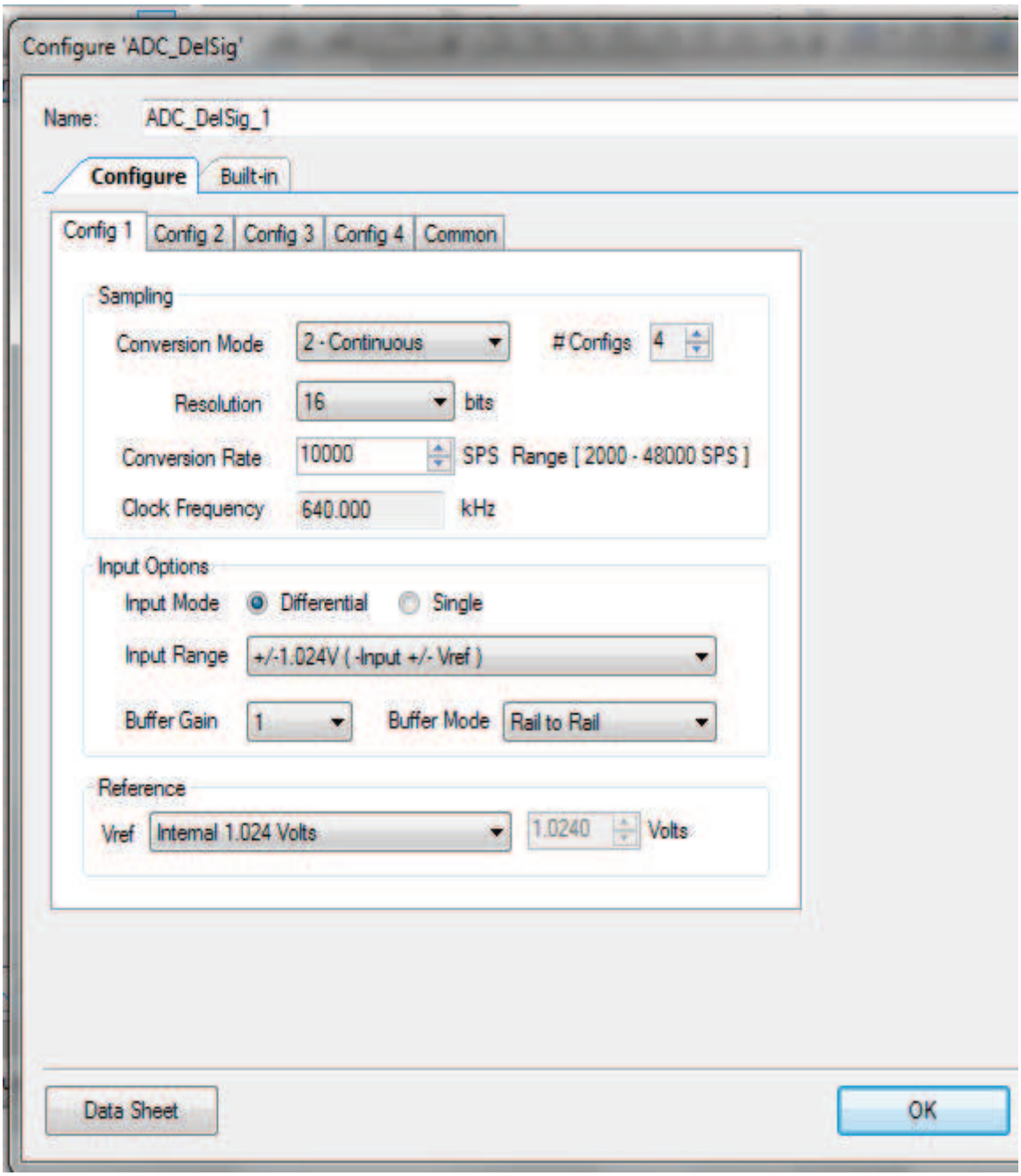
OVERVIEW

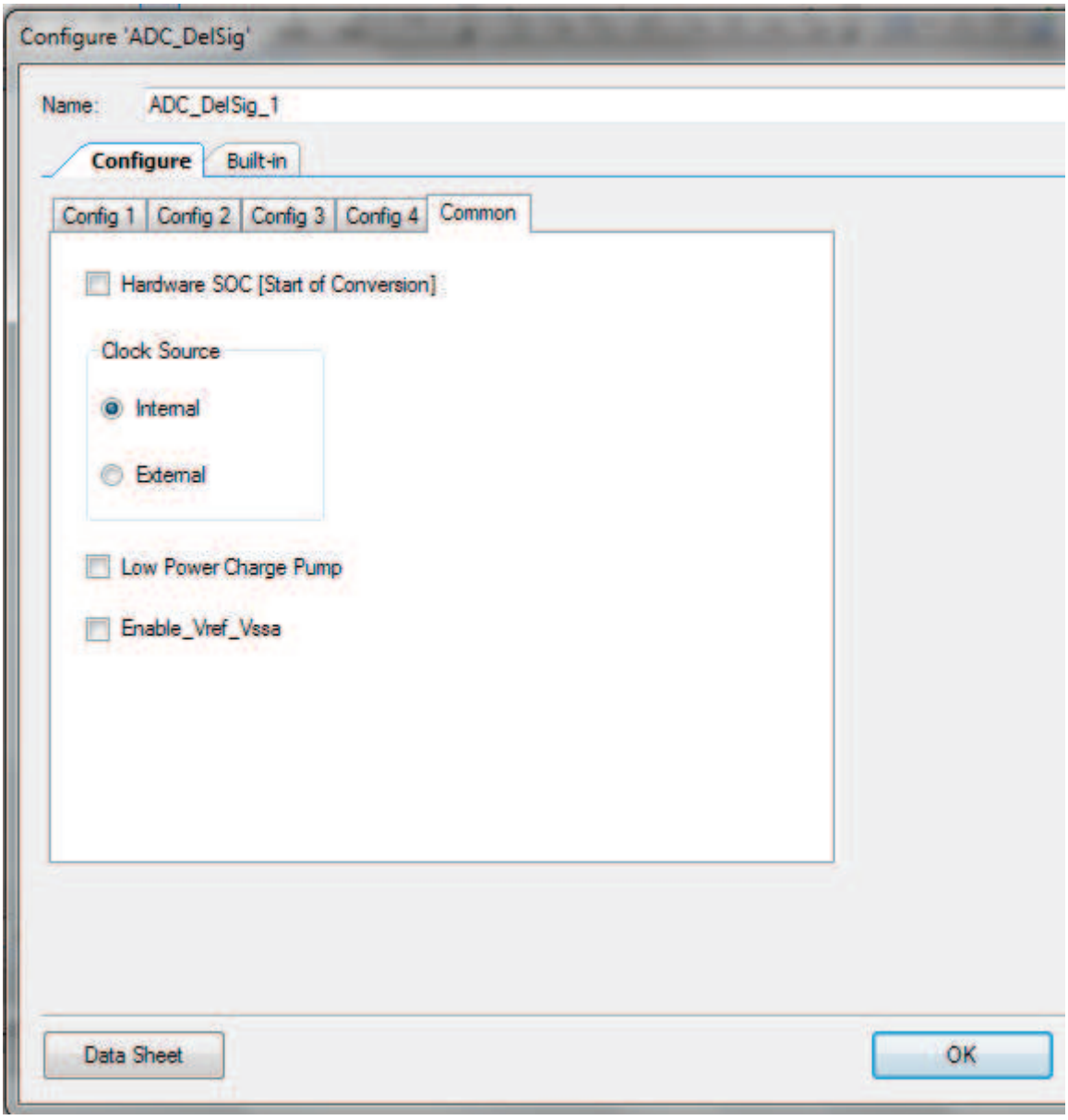
Most real life signals are analog. However, processing them in the analog domain may not be easy. So data converters are used to change data from analog form to digital and vice versa.

Operational amplifiers are fundamental building blocks of most analog devices. So familiarizing with them will also be useful. In this exercise we use opamps in a voltage buffer configuration.

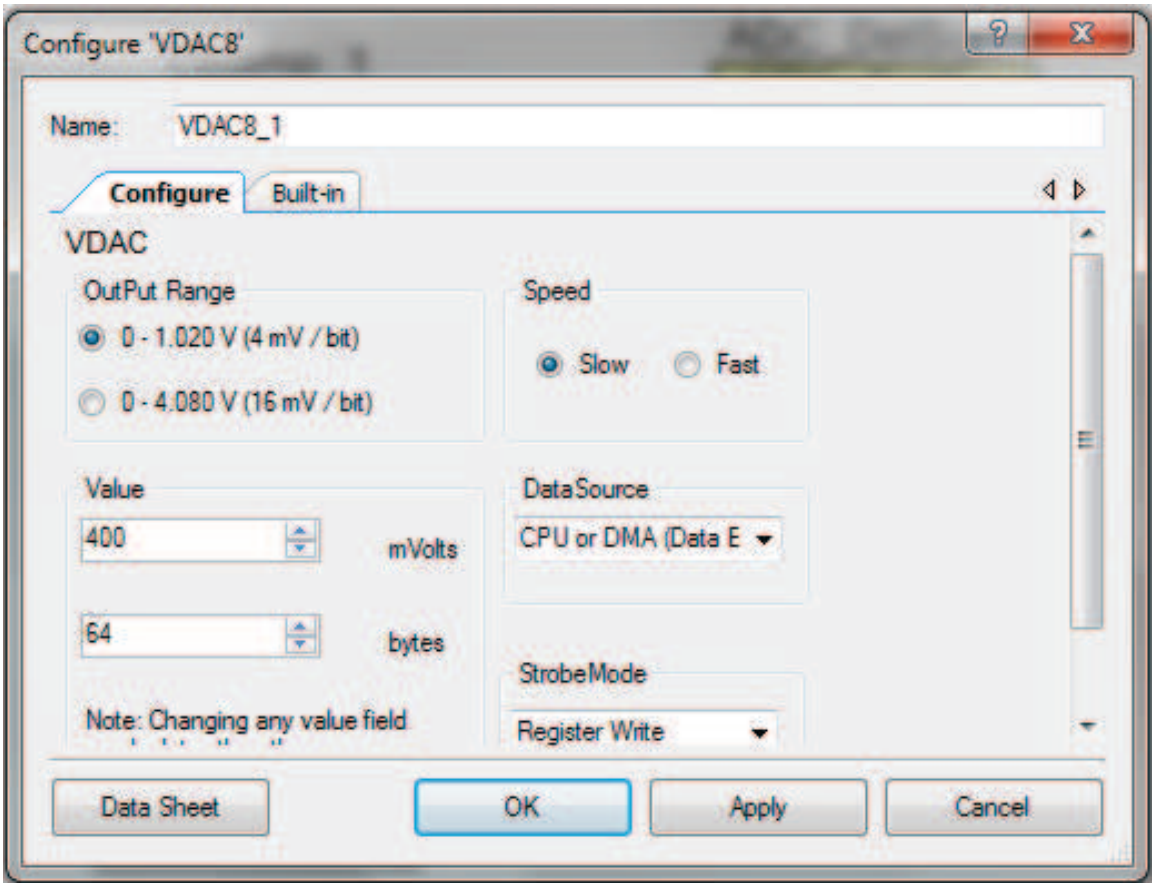
HARDWARE CONFIGURATIONS

- Open a new project is PSoC creator 3
- Name the project as basic_analog.
- From the Analog components-> ADC-> Delta Sigma ADC has to be dragged and dropped. Double click on it and configure it as shown in the following screenshots. (Continuous mode, 16 bit resolution and internal clock)



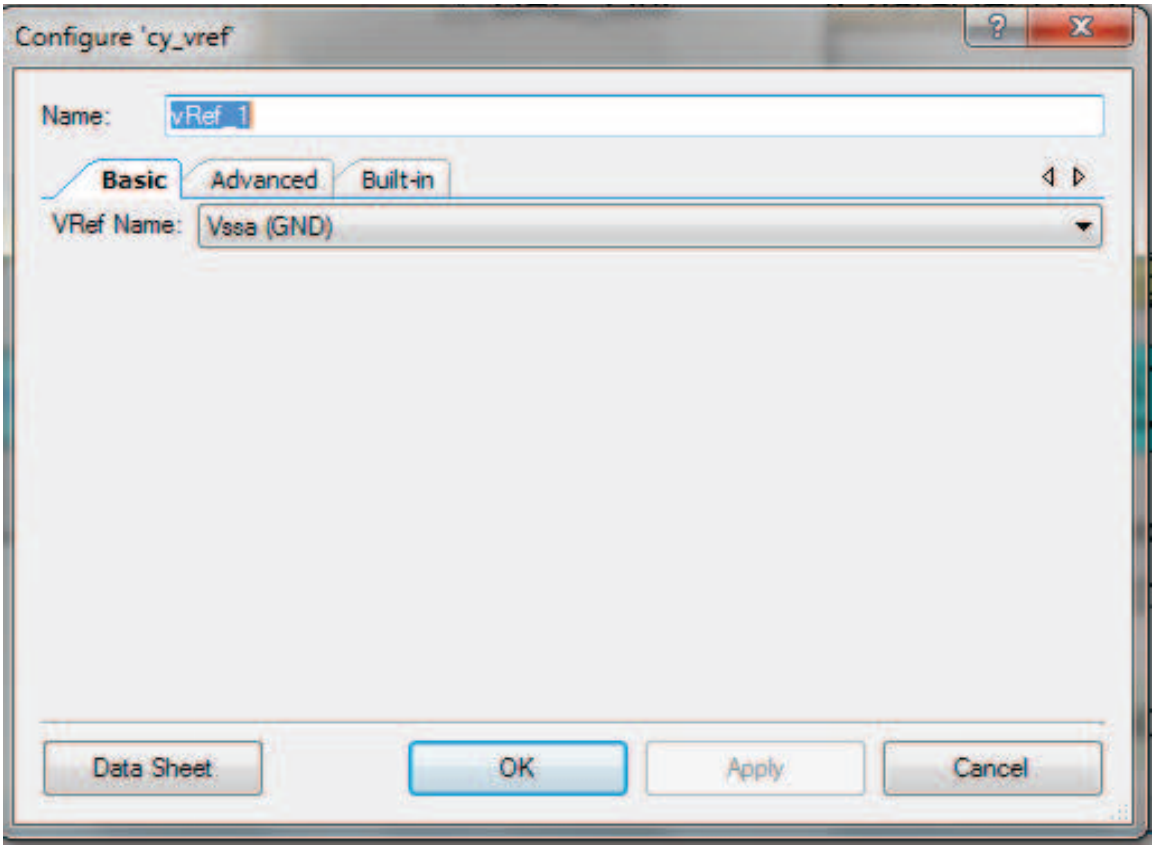

Configure 'ADC_DelSig'
Name:
ADC_DelSig_1
Configure
Built-in
Config 1
Config 2
Config 3
Config 4
Common
Hardware SOC [Start of Conversion]
Clock Source
Internal
External
Low Power Charge Pump
Enable_Vref_Vssa
Data Sheet
OK

- From the Analog components-> DAC-> VDAC8 has to be dragged and dropped. Double click on it and configure it as follows.

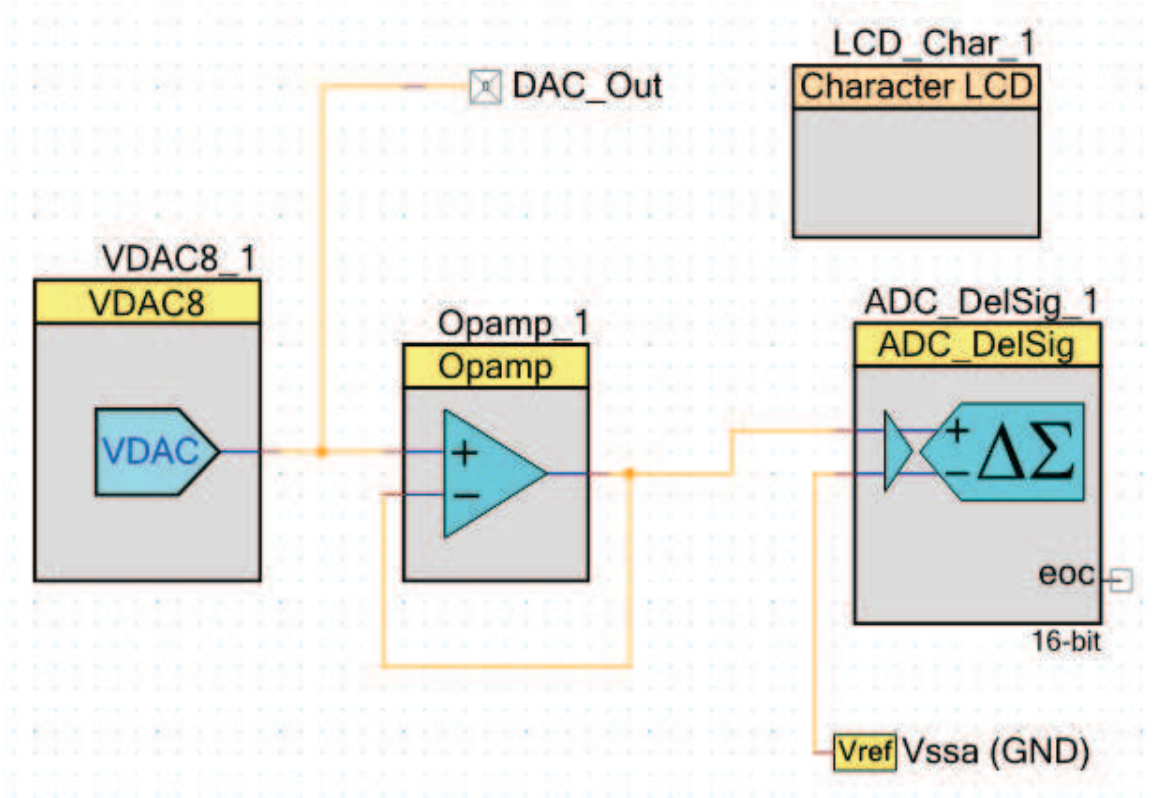


- From Ports and Pins>Analog Pin and rename it DAC_Out.
- Add a character LCD from Display>Character LCD.
- From Analog> Opamp add an opamp and connect it in the non inverting voltage buffer configuration.

- From Analog>VRef add a VRef and configure it as follows.

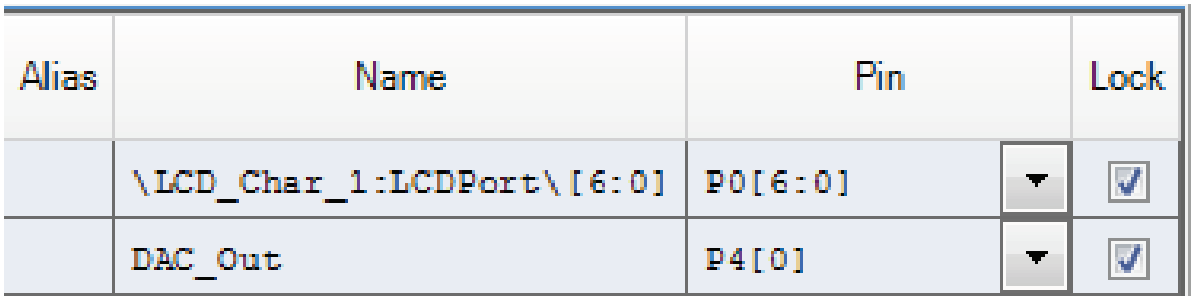


- Make the necessary connections. Your final Schematic should look like this.



ASSIGNING PINS

- Double click on basic_analog.cydwr and assign pins as shown in the screenshot below.



Alias	Name	Pin	Lock
	\LCD_Char_1:LCDPort\[6:0]	P0[6:0]	☑
	DAC_Out	P4[0]	☑

WRITING YOUR FIRMWARE

- Once this is done double click on main.c and write the following code.

```
/* ========================================
 *
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/

#include <device.h>
byte ADC_Result=0
void main()
{
LCD_Char_1_Start();
VDAC8_1_Start();
Opamp_1_Start();
ADC_Delsig_1_Start();
VDAC8_SetValue(100);// This is the value you are feeding into the DAC.
ADC_Delsig_StartConvert();
while(ADC_Delsig_IsEnfConversion(ADC_WAIT_FOR_RESULT)!=0)
      {
      ADC_Result=ADC_Delsig_GetResult();
      LCD_Char_1_Position(0, 0);
      LCD_Char_1_PrintNumber(ADC_Result);
      }
      // ADC_Result contains the value after analog to digital conversion.
}
```

PROGRAMMING THE PSOC

- Build your project and ensure that it is error free.
- Plug in the USB Connector and click on the program option. 

TESTING YOUR DESIGN

- Monitor the value from the ADC using an LCD and compare it with the value fed into the DAC. Ideally, they should be the same.
- The output of a DAC can also be monitored on the LED at Pin 4[0] or probed onto a CRO.
- Use a "for" loop to continuously increment the value fed into the DAC. Does the ADC return the correct value?
- If there is a difference in the value fed into the DAC and the value returned by the ADC, what might the possible reason be?
- Change the resolution of the ADC and retest your design.

ANALOG AND DIGITAL INTERFACING

THE ADC AND PWM INTERFACE

OBJECTIVE

To develop a servo motor controller driven by a potentiometer by interfacing the ADC component with a PWM.

REQUIRED MATERIALS

1. PSoC 3 (001) board along with a character LCD
2. A servo motor
3. A potentiometer

REQUIRED EQUIPMENT

1. Cathode Ray Oscilloscope(CRO)

RELATED REFERENCE MATERIAL

1. Delta Sigma ADC Component Data Sheet
2. PWM Component Data Sheet
3. Clock Component Data Sheet

OVERVIEW

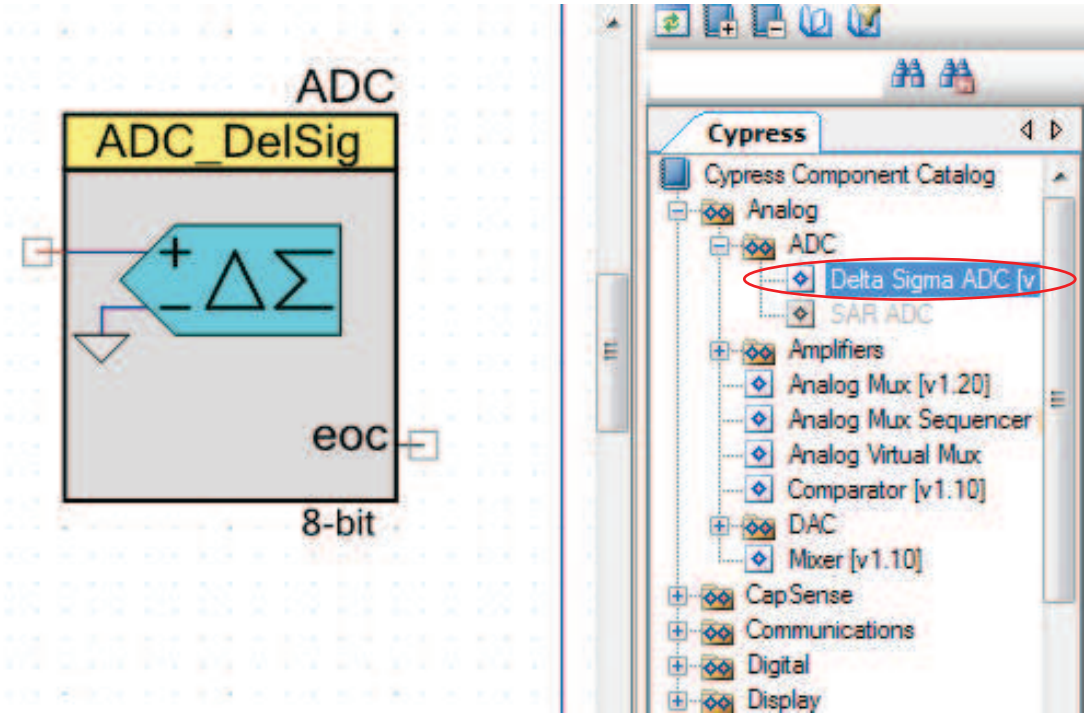
Most real world signals are analog. However, since they are continuously changing it is difficult to perform complex operations with them. The digital domain is easier to operate on and storing signals is easier.

This experiment elaborates on taking a real world analog signal (from the potentiometer), transforming it into digital and driving a servo motor using the converted signal (using a PWM).

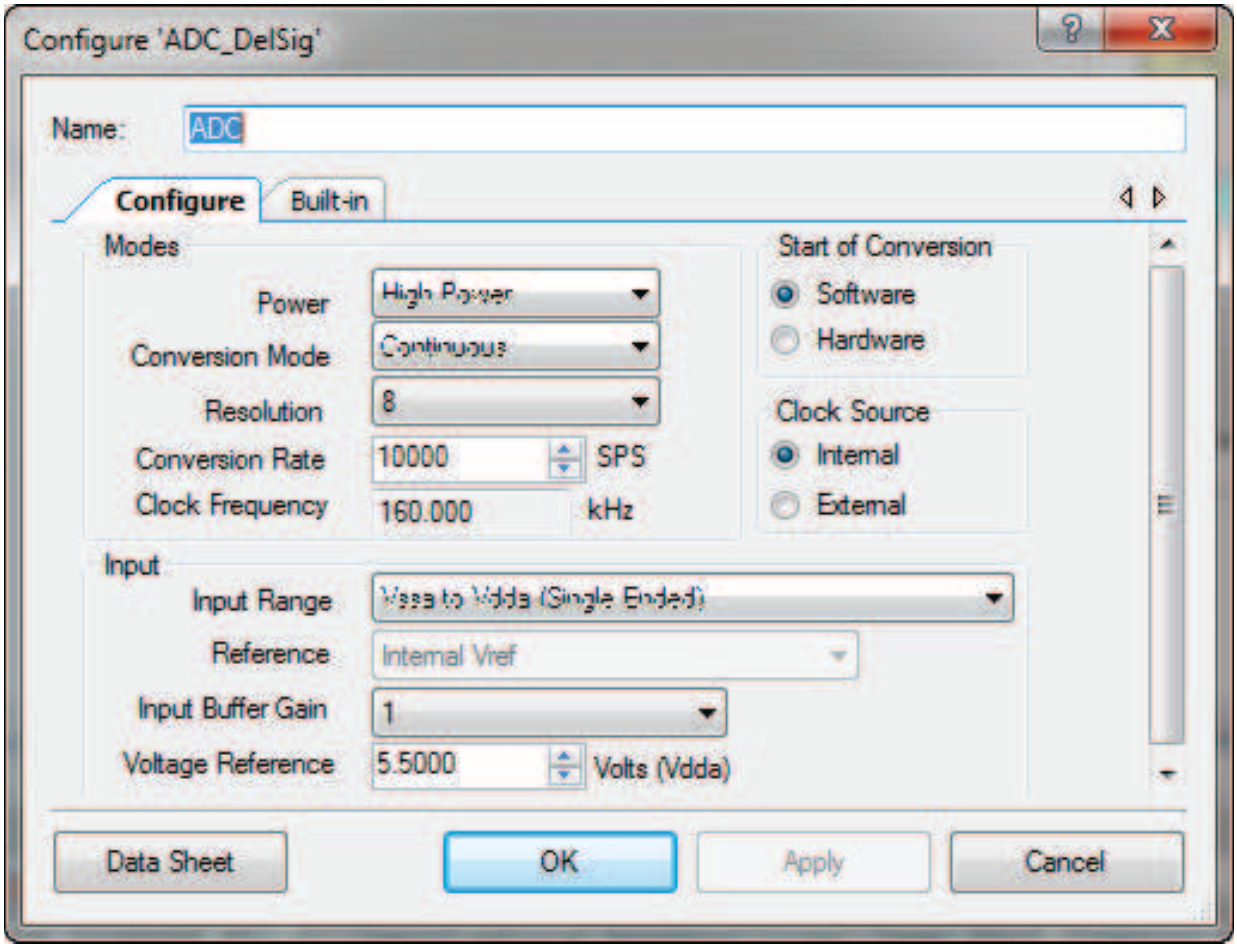
It also illustrates the ease of interfacing the analog and digital Components on a PSoC 3.

HARDWARE CONFIGURATIONS

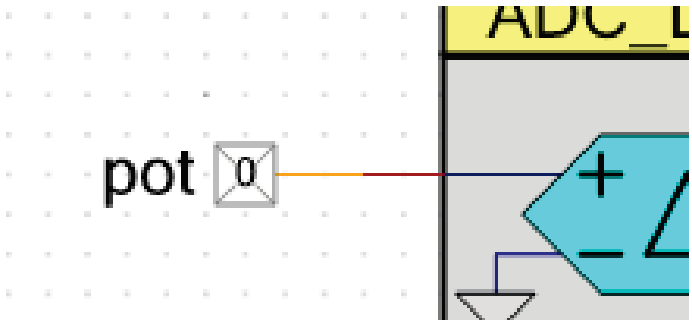
- Start a New Project. Open PSoC Creator and start "Create new project".
- Name the Project "ADC_PWM".
- Add a ADC Delta Sigma Component



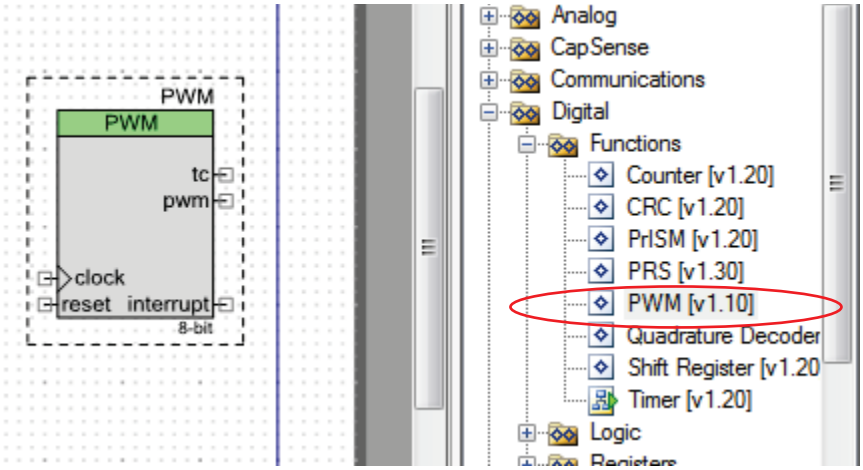
- Configure it as follows



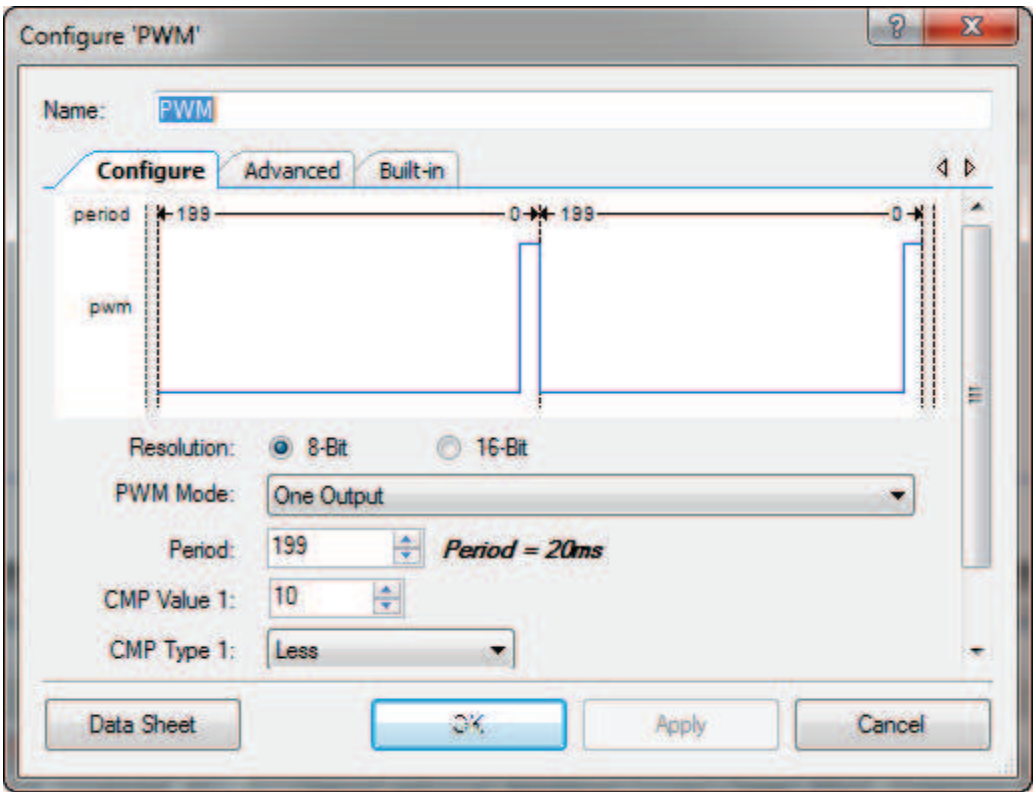
- Connect an Analog Output Pin to the ADC Input. Name it pot.



- Add a PWM Component to the project. Name it PWM

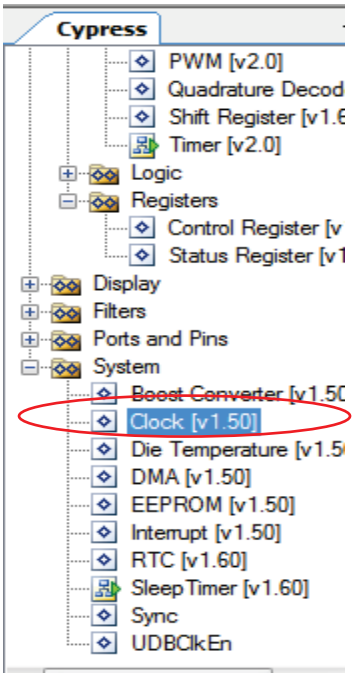


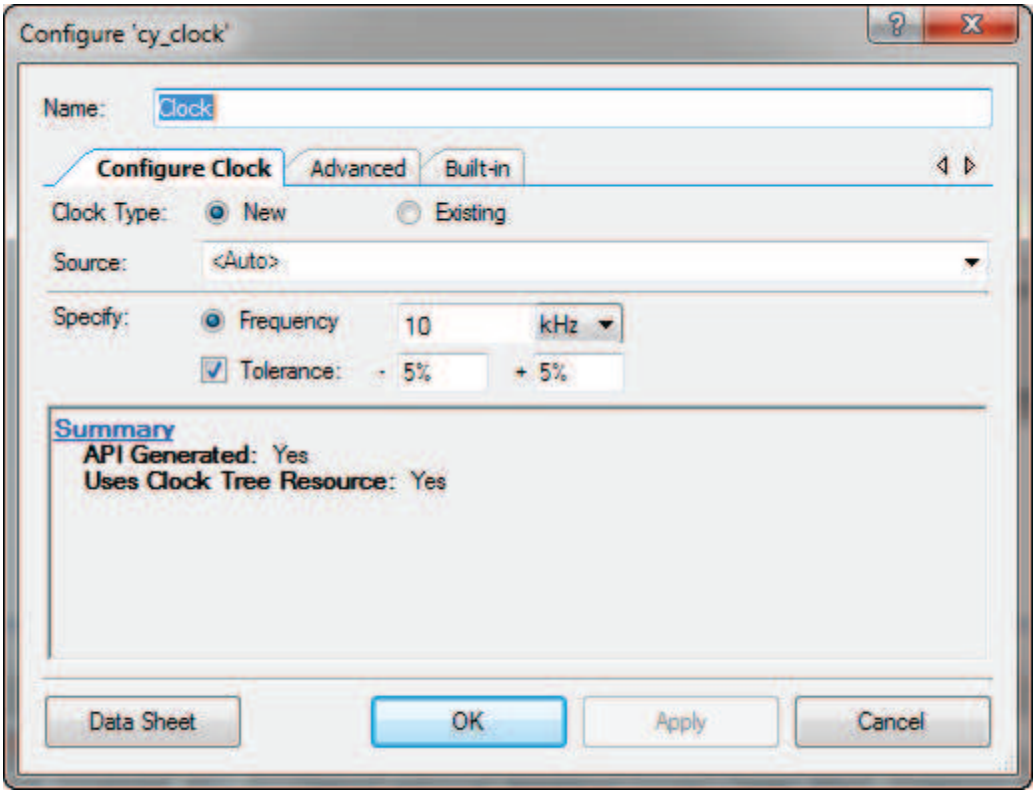
- Configure it as shown in the screenshot below.



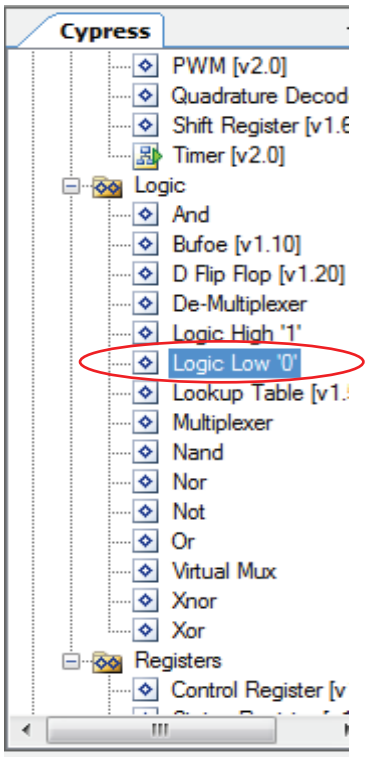
- The Clock and the Period are set such that the Period of the PWM O/P should be 20ms.
- Connect the following peripherals to the PWM block Configure each as shown

- Clock - from System> Clock to be connected to PWM clock
- Configure it as shown in the screenshot below



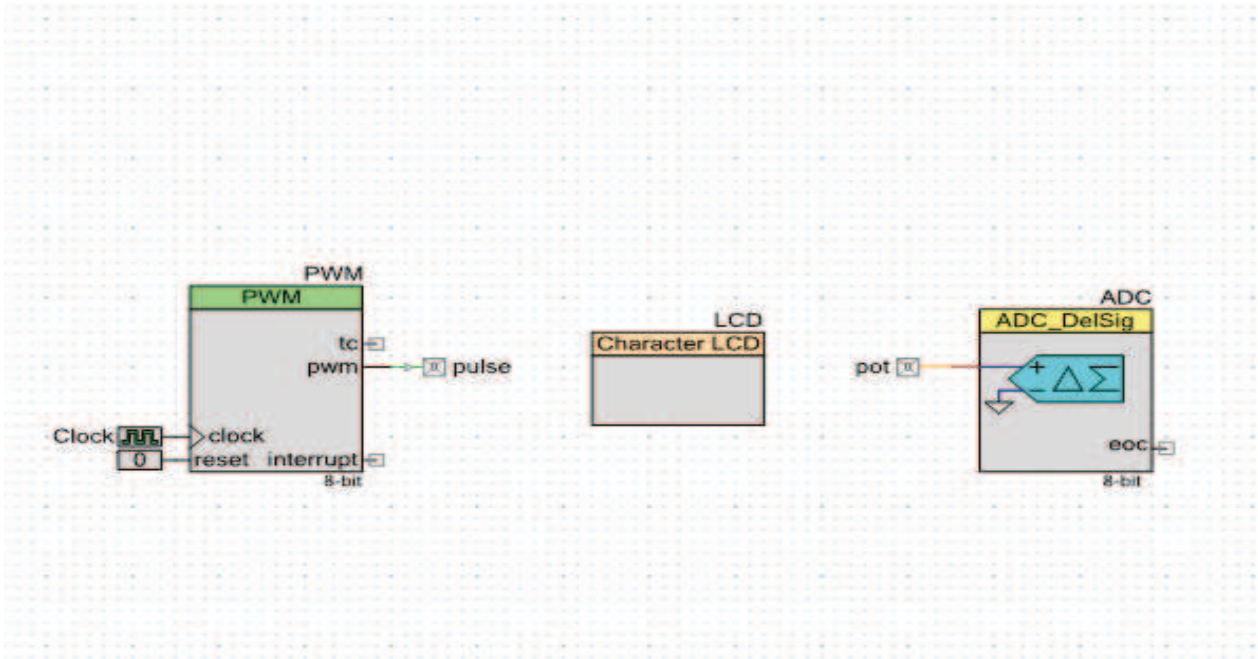


- Logic Zero from Digital>Logic>Logic Low '0' which is to be connected to PWM reset



- Digital Output pin to PWM output which is renamed as pulse.

- Add an LCD_Char Component and name it as LCD.
- The final schematic should look as shown in the screenshot below.



WRITING YOUR FIRMWARE

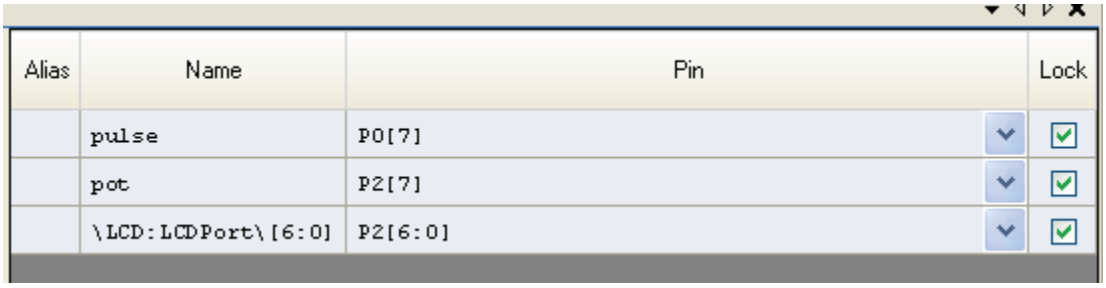
- Double click on main.c and write the following code.

```
/* ========================================
 *
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/
#include <device.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

uint8 get_width(uint8 volt);

void main()
{
      uint8 Potential,Width,LCD_OutputString;

      Clock_Enable();
    PWM_Start();
      ADC_Start();
      LCD_Start();
      LCD_Position(0,0);
      LCD_PrintString("Servo Pos:");
      for(;;)
    {
            ADC_StartConvert();
      ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
      Potential=ADC_GetResult8();
            Width=get_width(Potential);

            if((Width >9)&( Width <20)) LCD_OutputString= Width +6;
            else if (Width >19) LCD_OutputString= Width +12;
            else LCD_OutputString= Width;

            LCD_Position(0,10);
            LCD_PrintInt8(LCD_OutputString);
            PWM_WriteCompare(Width);

    }
}

uint8 get_width(uint8 volt)
{
      if          (volt<0x0C) return 5;    //12 0x0C
      else if     (volt<0x19) return 6;    //25 0x19
      else if     (volt<0x26) return 7;    //38 0x26
```

```
        else if     (volt<0x32) return 8;    //50 0x32
        else if     (volt<0x3e) return 9;    //62 0x3e
        else if     (volt<0x4B) return 10;   //75 0x4b
        else if     (volt<0x57) return 11;   //87 0x57
        else if     (volt<0x64) return 12;   //100 0x64
        else if     (volt<0x70) return 13;   //112 0x70
        else if     (volt<0x7D) return 14;   //125 0x7d
        else if     (volt<0x89) return 15;   //137 0x89
        else if     (volt<0x96) return 16;   //150 0x96
        else if     (volt<0xA2) return 17;   //162 0xA2
        else if     (volt<0xAF) return 18;   //175 0xAF
        else if     (volt<0xBB) return 19;   //187 0xBB
        else if     (volt<0xC8) return 20;   //200 0xC8
        else if     (volt<0xD4) return 21;   //212 0xD4
        else if     (volt<0xE1) return 22;   //225 0xE1
        else if     (volt<0xED) return 23;   //237 0xED
        else if     (volt<0xFA) return 24;   //250 0xFA
        else if     (volt>0xFA) return 25;   //250 0xFA
}

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

ASSIGNING PINS

- Configure the PinOuts as shown in the screenshot below.



Alias	Name	Pin	Lock
	pulse	P0[7]	☑
	pot	P2[7]	☑
	\LCD:LCDPort\[6:0]	P2[6:0]	☑

PROGRAMMING THE PSOC

- Build the code and ensure it error free.
- Plug in the USB Connector and click on the program option. 

TESTING YOUR DESIGN

- Connect the potentiometer to Pin2_7 and the output of the PWM i.e. Pin0_7 to an LED, the control pin of the servo motor and to the input of a CRO.

- The motor should move depending on the position of the potentiometer.

- On the CRO the duty cycle of the PWM should vary as the position of the potentiometer is varied.

UART

THE UART INTERFACE

OBJECTIVE

To use the UART interface on the PSoC 3 to switch an electric lamp using a computer.

REQUIRED MATERIALS

1. PSoC 3 (001) board.
2. A 5V relay
3. An electric lamp(or any other switch controlled appliance)
4. A computer with a serial port and a terminal

RELATED REFERENCE MATERIAL

1. UART Component Data Sheet
2. Internal diagram of the 5V relay used.

OVERVIEW

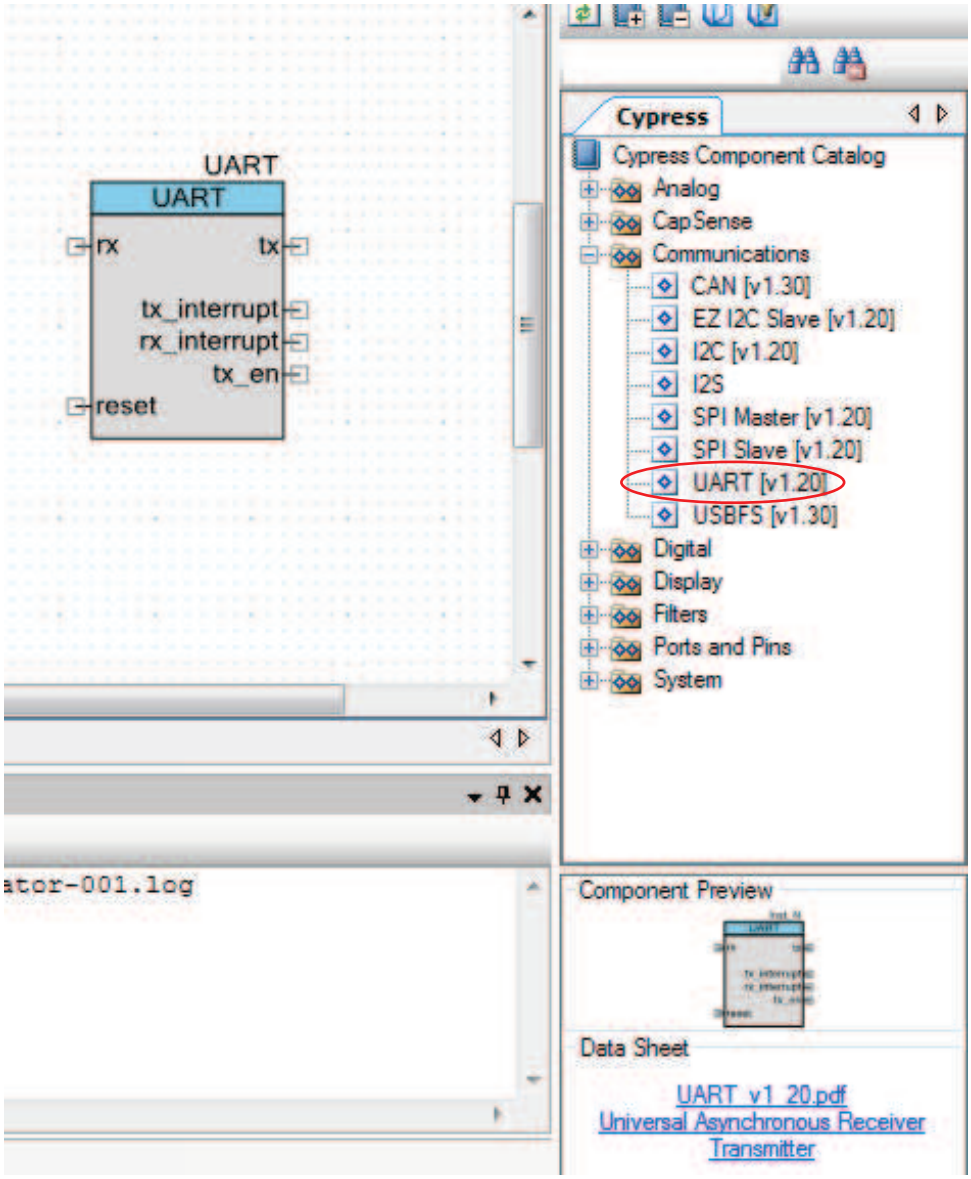
UART short for Universal Asynchronous Receiver Transmitter is a hardware which can interchange serial and parallel data. Its operation is essentially 2 step. At the transmitter end, data is broken into individual bits and transmitted. At the receiver end, the data is reassembled. Error correction is taken care of at the receiver end.

The UART can easily be interfaced with other standard communication devices. This experiment will demonstrate the above idea by using a computer command to drive a relay. The relay in turn will toggle a switch controlled appliance.

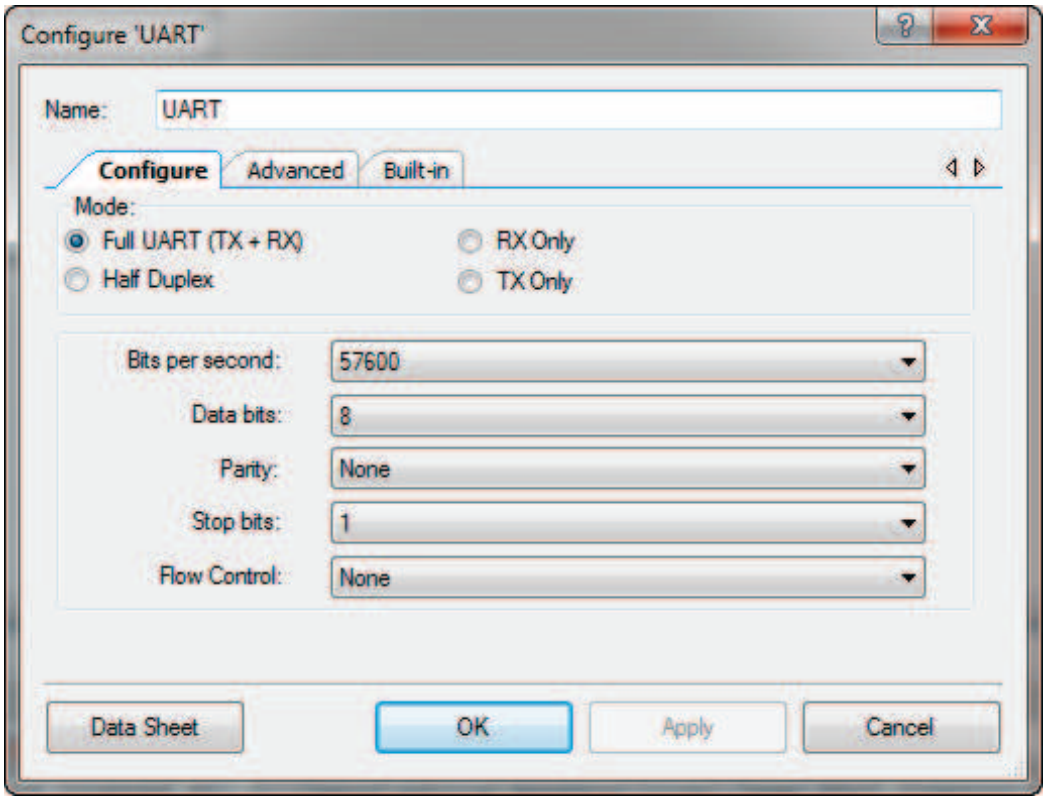
HARDWARE CONFIGURATIONS

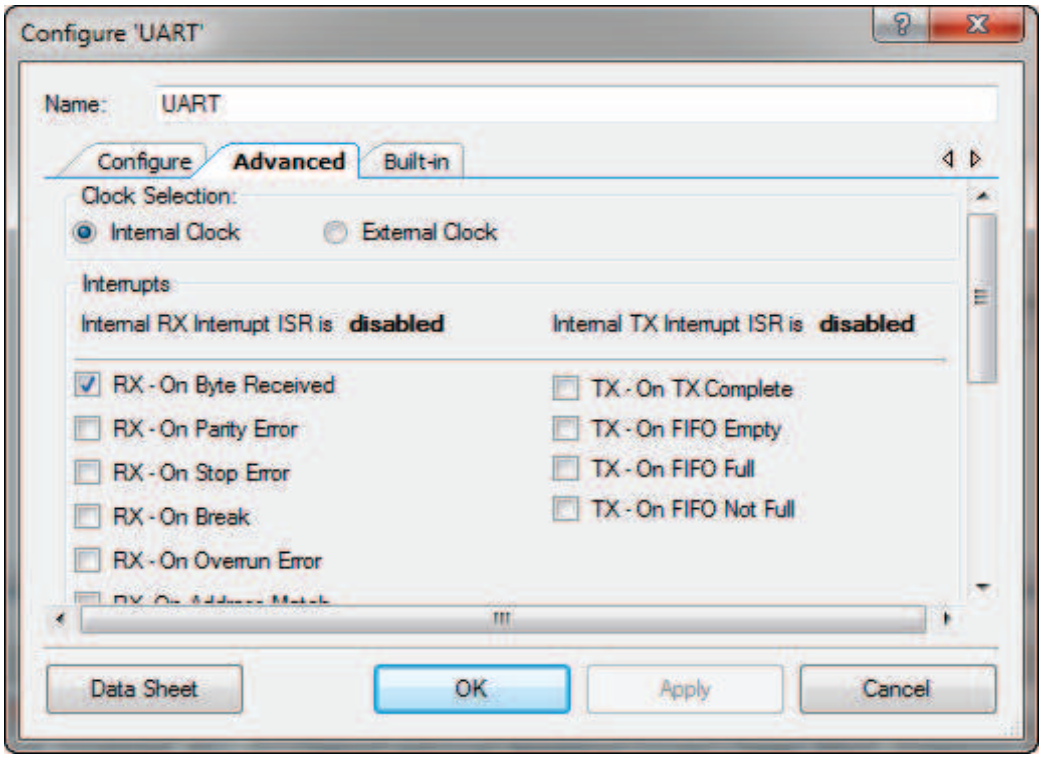
- Start a New Project. Open PSoC Creator and start "Create new project".
- Name the Project "Lab_UART".

- Make sure that the window "TopDesign.cysch" is open.
- From Component Catalog->Communications, select UART. Drag and Drop it in the grid.

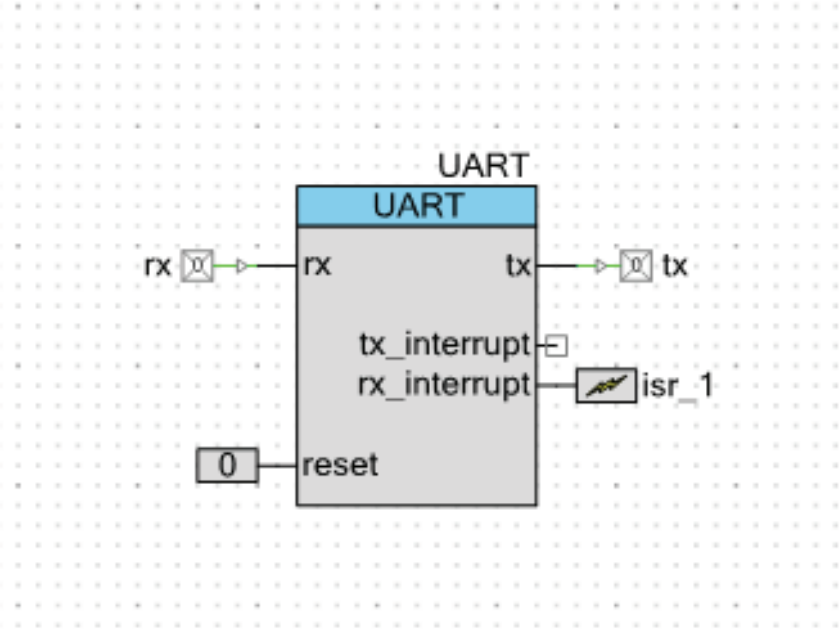


➢ Configure the UART component with the following properties.

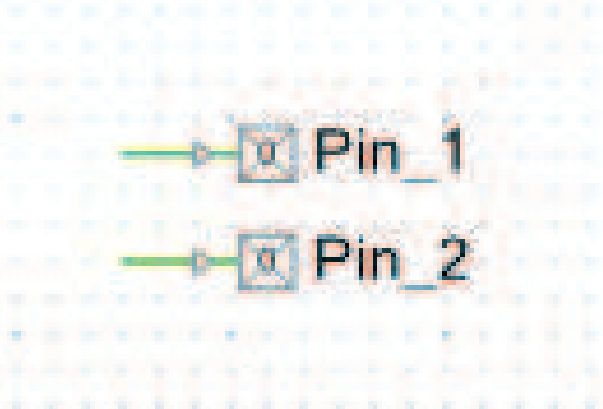




- Add the following components to the UART



  - A Digital I/P pin to the tx Input
  - A Digital O/P pin to the tx output
  - The interrupt generated by the UART should be received by "isr_1".
- Add two Digital O/P pins to the TopDesign. Name them Pin_1 and Pin_2



  - Pin_1 is the Output pin to the Electric Lamp which is connected via a 5V relay
  - Pin_2 is connected to LED.

- Add a character LCD component from Display> Character LCD and rename it as LCD_Char_1.

WRITING YOUR FIRMWARE

- Open main.c and insert the following code.

```
/* ========================================
 *
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/

#include <device.h>
#include "LCD_Char_1.h"


uint8 i,confirmed;
CY_ISR(routine1)

{
      uint8 status;
      uint8 buffer;
      ////////////////////////
      Pin_1_Write(!((Pin_1_Read()&0x01)==0x01));
      Pin_2_Write(!((Pin_2_Read()&0x01)==0x01));
      ////////////////////////
      UART_GetChar();
      buffer=UART_GetChar();
      UART_ClearRxbuffer();
      if(!((Pin_1_Read()&0x01)==0x01))
       UART_PutString("Press any key to turn off the Lamp");
      else
      UART_PutString("Press any key to to turn on the Lamp");
      UART_PutCRLF();
      UART_PutCRLF();
      LCD_Char_1_Position(1, 0);
      LCD_Char_1_PrintString(&buffer);

      LCD_Char_1_Position(1, 10);
      LCD_Char_1_PrintInt8(buffer);

      i++;
      LCD_Char_1_Position(0, 10);
      LCD_Char_1_PrintInt8(i);

      isr_1_ClearPending();
}
```

```
void main()
{
      uint8 i = 0;
      LCD_Char_1_Start();
      LCD_Char_1_Position(0, 0);
      LCD_Char_1_PrintString("Name:");
      //UART_EnableRxInt();

      UART_Start();
      isr_1_Start();
      isr_1_SetVector(routine1);
      CYGlobalIntEnable ;
      UART_PutString("Press Enter to turn off the Lamp");
      while(1)
            {
            }
}
```

ASSIGNING PINS

- Assign the Pins as shown in the screenshot below after opening Lab_UART.cydwr.

Alias	Name	Pin	Lock
	\LCD_Char_1:LCDPort\[6:0]	P0[6:0]	☑
	tx	P12[3]	☑
	rx	P2[7]	☑
	Pin_1	P2[0]	☑
	Pin_2	P12[2]	☑

PROGRAMMING THE PSOC

- Build the design and make sure it is error free.
- Plug in the USB Connector and click on the program option. 

TESTING YOUR DESIGN

- Connect the Computer's serial port to the PSoC. Connect the Pin Pin_2_0 to the Relay which is then connected to the lamp.
- Ensure that the connections for the relay are correct. Refer the internal diagram if doubtful. Reversing the connections may blow up the lamp!!!
- Turn on the terminal interface. Press a key to toggle the electric lamp with your computer!!.

THE WIRELESS INTERFACE

THE WIRELESS INTERFACE

OBJECTIVE

To develop a wireless switch using the PSoC 3

REQUIRED MATERIALS

1. PSoC 3 (001) board.
2. Artaflex wireless modules (CYRF 7936)

RELATED REFERENCE MATERIAL

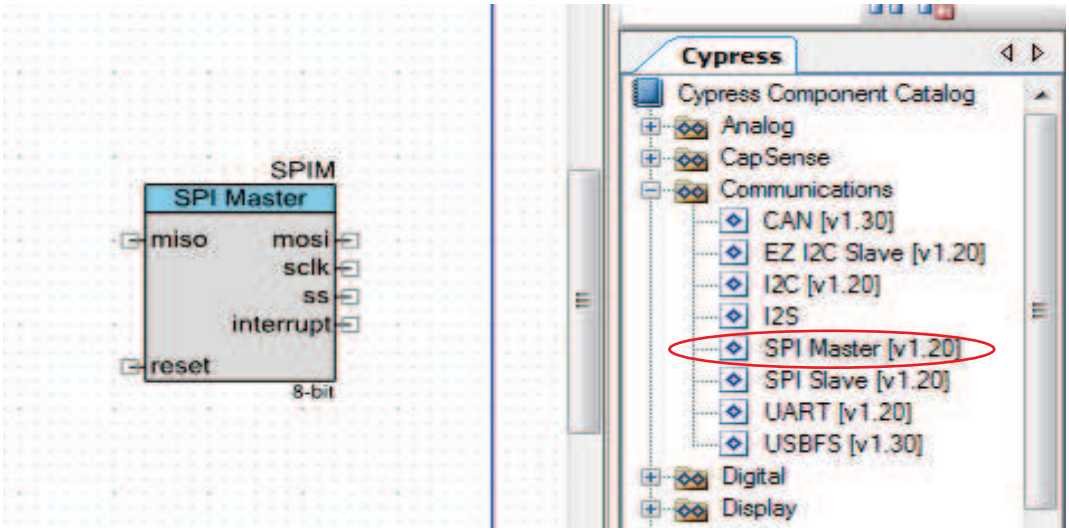
1. SPI Component Data Sheet.

OVERVIEW

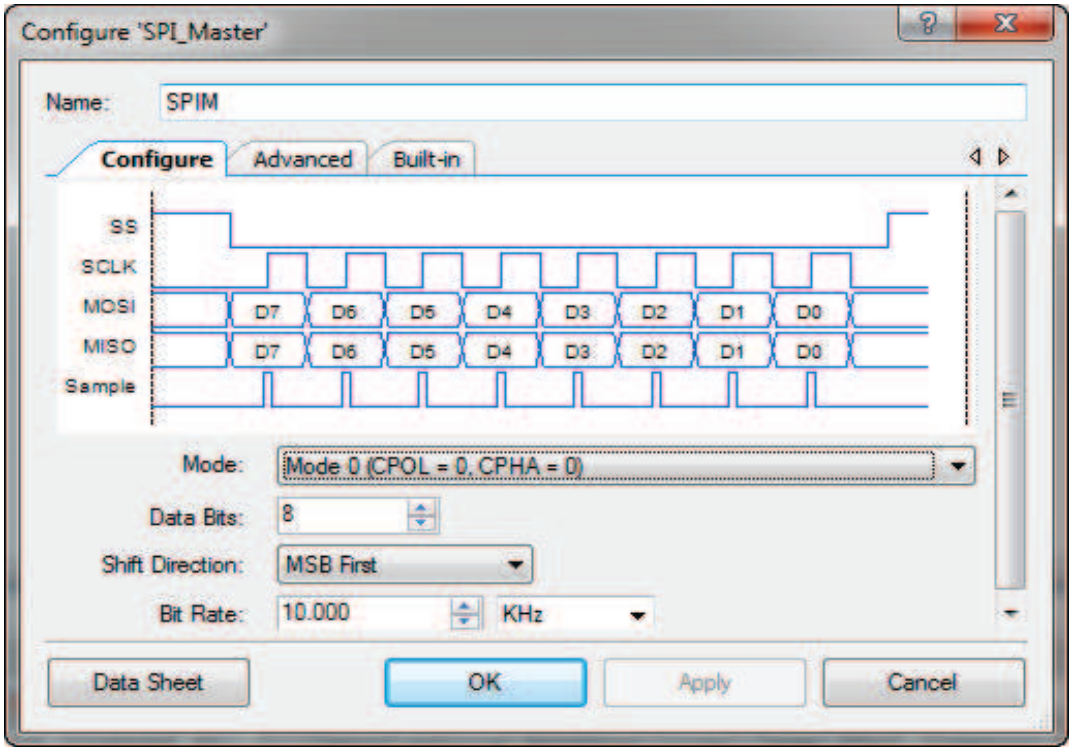
Wireless applications have become an integral part of appliances today. The need is for low power long range wireless components. This experiment focuses on developing a wireless switch with a range of about 20m.

HARDWARE CONFIGURATIONS

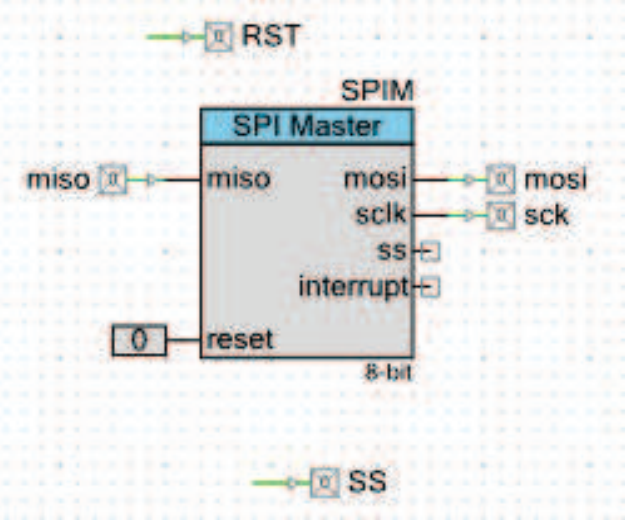
- Start a New Project. Open PSoC Creator and start "Create new project".
- Name the Project "Wireless_Transmit".
- Add an SPIM component to the Project from Communications>SPI Master.



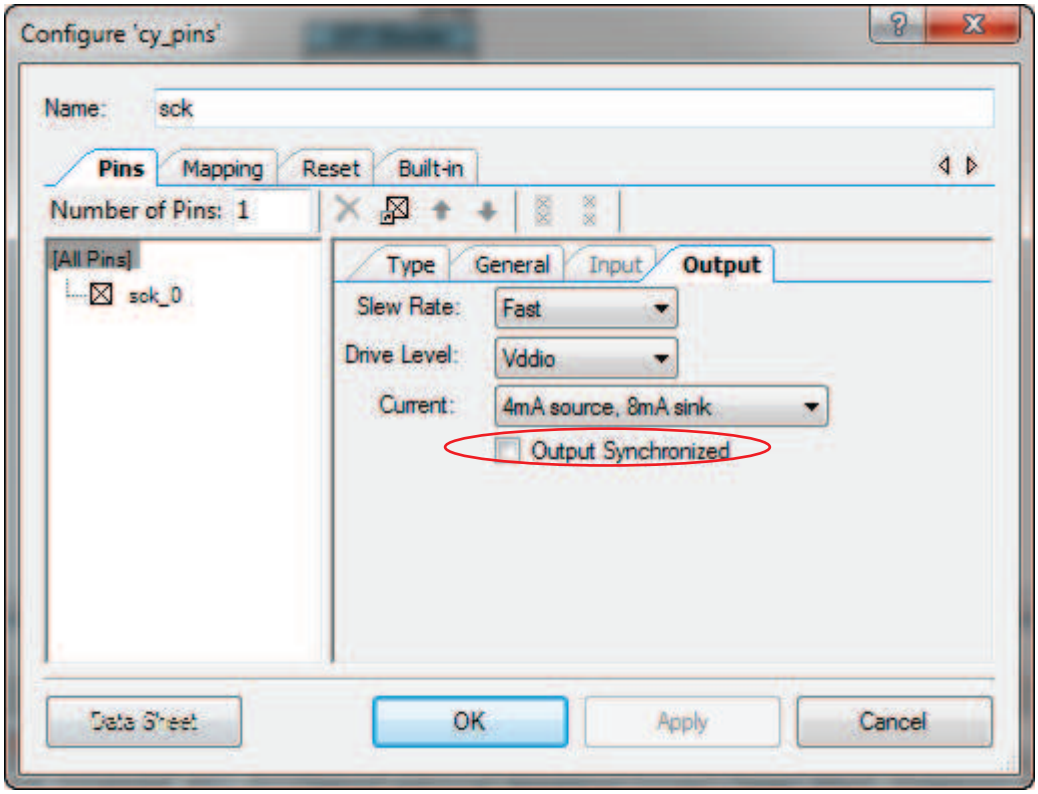
- Configure the SPIM Block as shown in the screenshot below.



- Connect the following peripherals to the SPIM Block.
  - Two Digital O/P pins (RST and SS)
  - Two Digital O/P pins (MISO and SCK) each connected to MISO and SCK of the SPIM block
  - A Digital I/P pin as MOSI to the MOSI I/P of the SPIM block
  - Connect Logic 0 to the Reset I/P.



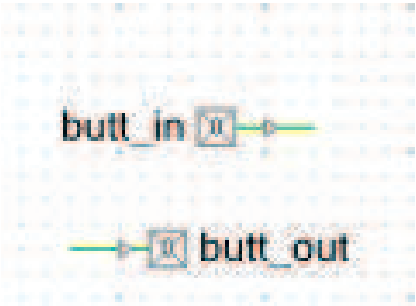
- For each Pin **Unselect** Output Sync box.



- Save this as “Transmit” and another of the same project as “Receive”. Continue with the “Transmit” project.

HARDWARE CONFIGURATIONS (TRANSMITTER)

- Add a Digital O/P and Digital I/P to the project and name them as butt_in and butt_out.



- Add a character LCD Component from Display>Character LCD.

WRITING YOUR FIRMWARE (TRANSMITTER)

- Write the following code in the main.c

```
/* ========================================
 *
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/

#include <device.h>
#include <lpregs.h>
#include <radio.h>
#include <radio_fn.h>

void main()
{
      unsigned char Status = 0;
      uint8 buff, out;
      LCD_Start();
      SPIM_Start();

      MyRadioHWReset(); //MyRadioWrite(0x1D,0x00); //Reset in SW
      CyDelay(100);
```

```
    ////////////////////////////////////////
    //Write data=0x45 to Tx buffer
    WritePayloadToRadio(0x01);
    MyRadioWrite(TX_CTRL_ADR,TX_GO);//TX_CTRL_ADR,TX_GO
    CyDelay(100);//wait
    //Setting To idle mode
    MyRadioWrite(TX_CTRL_ADR,TX_CLR);//TX_CTRL_ADR,clear TX_BUFFER
    CyDelay(100);//wait
    ////////////////////////////////////////

    MyRadioInit(DATMODE_64SDR);
    //Transmit length
    MyRadioWrite(TX_LENGTH_ADR,0x01);


    LCD_Position(0,0);LCD_PrintString("CH:");LCD_Position(0, 3);
    LCD_PrintInt8(MyRadioGetChannel());

  for(;;)
  {
          CyDelay(100);

          if ((butt_in_Read()&0x01)==0x01) buff=0x0a;
          else buff=0;

          out= butt_out_Read();
          if (buff==0x0a) butt_out_Write(!(out));

          MyRadioWrite(TX_CTRL_ADR,TX_CLR);//TX_CTRL_ADR,clear TX_BUFFER
          CyDelay(100);//wait
          LCD_Position(1,13); LCD_PrintInt8(buff);
          WritePayloadToRadio(buff);                //set to i later
          MyRadioWrite(TX_CTRL_ADR,TX_GO);//TX_CTRL_ADR,TX_GO
          CyDelay(100);//wait
          //Setting To idle mode
          while (!(( Status = MyRadioRead(TX_IRQ_STATUS_ADR)) & TXC_IRQ));

  }
}
```

- Add the following files to the project
  - lpregs.h
  - radio.h
  - radio_fn.h
  - radio.c
  - radio_fn.c

ASSIGNING PINS (TRANSMITTER)

- Assign the PinOuts as shown in the screenshot below.

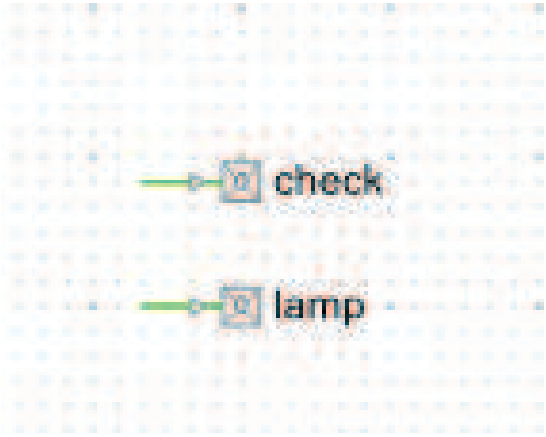
Alias	Name	Pin	Loc
	\LCD:LCDPort\[6:0]	P0[6:0]	
	SS	P5[3]	
	mosi	P12[0]	
	sck	P5[0]	
	miso	P1[4]	
	RST	P5[2]	
	butt_in	P15[3]	
	butt_out	P2[0]	

PROGRAMMING THE PSOC

- Build the design and ensure it is error free
- Plug in the USB Connector and click on the program option. 
- Now Continue with the "Receive" Project we had saved earlier.

HARDWARE CONFIGURATIONS (RECEIVER)

- Add two Digital O/P Pins to the project. Name them as Lamp and Check.



- Add a character LCD from Display>LCD and name it as LCD.

WRITING YOUR FIRMWARE (RECEIVER)

- Write the following code in the main.c

```
/* ========================================
 *
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/

#include <device.h>
#include <lpregs.h>
#include <radio.h>
#include <radio_fn.h>


void main()
{
      uint8 Payload,out;

      LCD_Start();
      SPIM_Start();

      MyRadioHWReset(); //MyRadioWrite(0x1D,0x00); //Reset in SW
      CyDelay(100);

      MyRadioInit(DATMODE_64SDR);
      //Receive length
```

```
    MyRadioWrite(RX_LENGTH_ADR,0x01);

    ////////////////////////////////////////
    MyRadioStartReceive();
    CyDelay(100);//wait
    //Setting To idle mode
    Payload = MyRadioRead(RX_BUFFER_ADR);
    //MyRadioWrite(RX_CTRL_ADR,RX_CLR);//TX_CTRL_ADR,clear TX_BUFFER
    CyDelay(100);//wait
    ////////////////////////////////////////


    LCD_Position(0,0);LCD_PrintString("CH:");LCD_Position(0, 3);
    LCD_PrintInt8(MyRadioGetChannel());

  for(;;)
  {
          CyDelay(100);
          MyRadioStartReceive();
          CyDelay(100);//wait
          //Setting To idle mode
          Payload = MyRadioRead(RX_BUFFER_ADR);
          //MyRadioWrite(RX_CTRL_ADR,RX_CLR);//TX_CTRL_ADR,clear TX_BUFFER
          CyDelay(100);//wait
          out= check_Read();
          if (Payload==0x0a)
                {
                      check_Write(!(out));
                      lamp_Write(!(out));
                }


          LCD_Position(1,0);LCD_PrintInt8(Payload);
    }
}
```

- Add the following files to the project
  - lpregs.h
  - radio.h
  - radio_fn.h
  - radio.c
  - radio_fn.c

ASSIGNING PINS (RECEIVER)

- Assign the PinOuts as follows

Alias	Name	Pin	Loc
	\LCD:LCDPort\[6:0]	P0[6:0]	
	SS	P5[3]	
	mosi	P12[0]	
	sck	P5[0]	
	miso	P1[4]	
	RST	P5[2]	
	check	P2[0]	
	lamp	P12[2]	

PROGRAMMING THE PSOC

- Build the design and ensure it is error free
- Plug in the USB Connector and click on the program option. 

TESTING YOUR DESIGN

- Connect the Artaflex Module to the 12 Pin port.
- Connect the relay which is connected to the lamp to the FT kit
- Now power up both the FT kits.
- When the button is pressed on the Tx kit should toggle the LED on the Rx kit and lamp. The range of the wireless module is approx 20m in Line of Sight.

CAPSENSE

WIRELESS TOUCH CONTROL FOR APPLIANCES

OBJECTIVE

To understand capsense and use it to develop a wireless touch control for a servo motor.

REQUIRED MATERIALS

1. PSoC 3 (001) boards
2. Servo motor
3. Artaflex wireless modules (CYRF7936).

RELATED REFERENCE MATERIAL

1. SPI Component Data Sheet
2. Capsense Component Data Sheet

OVERVIEW

Touch controlled appliances are becoming an everyday phenomenon. Touch screen phones, computers and tablets are capturing the markets. The principle behind most of them is capsense, a technology which will be the focus of this experiment.

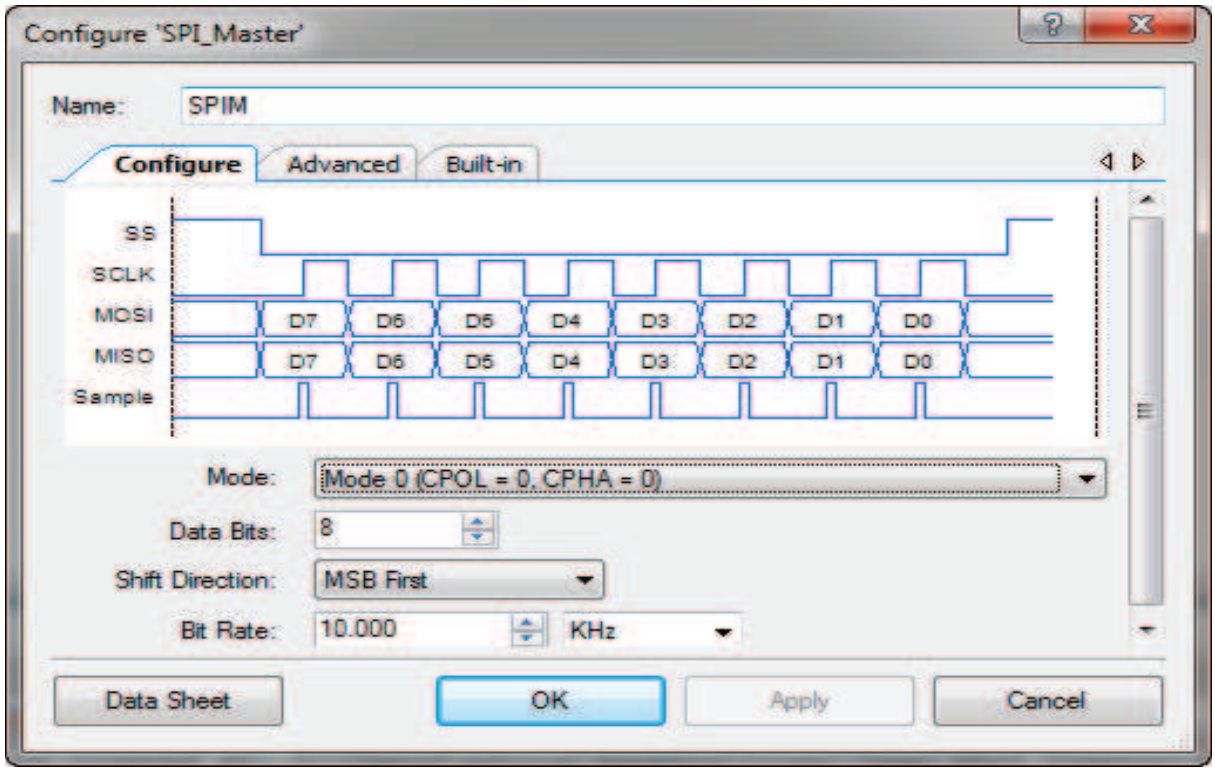
Previous experiments dealt with the wireless interface. The wireless interface will be used to convert a human touch on one PSoC 3 kit to control a servo motor interfaced with another PSoC 3 kit.

HARDWARE CONFIGURATIONS

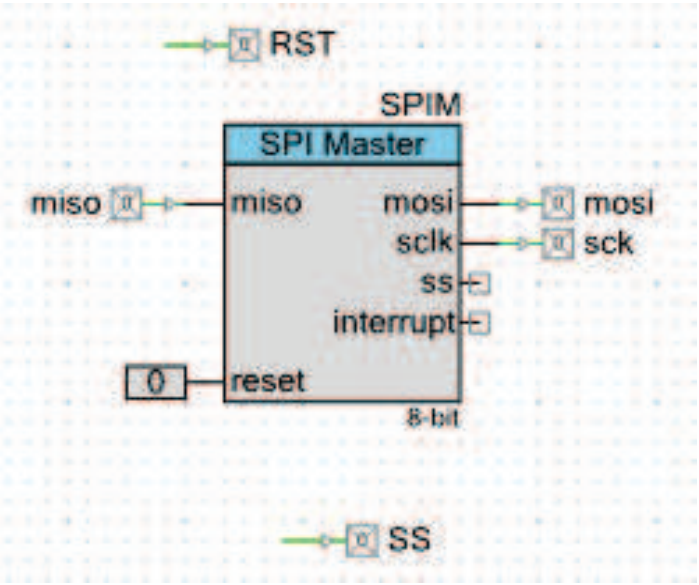
- Start a New Project. Open PSoC Creator and start "Create new project".
- Name the Project "Cap_Transmit".

HARDWARE CONFIGURATIONS (TRANSMITTER)

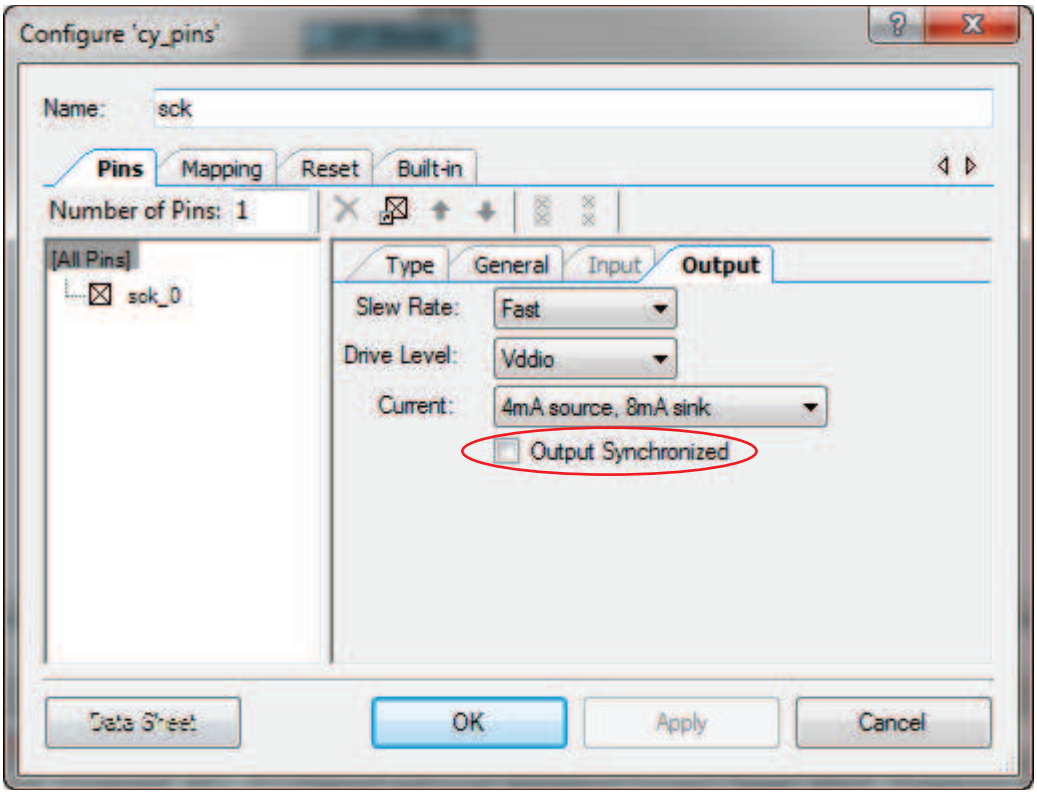
- Configure the SPIM Block as follows.



- Connect the following peripherals to the SPIM Block as shown below.
  - Two Digital O/P pins (RST and SS)
  - Two Digital O/P pins (MISO and SCK) each connected to MISO and SCK of the SPIM block
  - A Digital I/P pin as MOSI to the MOSI I/P of the SPIM block
  - Connect Logic 0 to the the Reset I/P.

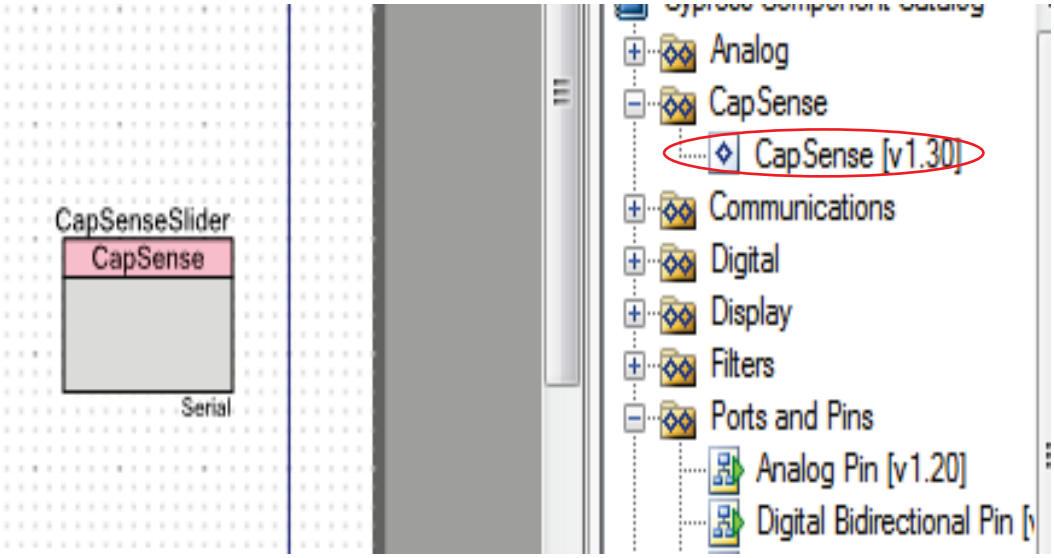


- For each Pin **Unselect** Output Sync box.

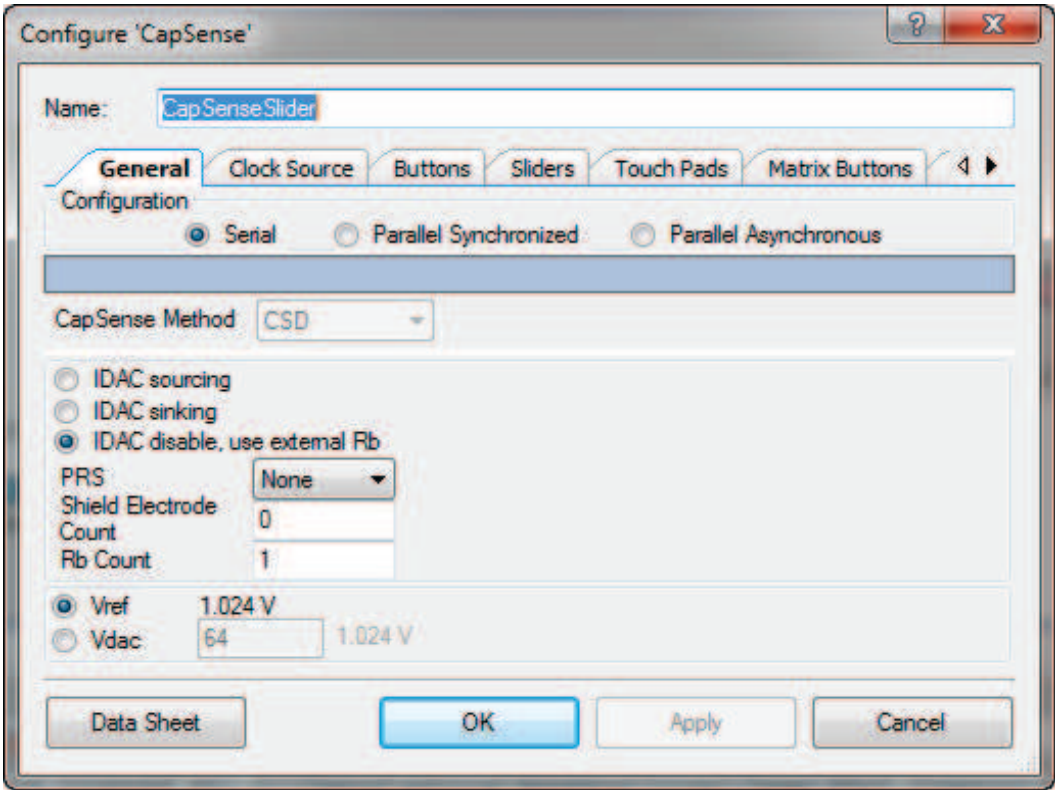


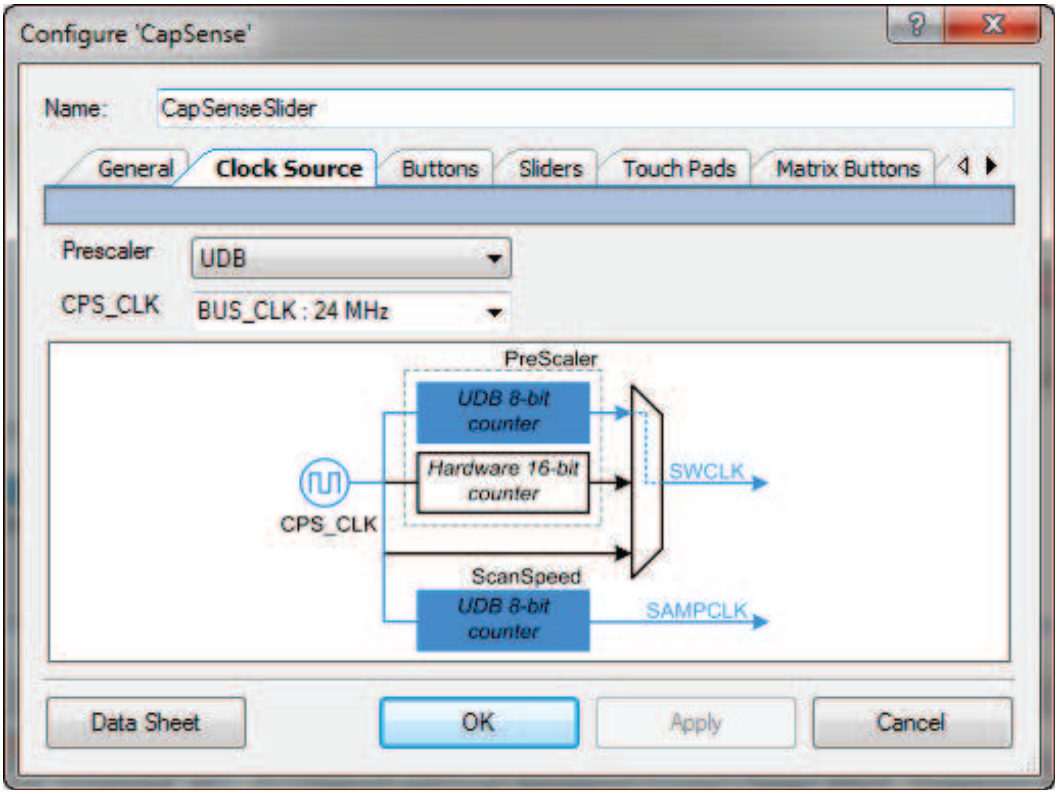
- Save this as "Transmit" and another copy also as "Capsense_Receive". Continue with the "Capsense _Transmit" project.

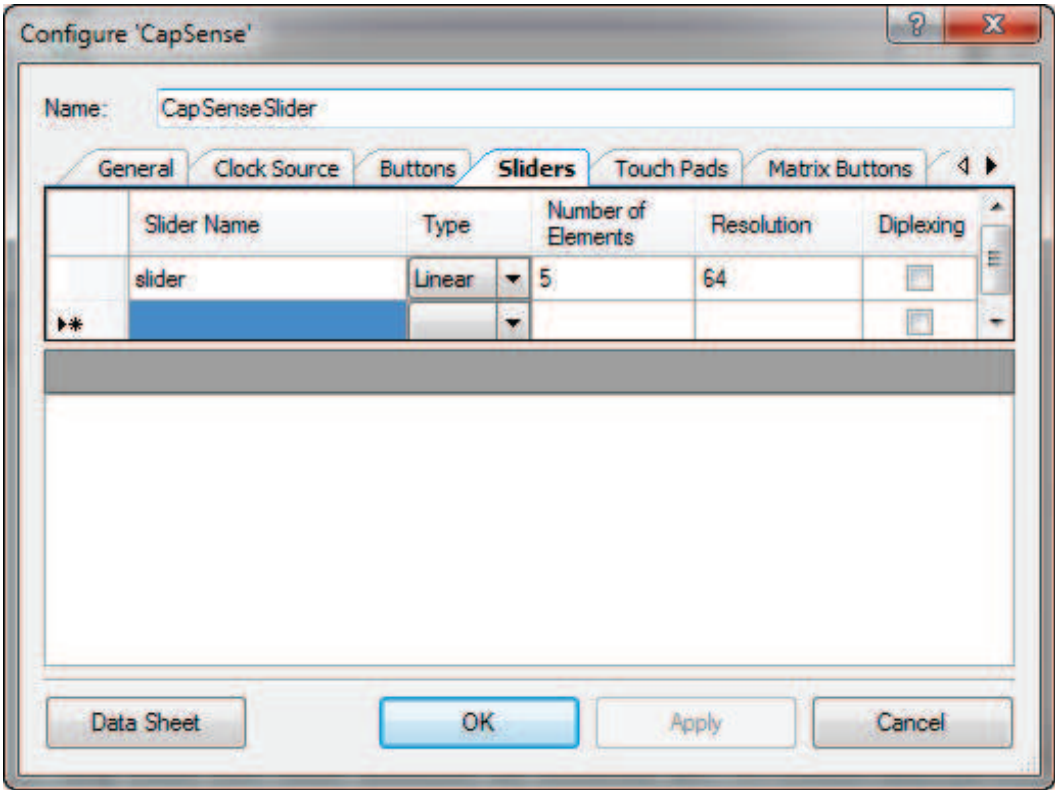
- Add a Capsense component from Capsense>Capsense

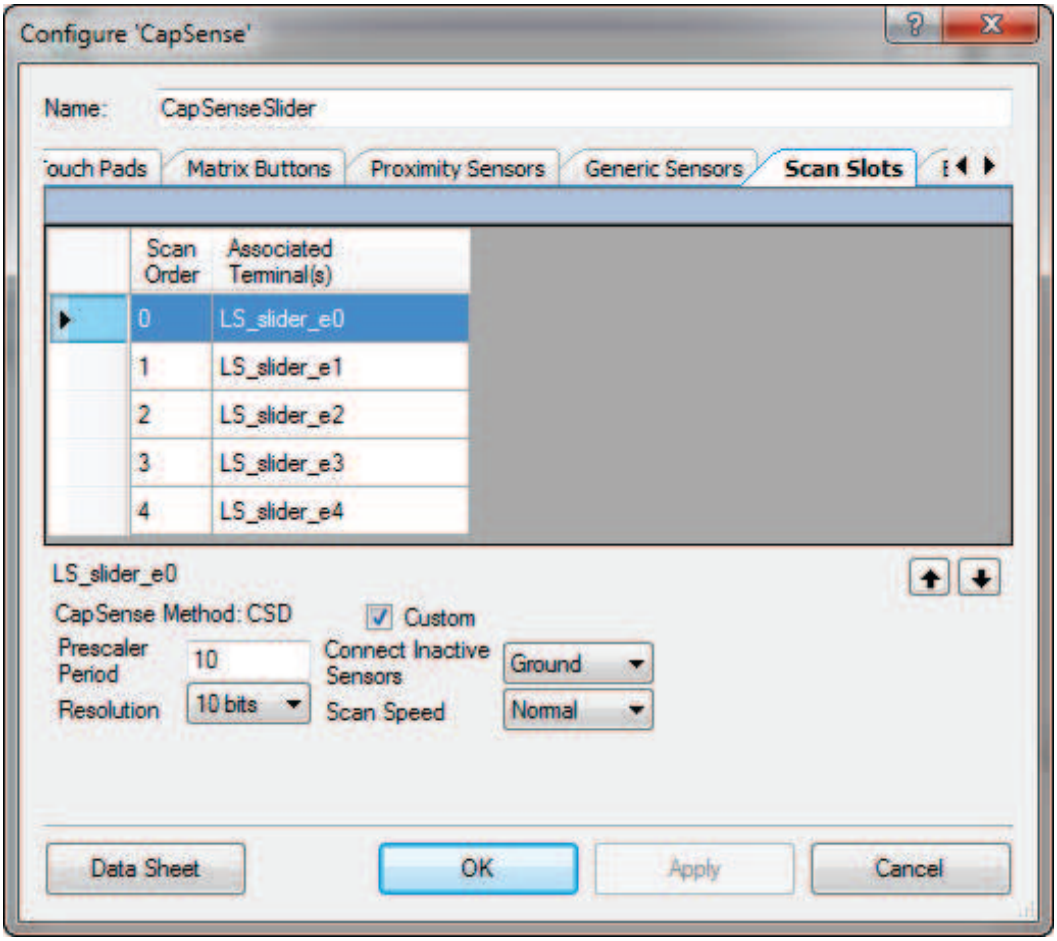


- Configure the Capsense component as shown in the screenshots below.

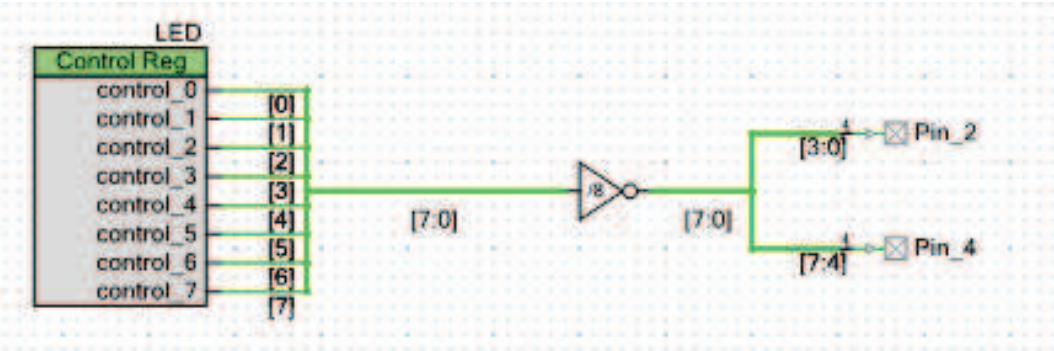



Configure 'CapSense'
Name:
CapSenseSlider
General
Clock Source
Buttons
Sliders
Touch Pads
Matrix Buttons
Prescaler
UDB
CPS_CLK
BUS_CLK : 24 MHz
PreScaler
UDB 8-bit counter
Hardware 16-bit counter
SWCLK
CPS_CLK
ScanSpeed
UDB 8-bit counter
SAMPCLK
Data Sheet
OK
Apply
Cancel


Configure 'CapSense'
Name:
CapSenseSlider
General
Clock Source
Buttons
Sliders
Touch Pads
Matrix Buttons
Slider Name
Type
Number of Elements
Resolution
Diplexing
slider
Linear
5
64
Data Sheet
OK
Apply
Cancel

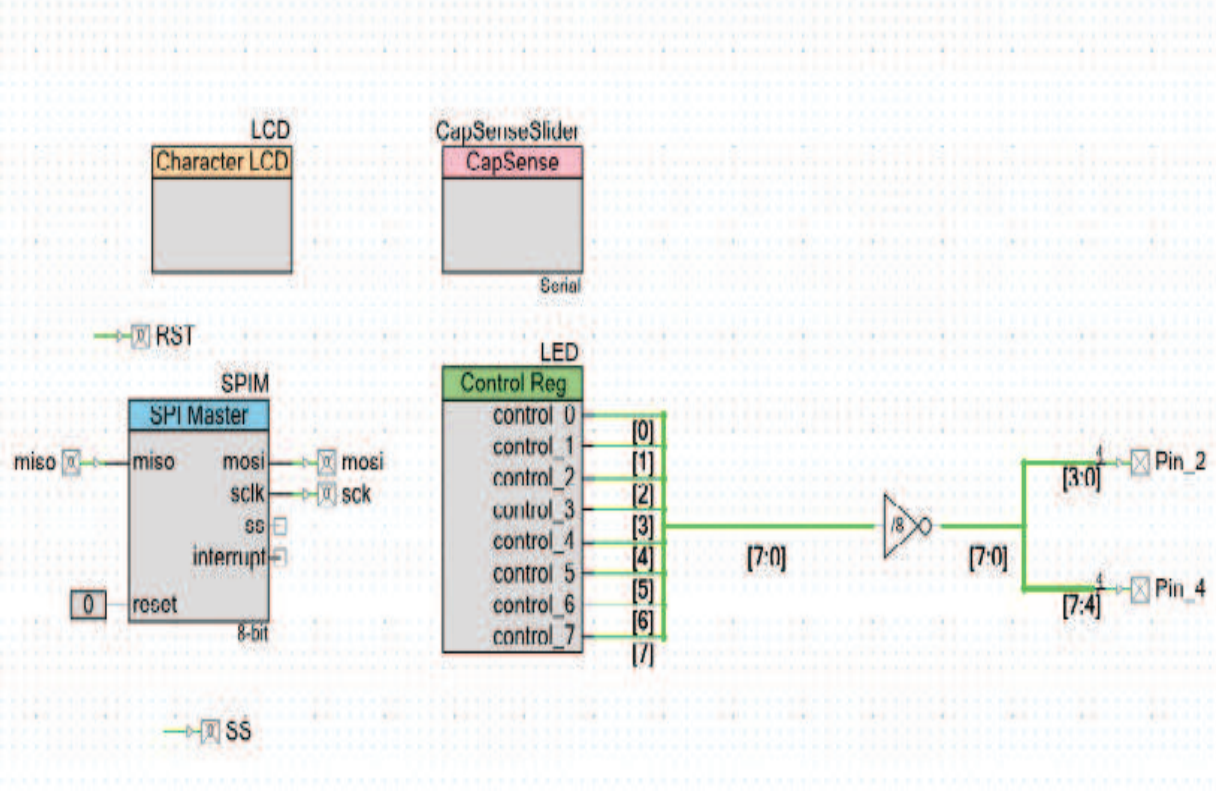


- Add 8 LEDs and connect them to the pins as shown.



- Add a LCD component from Display>Character LCD.

- The final schematic for the transmitter should look as shown in the screenshot below.



- Add the following files to the project

lpregs.h

radio.h

radio_fn.h

radio.c

radio_fn.c

WRITING FIRMWARE (TRANSMITTER)

- Paste the following code in main.c

```
/* ========================================
 *
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/

#include <device.h>
#include <lpregs.h>
#include <radio.h>
#include <radio_fn.h>

#define NUM_LED (3)
uint8 CentroidPosition=0xFF;
uint8 LedData=0;
uint8 TempVar=0;

void hardware_init(void);
void wireless_init(void);
void wireless_send_data(uint8 payload);
uint8 slider_data(void);

uint8 get_width(uint8 volt);

void main()
{
uint8 payload;
uint8 adc_pot=0;

hardware_init();
wireless_init();

CYGlobalIntEnable;
CapSenseSlider_Start();
CapSenseSlider_CSHL_InitializeAllBaselines();

LCD_Position(0,0);LCD_PrintString("CH:");LCD_Position(0, 3);
LCD_PrintInt8(MyRadioGetChannel());

for(;;)
{
adc_pot=slider_data();
payload=get_width(adc_pot);
LED_Write(payload);
wireless_send_data(payload);
//LCD_Position(1,13); LCD_PrintInt8(adc_pot);
```

```
//wireless_send_data(adc_pot);
}
}


uint8 get_width(uint8 volt)
{
if          (volt<0x03) return 5;   //03 0x0C
else if     (volt<0x06) return 6;   //06 0x19
else if     (volt<0x09) return 7;   //09 0x26
else if (volt<0x0C) return 8; //12 0x32
else if     (volt<0x0F) return 9;   //15 0x3e
else if (volt<0x12) return 10;      //18 0x4b
else if     (volt<0x15) return 11;  //21 0x57
else if (volt<0x18) return 12;      //24 0x64
else if     (volt<0x1B) return 13;  //27 0x70
else if (volt<0x1E) return 14;      //30 0x7d
else if (volt<0x21) return 15;      //33 0x89
else if     (volt<0x24) return 16;  //36 0x96
else if (volt<0x27) return 17;      //39 0xA2
else if     (volt<0x2A) return 18;  //42 0xAF
else if (volt<0x2D) return 19;      //45 0xBB
else if     (volt<0x30) return 20;  //48 0xC8
else if (volt<0x33) return 21;      //51 0xD4
else if     (volt<0x36) return 22;  //54 0xE1
else if (volt<0x39) return 23;      //57 0xED
else if     (volt<0x3C) return 24;  //60 0xFA
else if (volt>=0x3C) return 25;     //60 0xFA
}

uint8 slider_data(void)
{
CapSenseSlider_CSD_ScanAllSlots();
CapSenseSlider_CSHL_UpdateAllBaselines();

/* Get Centroid position of the finger on the slider */
CentroidPosition =
(uint8)CapSenseSlider_CSHL_GetCentroidPos(CapSenseSlider_CSHL_LS_SLIDER);

/* If a finger is detected on the slider then turn on the associated LED*/
LedData = 0;
if(CentroidPosition != 0xFF)
{
/* Find the finger position on slider based on 8 LEDs of the total resolution of 64
counts */
LedData = 1 << (CentroidPosition >> NUM_LED);
}

/* Write to the LED control register and update LED status*/
// LED_Control_Reg_Write(LedData);
return CentroidPosition;
}

void hardware_init(void)
{
LCD_Start();
}
```

```
void wireless_init(void)
{
SPIM_Start();
MyRadioHWReset(); //MyRadioWrite(0x1D,0x00); //Reset in SW
CyDelay(100);

//////////////////////////////////////////
//Write data=0x45 to Tx buffer
WritePayloadToRadio(0x01);
MyRadioWrite(TX_CTRL_ADR,TX_GO);//TX_CTRL_ADR,TX_GO
CyDelay(100);//wait
//Setting To idle mode
MyRadioWrite(TX_CTRL_ADR,TX_CLR);//TX_CTRL_ADR,clear TX_BUFFER
CyDelay(100);//wait
//////////////////////////////////////////
MyRadioInit(DATMODE_64SDR);
//Transmit length
MyRadioWrite(TX_LENGTH_ADR,0x01);
}

void wireless_send_data(uint8 payload)
{
unsigned char Status = 0;
MyRadioWrite(TX_CTRL_ADR,TX_CLR);//TX_CTRL_ADR,clear TX_BUFFER
CyDelay(100);//wait

WritePayloadToRadio(payload);                    //set to i later
MyRadioWrite(TX_CTRL_ADR,TX_GO);          //TX_CTRL_ADR,TX_GO
CyDelay(100);//wait

while (!(( Status = MyRadioRead(TX_IRQ_STATUS_ADR)) & TXC_IRQ));
CyDelay(100);//wait
}
```

ASSIGNING PINS (TRANSMITTER)

- Assign the Pinouts as shown.

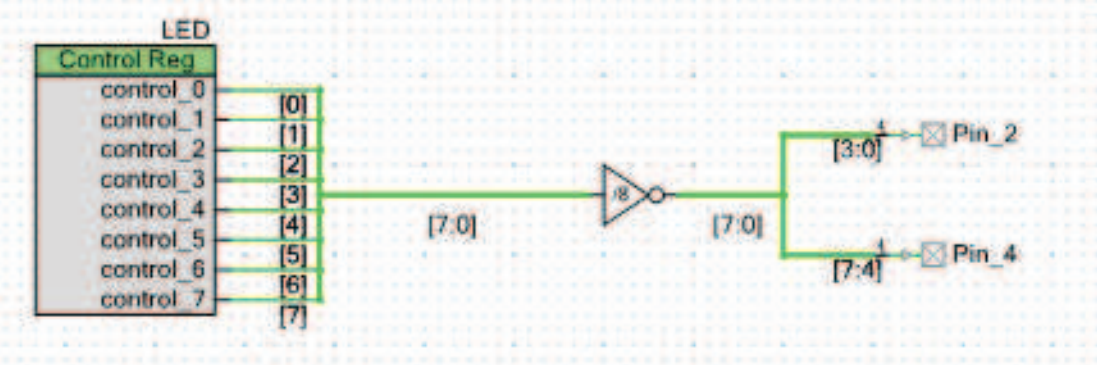
Alias	Name	Pin
	\LCD:LCDPort\[6:0]	P0[6:0]
	SS	P5[3]
	mosi	P12[0]
	sck	P5[0]
	miso	P1[4]
	RST	P5[2]
	Pin_2[3:0]	P2[3:0]
	Pin_4[3:0]	P4[3:0]
sCmod	\CapSenseSlider:sbCSD:cCmod\	P5[4]
sRb0	\CapSenseSlider:sbCSD:cRb0\	P1[6]
LS_slider_e4	\CapSenseSlider:sbCSD:cPort\[4]	P3[4]
LS_slider_e3	\CapSenseSlider:sbCSD:cPort\[3]	P3[3]
LS_slider_e2	\CapSenseSlider:sbCSD:cPort\[2]	P3[2]
LS_slider_e1	\CapSenseSlider:sbCSD:cPort\[1]	P3[1]
LS_slider_e0	\CapSenseSlider:sbCSD:cPort\[0]	P3[0]

PROGRAMMING THE PSOC

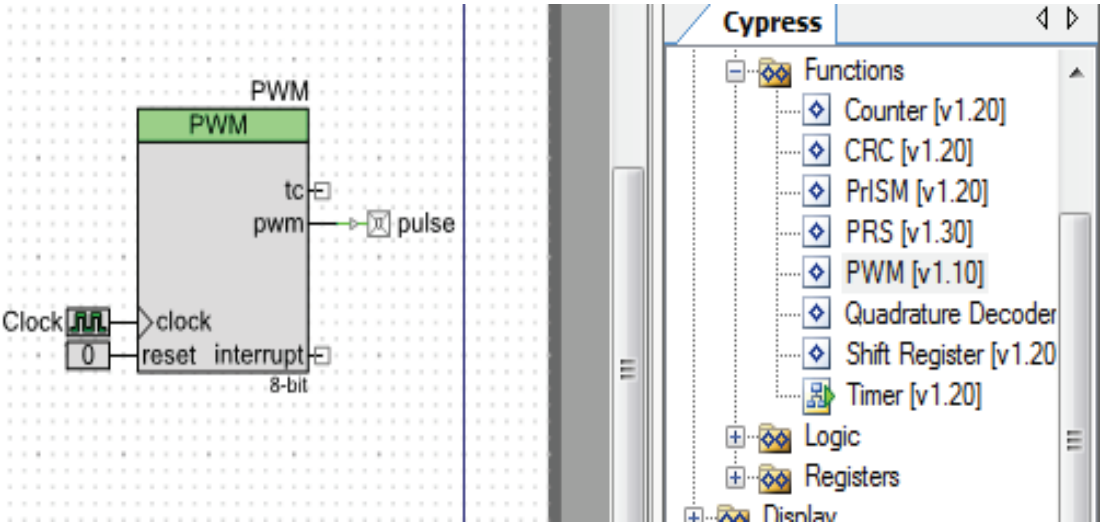
- Build the design and ensure it is error free.
- Plug in the USB Connector and click on the program option. 
- **Now continue with the "Receive" Project we had saved earlier.**

HARDWARE CONFIGURATIONS (RECEIVER)

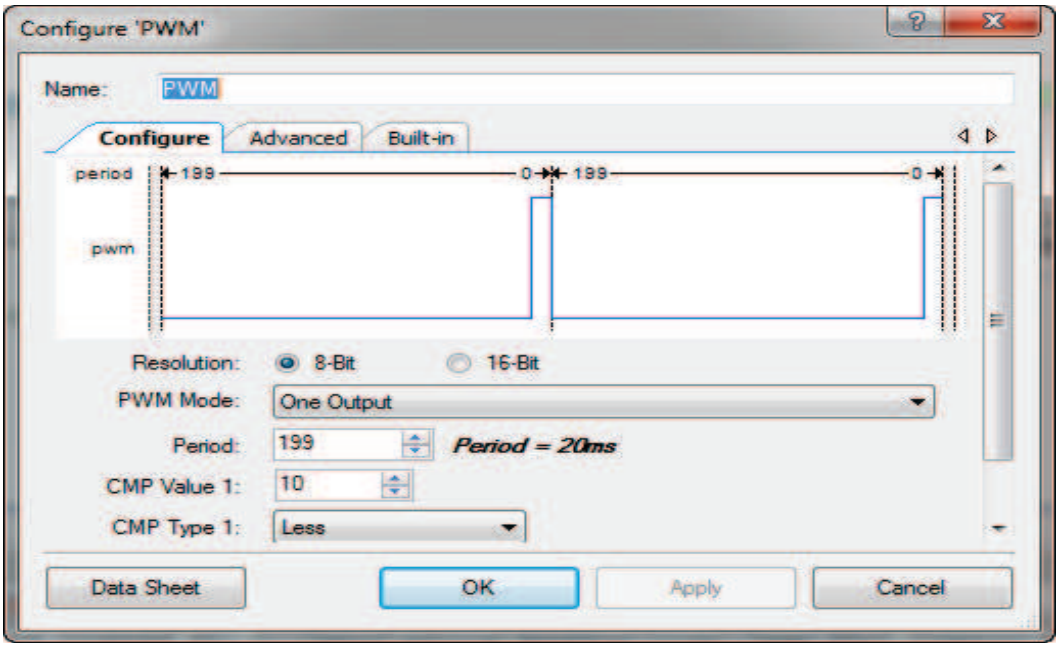
- Add 8 LEDs and connect them to the pins as shown connected to a control register.



- Add a LCD component from Display>Character LCD.
- Add a PWM Component and assign pins and clock as shown (same as the PWM_ADC Project)

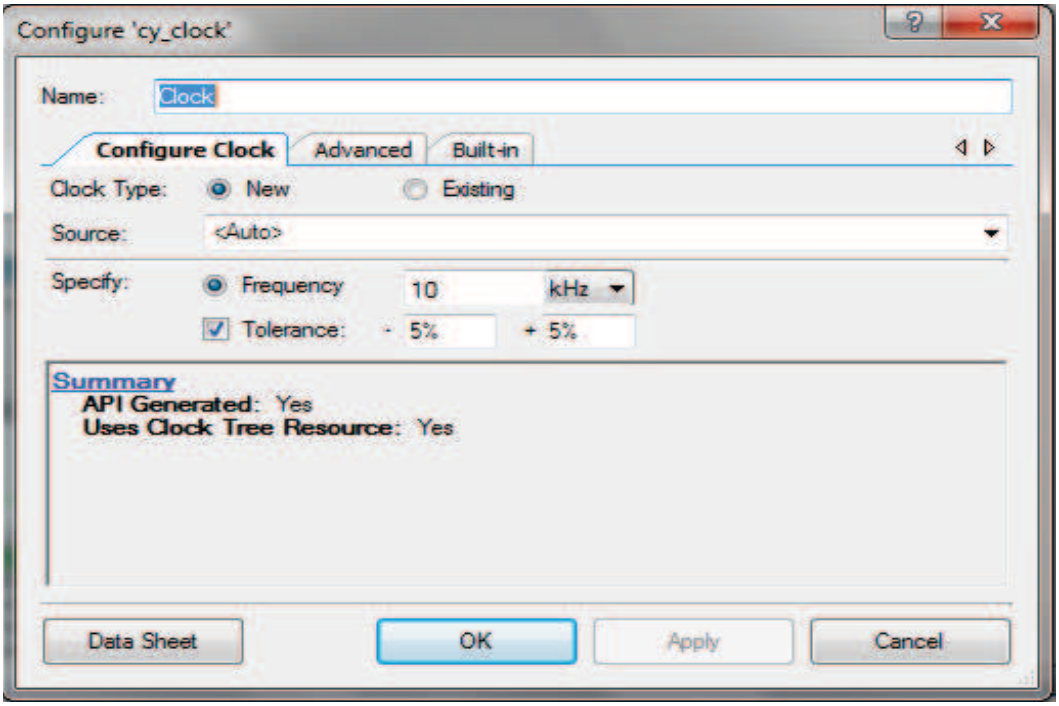


➢ Configure the PWM Component as shown.

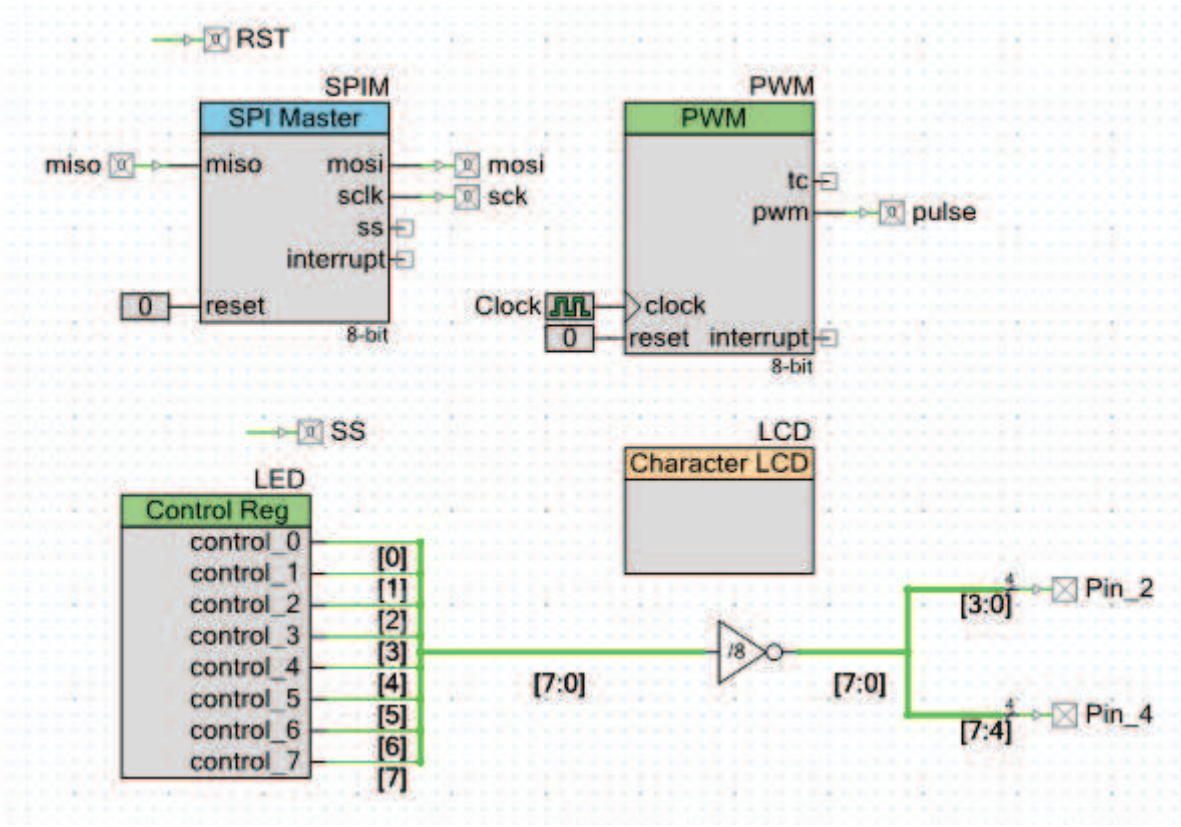


➢ Connect the following peripherals to the PWM block Configure each as shown

- Clock - from System> Clock to be connected to PWM clock and configured as shown below.
- Logic 0 to be connected to reset
- Digital output pin to be named as pulse.



➢ Your final receiver schematic should look as shown in the screenshot below.



➢ Add the following files to the project

lpregs.h

radio.h

radio_fn.h

radio.c

radio_fn.c

WRITING YOUR FIRMWARE (RECEIVER)

- Paste the following code in main.c

```
/* ========================================
 *
 * Copyright PSoC 3 LABBOOK
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION BELONGING TO
 * ADITYA YADAV
 * ASHWATH KRISHNAN
 * ========================================
*/

#include <device.h>
#include <lpregs.h>
#include <radio.h>
#include <radio_fn.h>

void hardware_init(void);
void wireless_init(void);
uint8 clear_payload(uint8 volt);
uint8 wireless_get_data(void);
uint8 old_buffer;

void main()
{
uint8 Payload;
uint8 buffer,old_buffer;
hardware_init();
wireless_init();
wireless_get_data();

LCD_Position(0,0);LCD_PrintString("CH:");LCD_Position(0, 3);
LCD_PrintInt8(MyRadioGetChannel());
old_buffer=5;
for(;;)
{

Payload = wireless_get_data();
buffer=clear_payload(Payload);
if(buffer==0) buffer=old_buffer;
LED_Write(buffer);
PWM_WriteCompare(buffer);
old_buffer=buffer;
}
}

uint8 clear_payload(uint8 volt)
{
if          (volt==5) return 5;     //03 0x0C
else if     (volt==6) return 6;     //06 0x19
else if     (volt==7) return 7;     //09 0x26
```

```
else if (volt==8) return 8;   //12 0x32
else if     (volt==9) return 9;     //15 0x3e
else if (volt==10) return 10; //18 0x4b
else if     (volt==11) return 11;   //21 0x57
else if (volt==12) return 12; //24 0x64
else if     (volt==13) return 13;   //27 0x70
else if (volt==14) return 14; //30 0x7d
else if (volt==15) return 15; //33 0x89
else if     (volt==16) return 16;   //36 0x96
else if (volt==17) return 17; //39 0xA2
else if     (volt==18) return 18;   //42 0xAF
else if (volt==19) return 19; //45 0xBB
else if     (volt==20) return 20;   //48 0xC8
else if (volt==21) return 21; //51 0xD4
else if     (volt==22) return 22;   //54 0xE1
else if (volt==23) return 23; //57 0xED
else if     (volt==24) return 24;   //60 0xFA
else if (volt==25) return 25;
else return old_buffer;

}

void hardware_init(void)
{
LCD_Start();
SPIM_Start();
PWM_Start();
}

void wireless_init(void)
{
MyRadioHWReset(); //MyRadioWrite(0x1D,0x00); //Reset in SW
CyDelay(100);
MyRadioInit(DATMODE_64SDR);
//Receive length
MyRadioWrite(RX_LENGTH_ADR,0x01);
}

uint8 wireless_get_data(void)
{
uint8 Payload;
MyRadioStartReceive();
CyDelay(100);//wait
//Setting To idle mode
Payload = MyRadioRead(RX_BUFFER_ADR);
//MyRadioWrite(RX_CTRL_ADR,RX_CLR);//TX_CTRL_ADR,clear TX_BUFFER
CyDelay(100);//wait
return Payload;
}
```

ASSIGNING PINS (RECEIVER)

- Assign the PinOuts as shown in the screenshot below.

Alias	Name	Pin	Loc
	\LCD:LCDPort\[6:0]	P0[6:0]	☑
	SS	P5[3]	☑
	mosi	P12[0]	☑
	sck	P5[0]	☑
	miso	P1[4]	☑
	RST	P5[2]	☑
	Pin_2[3:0]	P2[3:0]	☑
	Pin_4[3:0]	P4[3:0]	☑
	pulse	P12[2]	☑

PROGRAMMING THE PSOC

- Build the design and ensure it is error free.

- Plug in the USB Connector and click on the program option. 

TESTING YOUR DESIGN

- Connect the Artaflex Module to the 12 Pin port to both kits.
- Connect the servo motor to the PSoC programmed with Capsense_Receive.
- Power both the kits. The motor kit should be powered using 5V.
- Slide the finger on the Controller Kit and the motor position will change and so will the LEDs on both the kits.

NOTES

NOTES

NOTES