The PSoC 5LP LABBOOK

1st Edition

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

Ahmed BOUZID

University of Miskolc
Department of Automation and Communication Technology

May 2018

Open Source Licence to Use and Reproduce

ACKNOWLEDGEMENTS

This document falls within the context of practical works for embedded systems subject organized for students of the University of Miskolc. I thank Dr. József Vásárhelyi for his precious help to elaborate this work.

Ahmed BOUZID (qgebouzid@uni-miskolc.hu)

June 2016

PREFACE

Embedded systems were initiated through aerospace needs, especially for the Apollo Guidance Computer. Real time processing and miniaturization are the key elements of an embedded system since it previously was not possible to embed controllers on vehicles because of big sizes.

In major cases, general purpose architectures are underdimensionned or overdimensionned solutions. In order to optimize architecture according to a specific application, design via reconfigurable systems is an alternative solution where architecture is in adequacy with algorithm.

Nowadays, PSoC® is a family of integrated circuits that have high presence on the market of reconfigurable systems. PSoC (Programmable System on Chip) is a family of integrated circuits introduced by Cypress Semiconductor in the beginning of 2000. Each PSoC IC has a microcontroller and some configurable analog and digital blocks. These components are programmably routed and interconnected using PSoC Designer (for PSoC1 family) or PSoC Creator (for PSoC 3, 4, 5 and 5LP families)

The document presents lab materials and a mini-project using CY8CKIT-050 Development Kit and some discrete components. This kit is based on CY8C5868AXI-LP035 chip that includes ARM Cortex-M3 microcontroller.

TABLE OF CONTENTS

LAB1

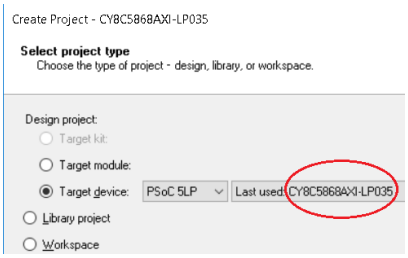
PART I Blinking LED

I.1 Introduction
I.1.1 Overview

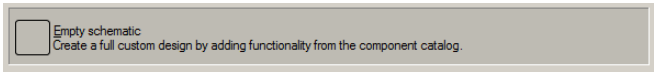
The purpose of this LAB is to build an application using CY8CKIT-050 PSoC development kit for ***LED blinking***. It must light for 50ms and turn off for 50ms.

I.1.2 Creation of the Project

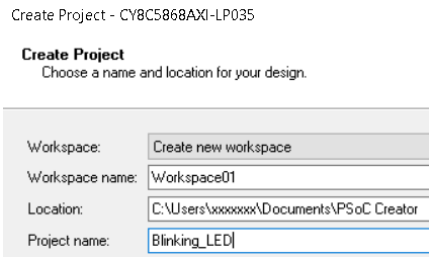
- Open PSoC Creator
- Go to file, new, project
- Choose the target device PSoC5LP CY8CKIT-050 that corresponds to the **CY8C5868AXI-LP035** chip, then click next.



- Choose Empty Schematic and click next

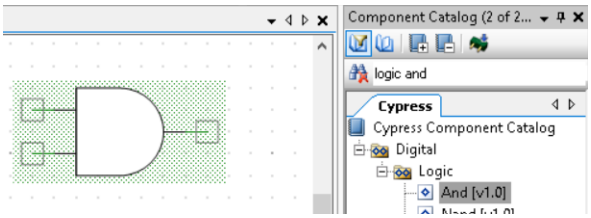


- Workspace : create new workspace
- Location :PSoC Creator
- Project name : Blinking_LED

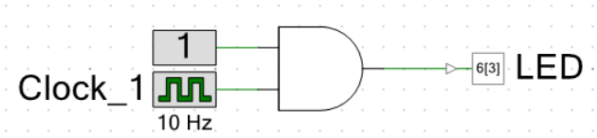


I.2 Hardware configuration
I.2.1 Blocs design

-Go to component catalog right the screen, write "logic and", drag and drop the “And” component to the TopDesign schematic. Make a zoom in by clicking “Ctrl + +”.

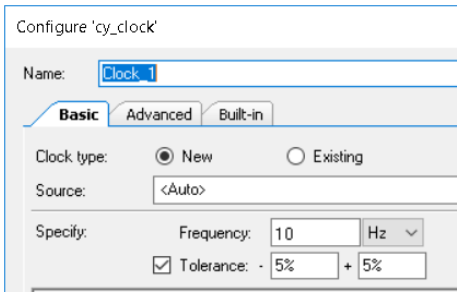


- Look for "digital output pin", drag and drop the component and connect it to the output of the AND component.
- Look for "Clock", drag and drop it and connect it to one of the inputs.
- Look for "Logic High", drag and drop it and connect it to the logic gate.



I.2.2 Configuration of the components

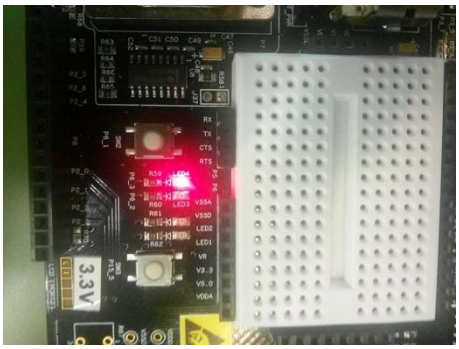
- Double click on "clock", set the frequency to 10 Hz with ±5% tolerance



- At the left of the screen in workspace explorer, double click on Pins under Blinking_LED.cydwr.
- At the right of the screen in Port, choose P6[3]

I.3 Program and results

- Build the design by clicking on build or shift+F6, it will take a while.
- When finish verify warnings and errors.
- Plug the PSoC5LP on J1 mini-USB.
- Click on program or ctrl+F5
- A red LED (LED4) will start blinking every 0.1 seconds.



PART II Blinking LED with PWM component

The purpose of this LAB is to build an application using CY8CKIT-050 PSoC development kit for ***blinking a LED***. It must light for 50ms and turn off for 950ms.

II.1 Creation of the Project

- Open PSoC Creator
- Go to file, new, project
- Choose the target device PSoC5LP CY8CKIT-050 that corresponds to the **CY8C5868AXI-LP035** chip, and then click next.

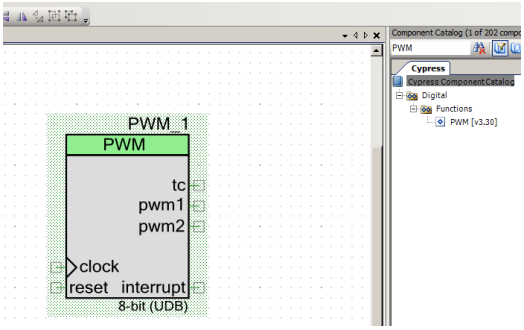
- Choose Empty Schematic and click next

- Workspace: create new workspace
- Location: PSoC Creator
- Project name: Blinking_LED_PWM

II.2 Hardware configuration

II.2.1 Blocs design

-Go to component catalog right the screen, write "PWM", drag and drop the component to the TopDesign schema.

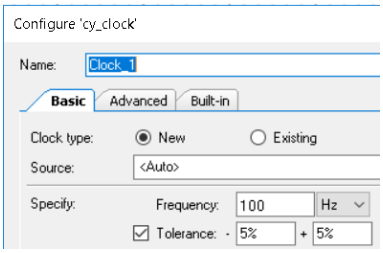


- Look for "pin", choose digital output pin, drag and drop it and connect it to the PWM bloc
- Look for "clock", drag and drop it and connect it to the PWM bloc.

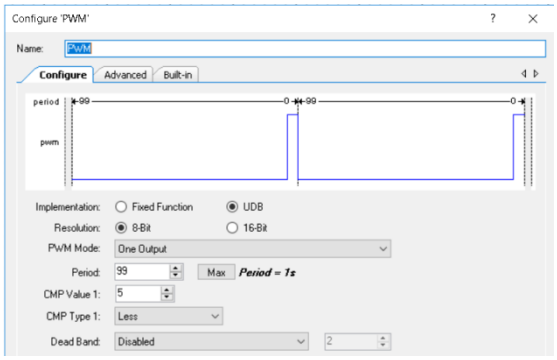
- Look for "Logic Low", drag and drop it and connect it to the reset port of the PWM bloc.

II.2.2 Configuration of the components

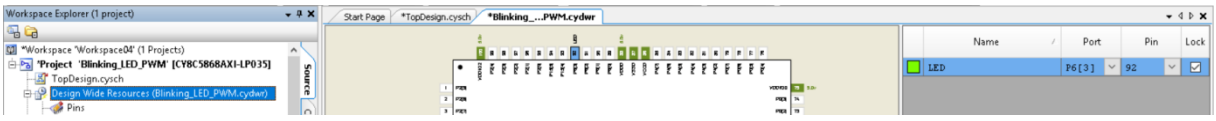
- Double click on "clock", set the frequency to 100 Hz with ±5% tolerance

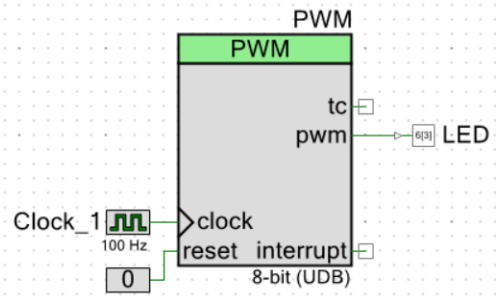


- Double click on PWM, rename it to PWM, set the PWM mode to one output, period to 99 and CMPvalue1 to 5.



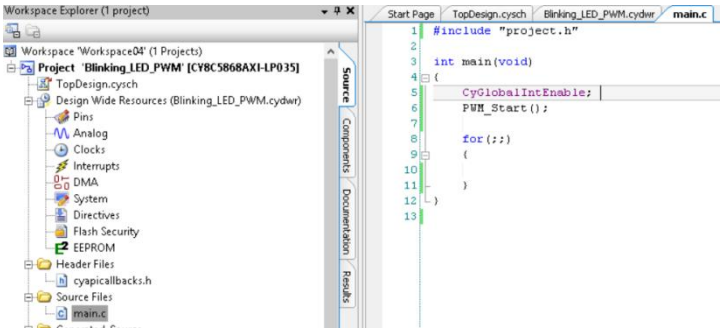
- At the left of the screen in workspace explorer, double click on Blinking_LED_PWM.cydwr.
- At the right of the screen in Port, choose P6[3]





II.3 Software configuration

- Double click on main.c in the workspace explorer under source files
- Under the code line "CyGlobalIntEnable;" write the following: "PWM_Start();"



II.4 Program and results

- Build the design by clicking on build or shift+F6, it will take a while.
- When finish verify warnings and errors.
- Plug the PSoC5LP on J1 mini-USB.
- Click on program or ctrl+F5
- A red LED (LED3) will start blinking every 1 second.

PART III Controlling LED (Hardware Design)

The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***controlling a LED***. It must light when a button is pressed.

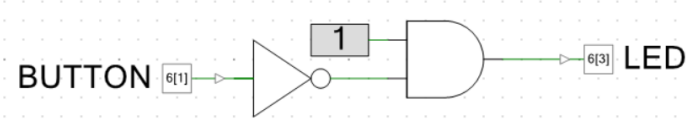
III.1 Creation of the Project

- Open PSoC Creator
- Go to file, new, project
- Choose the target device " **CY8C5868AXI-LP035** " and click next
- Choose Empty Schematic and click next
- Workspace : create new workspace
- Location : PSoC Creator
- Project name : Control_LED_HW

III.2 Hardware configuration

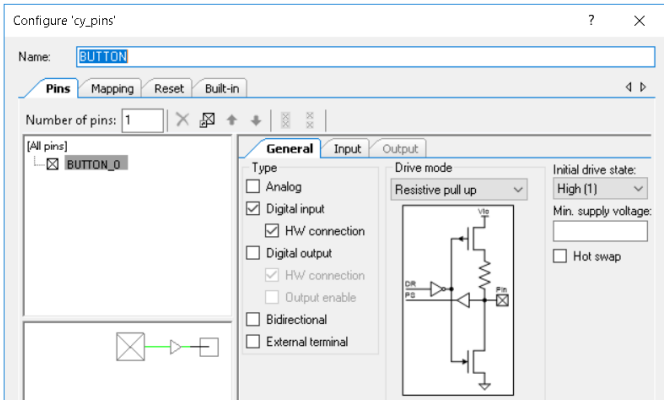
III.2.1 Blocs design

- Look for “Logic And”, drag and drop it.
- Look for “Logic Not”, drag and drop it.
- Look for “Logic High” drag and drop it.
- Look for “Digital Input Pin”, drag and drop it.
- Look for “Digital Output Pin”, drag and drop it.
- Make connections

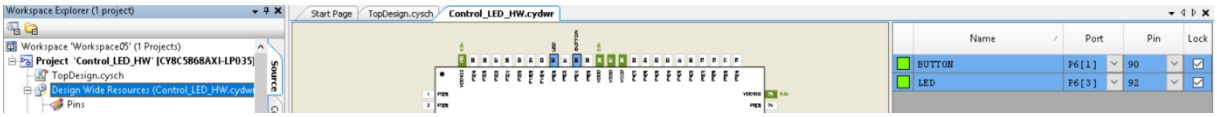


III.2.2 Configuration of the components

- Double click on the input pin, rename it to "BUTTON" and choose **Resistive pull up** Drive mode.



- At the left of the screen in workspace explorer, double click on Pins under Control_LED_HW.cydwr.
- At the right of the screen in Port, choose P6[1] for BUTTON and P6[3] for LED.



III.4 Program and results

- Build the design by clicking on build or shift+F6, it will take a while.
- When finish verify warnings and errors.
- Plug the PSoC5LP on J1 mini-USB.
- Click on program or ctrl+F5
- LED4 lights by clicking on SW2 switch.

PART IV Controlling LED (Hardware/Software Design)

The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***controlling a LED*** using hardware and software. By clicking a button the state of a LED has to change.

IV.1 Creation of the Project

- Open PSoC Creator
- Go to file, new, project
- Choose the target device "**CY8C5868AXI-LP035** " and click next
- Choose Empty Schematic and click next
- Workspace: create new workspace
- Location: PSoC Creator
- Project name: Control_LED_SW

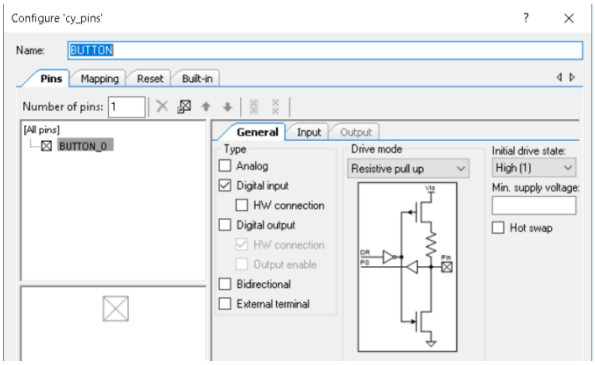
IV.2 Hardware configuration

IV.2.1 Blocs design

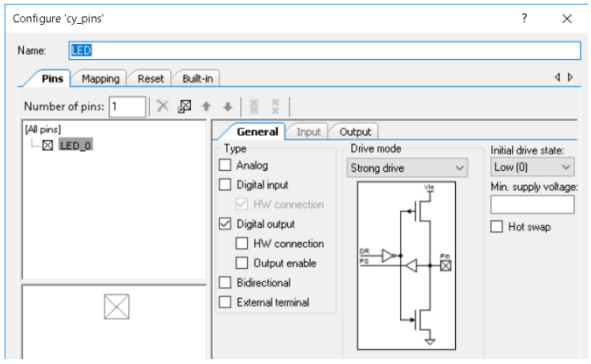
- Look for "pin", choose digital input pin, drag and drop it.
- Look for "pin", choose digital output pin, drag and drop it.

IV.2.2 Configuration of the components

- Double click on the input pin, rename it to "BUTTON", choose **Resistive pull up** Drive mode and uncheck HW connection.



- Double click on the output pin, rename it to "LED", choose **Strong** Drive mode, uncheck HW connection and the initial drive state must be set to **Low**



- At the left of the screen in workspace explorer, double click on Control_LED_SW.cydwr.
- At the right of the screen in Port, choose P6[1] for BUTTON and P6[3] for LED.

IV.3 Software configuration

- Double click on main.c in the workspace explorer under source files
- Write the following code:

Start Page | TopDesign.cysch | Control_LED_SW.cydwr | main.c

```c
#include "project.h"

int main(void)
{
    CyGlobalIntEnable;

    int a=0 ;
    for(;;)
    {
        if (BUTTON_Read() == 0)
        {
            if (a == 0)
            {
                LED_Write(1);
                a = 1;

            }
            else
            {
                LED_Write(0);
                a = 0;

            }
        }
    }
}
```

IV.4 Program and results

- Build, check warnings and errors, plug the target and program it.
- By clicking on SW2 switch in the board, you will notice that sometimes the LED doesn't light well. This is due to the bouncing effect of the mechanical switch.
- It can be solve by adding the component "debouncer" to the design, or simply adding a "delay" to the main program as following:

Start Page | TopDesign.cysch | Control_LED_SW.cydwr | main.c

```c
#include "project.h"

int main(void)
{
    CyGlobalIntEnable;

    int a=0 ;
    for(;;)
    {
        if (BUTTON_Read() == 0)
        {
            if (a == 0)
            {
                LED_Write(1);
                CyDelay(200);
                a = 1;
            }
            else
            {
                LED_Write(0);
                CyDelay(200);
                a = 0;
            }
        }
    }
}
```

PART V Controlling a LED with an interrupt

The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***controlling a LED using an interrupt***. By clicking a button the LED must blink for 0.9s every second, and then by clicking another button the LED stops blinking.

V.1 Creation of the Project

- Open PSoC Creator
- Go to file, new, project
- Choose the target device "**CY8C5868AXI-LP035**" and click next
- Choose Empty Schematic and click next
- Workspace: create new workspace
- Location: PSoC Creator
- Project name: Interrupt_LED

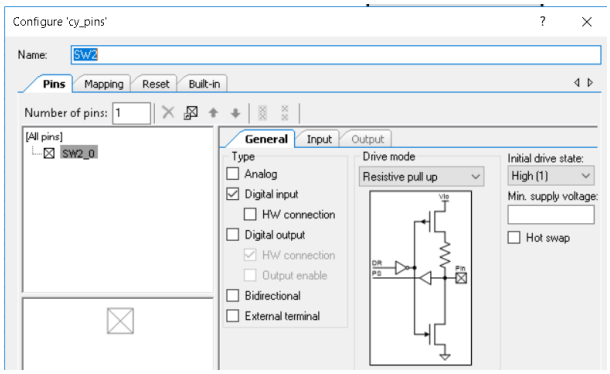
V.2 Hardware configuration

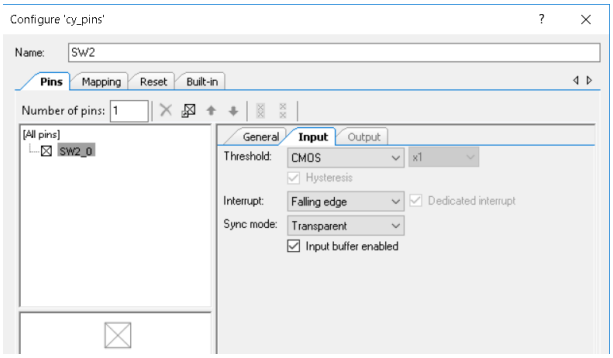
V.2.1 Blocs design

- Look for the following components, drag and drop them to the schema:
PWM, clock, Logic Low, Digital Input Pin (twice), Interrupt (twice), Digital Output Pin.

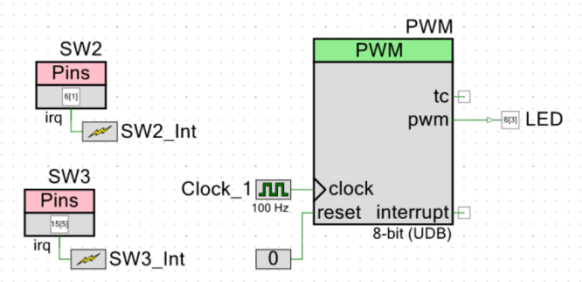
V.2.2 Configuration of the components

- Double click on the input pin, rename it to "SW2", choose **Resistive pull up** Drive mode and uncheck HW connection. Under **Input** tab, set **Falling edge** for **Interrupt.**





- Double click on the input pin, rename it to "SW3", choose **Resistive pull up** Drive mode and uncheck HW connection. Under **Input** tab, set **Falling edge** for **Interrupt.**
- Double click on the output pin, rename it to "LED", choose **Strong** Drive mode and the initial drive state must be set to **High.**
- Double click on "clock", set the frequency to 100 Hz with ±5% tolerance
- Double click on PWM, rename it to PWM, set the PWM mode to one output, period to 99 and CMPvalue1 to 90
- At the left of the screen in workspace explorer, double click on Pins under Interrupt_LED.cydwr.
- At the right of the screen in Port, choose P6[1] for SW2, P15[5] for SW3 and P6[3] for LED
- Change the names of the interrupt components to "SW2_Int" and "SW3_Int"
- Make connections between the components as following:



V.3 Software configuration

- Double click on main.c in the workspace explorer under source files
- Write the following code:

Start Page | TopDesign.cysch | Interrupt_LED.cydwr | main.c

```
#include<project.h>

CY_ISR(SW2_Handler)
{
    SW2_ClearInterrupt();
    PWM_Start();
}

CY_ISR(SW3_Handler)
{
    SW3_ClearInterrupt();
    PWM_Stop();
}

int main()
{
    CyGlobalIntEnable;

    SW2_Int_StartEx( SW2_Handler );
    SW3_Int_StartEx( SW3_Handler );

    for(;;)
    {
    }
}
```

V.4 Program and results

- Build, check warnings and errors, plug the target and program it.
- The LED light by clicking on SW2 and switch of by clicking on SW3.

LAB2

PART I Maximal value indicator (with comparator)

I.1 Introduction
I.1.1 Overview

The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***Lighting a LED using a potentiometer***. The LED must light when the potentiometer is at the maximum position.

I.1.2 Creation of the Project

- Open PSoC Creator
- Go to file, new, project
- Choose the target device " **CY8C5868AXI-LP035** " and click next
- Choose Empty Schematic and click next
- Workspace: create new workspace
- Location: PSoC Creator
- Project name: Comp_LED

I.1.3 Required Instrumentation

- CY8CKIT-050 is a PSoC 5LP development kit from CYPRESS based on CY8C5868-AXI-LP035 chip.
- 2x16 LCD Character Display.

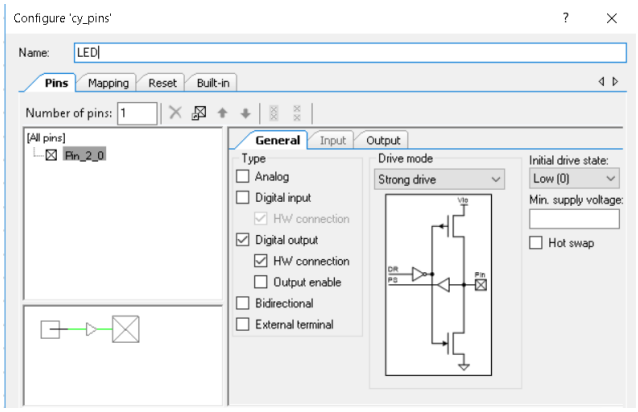
I.2 Hardware configuration
I.2.1 Blocs design

- Look for the following components, drag and drop them to the schema:
Comparator, Analog Pin, Digital Output Pin, VRef.

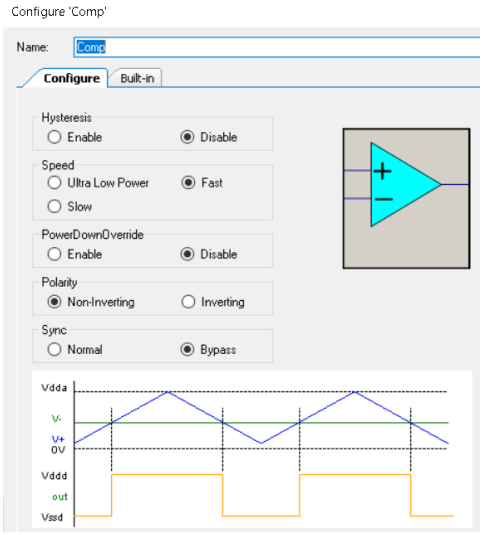
I.2.2 Configuration of the components

- Double click on the input pin, rename it to "Potar", choose **High impedance analog** Drive mode, set initial drive state to Low.

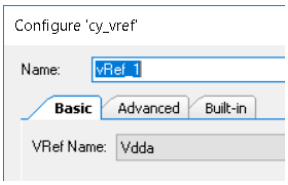
- Double click on the output pin, rename it to "LED", choose **Strong Drive** mode.

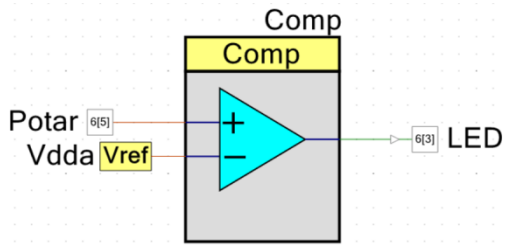


- Double click on the comparator, rename it to **Comp**, disable Hysteresis, set the speed to fast and sync must be bypassed.

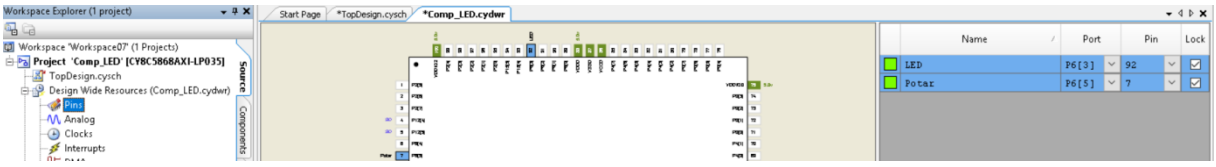


- Double click to VRef and set the value to Vdda.





- At the left of the screen in workspace explorer, double click on Pins under Comp_LED.cydwr.
- At the right of the screen in Port, choose P6[3] for LED and P6[5] for Potar.



I.3 Software configuration

- Double click on main.c in the workspace explorer under source files
- Enable the comparator by putting the command Comp_Start();

I.4 Program and results

- Build the design by clicking on build or shift+F6, it will take a while.
- When finish verify warnings and errors.
- Plug the PSoC5LP on J1 mini-USB.
- Click on program or ctrl+F5
- The LED light when the potentiometer is turned to the maximum position.

I.5 Exercise

The LED must light when the potentiometer is at the middle position.

PART II Maximal value indicator (with ADC)

II.1 Introduction
II.1.1 Overview

The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***Lighting a LED using a potentiometer***. The LED must light when the potentiometer is at the maximum position using ADC without comparator. The state of the LED must be displayed on an LCD.

II.1.2 Creation of the Project

- Open PSoC Creator
- Go to file, new, project
- Choose the target device " **CY8C5868AXI-LP035** " and click next
- Choose Empty Schematic and click next
- Workspace : create new workspace
- Location :PSoC Creator
- Project name :ADC_LED

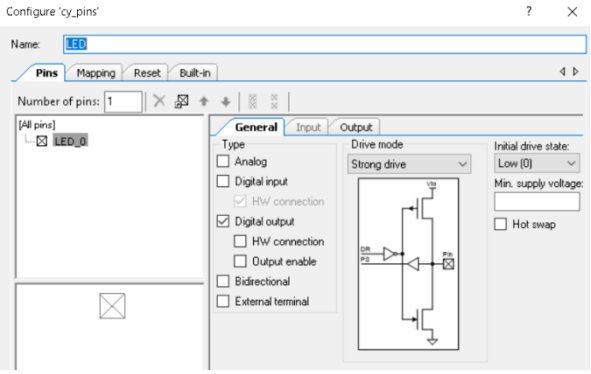
II.2 Hardware configuration
II.2.1 Blocs design

- Look for the following components, drag and drop them to the schema:
SAR ADC, Analog Pin, Digital Output Pin, LCD.

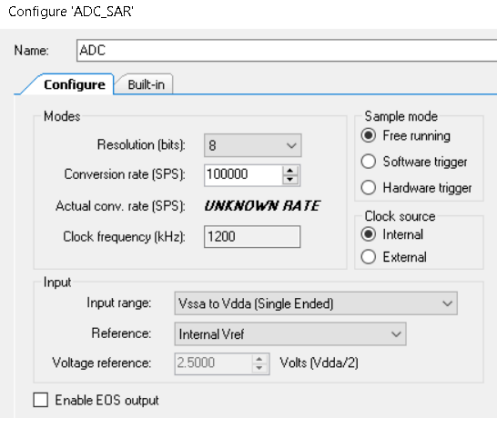
II.2.2 Configuration of the components

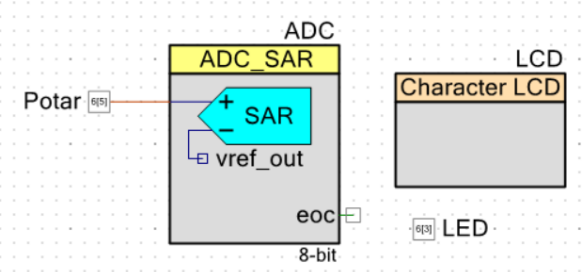
- Double click on the input pin, rename it to "Potar", choose **High Impedance analog** Drive mode.

- Double click on the output pin, rename it to "LED", choose **Strong Drive mode**, **uncheck** HW connection and the initial drive state must be set to **Low.**

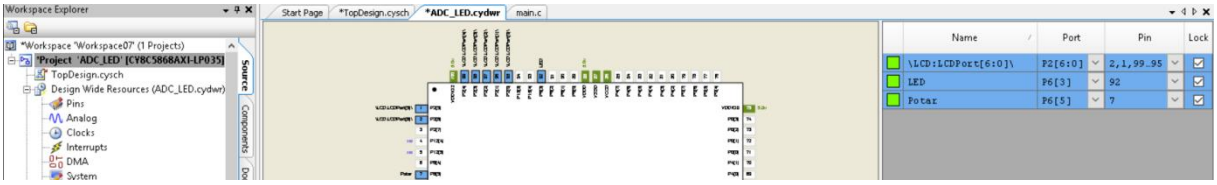


- Double click on the ADC, rename it to ADC, set the resolution to 8 bits, the conversion rate to 100000, the input range must be set to **Vssa to Vdda (single Ended)** and choose **internal Vref** as reference source.





- At the left of the screen in workspace explorer, double click on Pins under ADC_LED.cydwr.
- At the right of the screen in Port, choose P6[3] for LED, P6[5] for Potar and P2[6:0] for LCD.



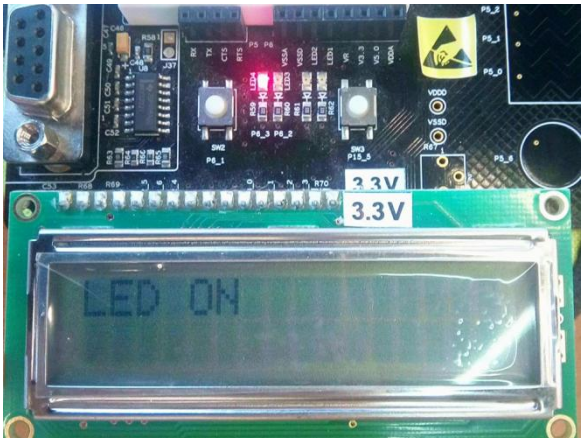
II.3 Software configuration

- Double click on main.c in the workspace explorer under source files
- Write the following code:

```
#include<project.h>

int main()
{
    ADC_Start();
    LCD_Start();

    int16 Potar_val;

for(;;)
    {
        ADC_StartConvert();
        ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
        Potar_val = ADC_GetResult16();

        if(Potar_val> 254)
        {

            LED_Write(1);
            LCD_Position(0u, 0u);
            LCD_PrintString("LED ON ");
            }
        else
        {

            LED_Write(0);
            LCD_Position(0u, 0u);
            LCD_PrintString("LED OFF");
        }
    }
}
```

II.4 Program and results

- Build the design by clicking on build or shift-F6, it will take a while.
- When finish verify warnings and errors.
- Plug the LCD on the board **according to pins numbers (!!!otherwise the LCD deteriorates !!!)**
- Plug the PSoC5LP on J1 mini-USB.
- Click on program or ctrl+F5
- The LED light when the potentiometer is turned to the maximum position and the LCD displays the state of the LED.



PART III Changing Speed and Brightness of Blinking LED

III.1 Introduction

The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***controlling the speed and the brightness of LED Blinking using a potentiometer and display the values on LCD***.

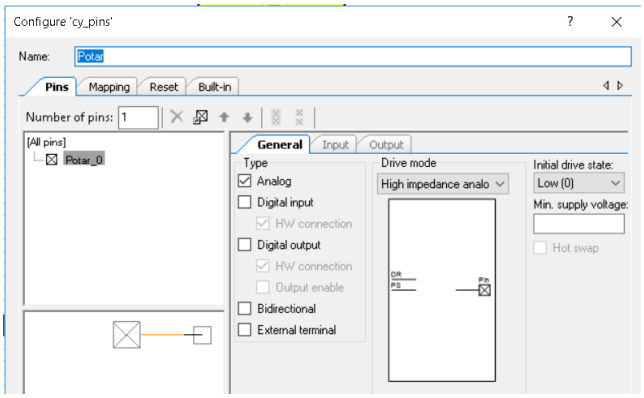
III.2 Hardware configuration

III.2.1 Blocs design

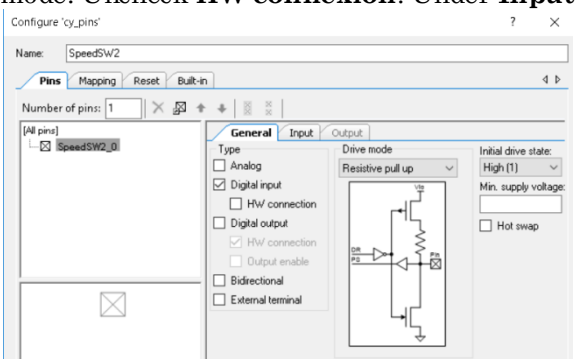
- Look for the following components, drag and drop them to the schema:
PWM, , SAR ADC, LCD, clock, Logic Low, Digital Input Pin (3 times), Analog Pin, Digital Output Pin, Interrupt (twice).

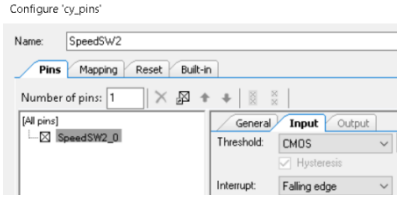
III.2.2 Configuration of the components

- Double click on the analog input pin, rename it to "Potar", choose **High Impedance analog** Drive mode.



- Double click on a Digital input pin, rename it to "SpeedSW2" and choose **Resistive Pull up** Drive mode. Uncheck **HW connexion**. Under **Input** tab, select **Falling Edge** Interrupt

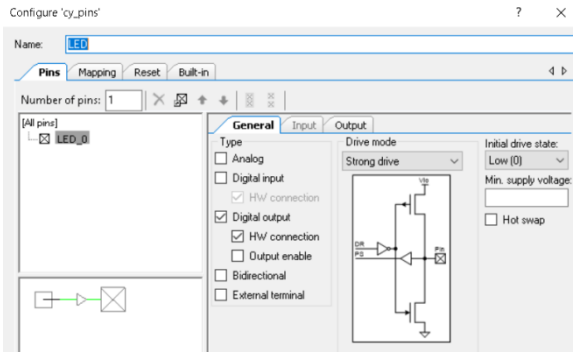




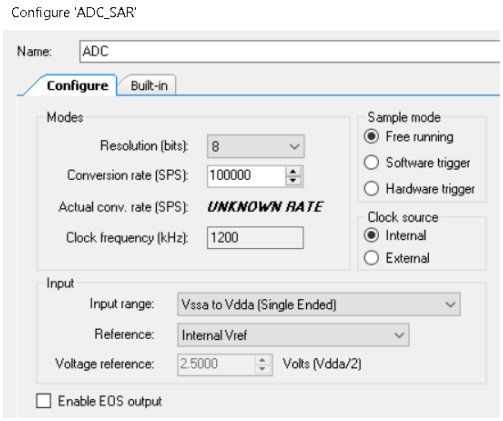
- Double click on another Digital input pin, rename it to "BrightSW3" and choose **Resistive Pull up** Drive mode. Uncheck **HW connexion**. Under **Input** tab, select **Falling Edge** Interrupt

- Double click on the remaining Digital input pin, rename it to “Reset” and choose **Pull up** Drive mode. Uncheck **HW connexion.**

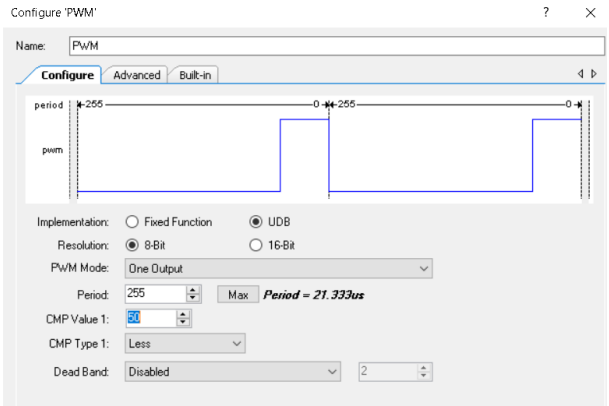
- Double click on the output pin, rename it to "LED", choose **Strong Drive mode**, the initial drive state must be set to **Low.**



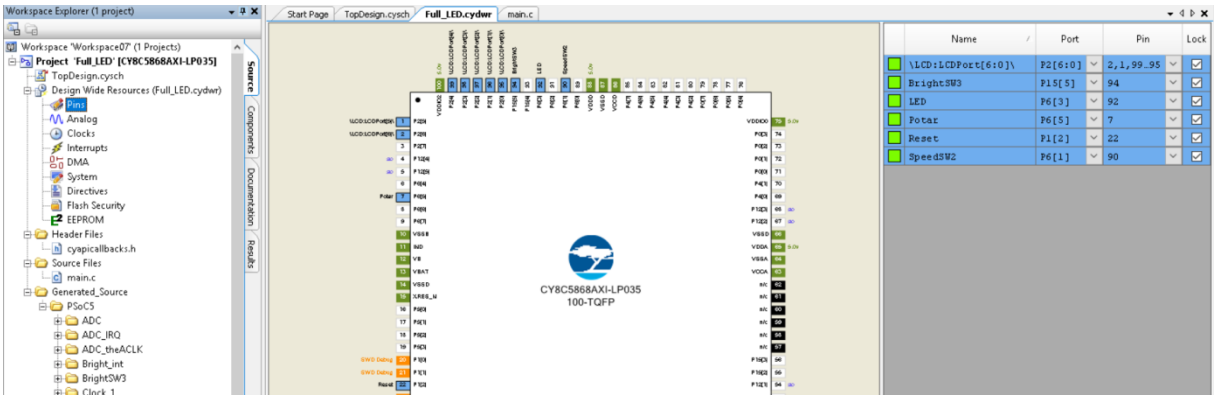
- Double click on the ADC, rename it to ADC, set the resolution to 8 bits, the conversion rate to 100000, the input range must be set to **Vssa to Vdda (single Ended)** and choose **internal Vref** as reference source.



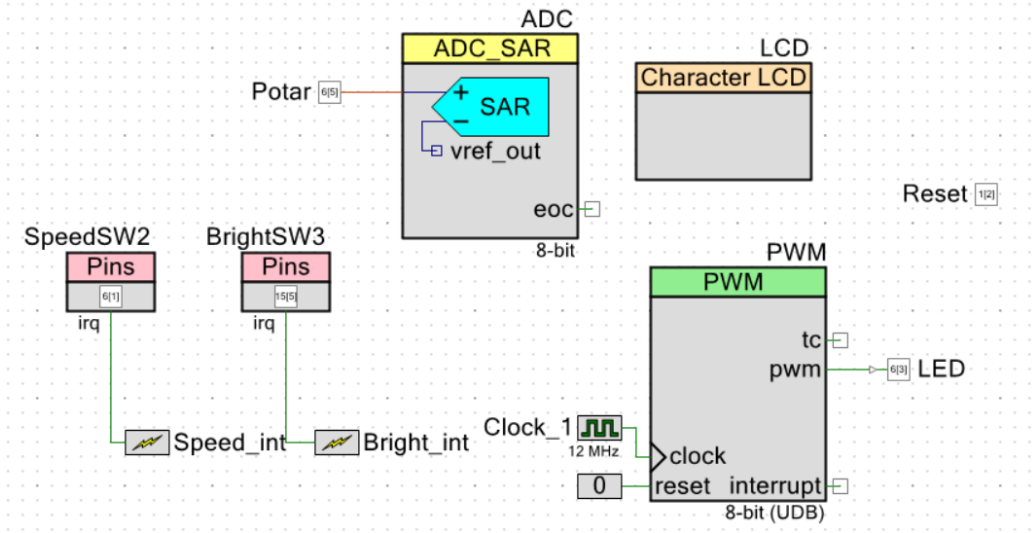
- Double click on PWM, rename it to “PWM” then set PWM mode to one output.



- At the left of the screen in workspace explorer, double click on Pins under Full_LED.cydwr.
- At the right of the screen in Port, set ports as following:



- Change the names of the interrupt components to "Speed_Int" and "Bright_Int"
- Make connections between the components as following:



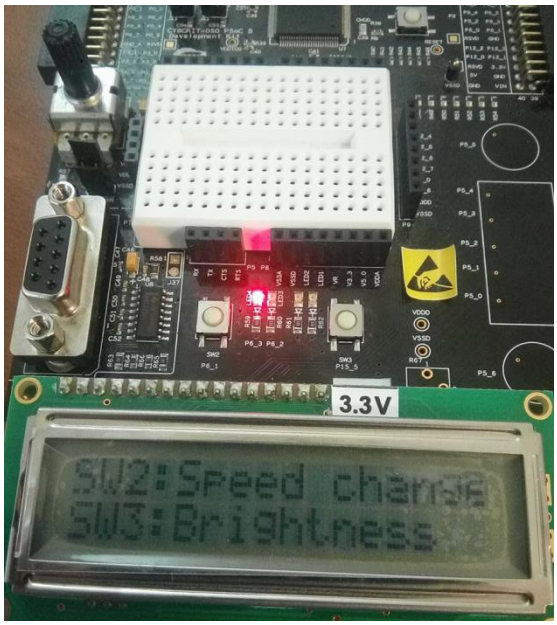
III.3 Software configuration

- Double click on main.c in the workspace explorer under source files
- Write the following code:

```c
#include<project.h>

unsigned char speedval = 250;
unsigned char brightval = 250;

CY_ISR(SpeedSW2_Handler)
{
    SpeedSW2_ClearInterrupt();
    CyDelay(200);

    while (SpeedSW2_Read() == 1)
{

    ADC_StartConvert();
    ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
    speedval = ADC_GetResult8();

    LCD_ClearDisplay();
    LCD_Position(0u, 0u);
    LCD_PrintString("Blinking Speed");
    LCD_Position(1u, 0u);
    LCD_PrintNumber(speedval);
    CyDelay(10);
    }
}

CY_ISR(BrightSW3_Handler)
{
    BrightSW3_ClearInterrupt();
    CyDelay(200);

    while (BrightSW3_Read() == 1)
    {
        ADC_StartConvert();
        ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
        brightval = ADC_GetResult8();

        LCD_ClearDisplay();
        LCD_Position(0u, 0u);
        LCD_PrintString("Brightness");
        LCD_Position(1u, 0u);
        LCD_PrintNumber(brightval);
        CyDelay(10);
    }
}
int main()
{
    CyGlobalIntEnable;

    Speed_int_StartEx(SpeedSW2_Handler );
    Bright_int_StartEx(BrightSW3_Handler );

    ADC_Start();
    LCD_Start();

    for(;;)
    {

        PWM_Start();
        PWM_WritePeriod(255);
        PWM_WriteCompare(brightval);
        LCD_Position(0u, 0u);
        LCD_PrintString("SW2:Speed change");
        LCD_Position(1u, 0u);
        LCD_PrintString("SW3:Brightness");
        CyDelay(speedval);

        PWM_Stop();

        CyDelay(speedval);
    }
}
```

III.4 Program and results

- !!! <u>Before</u> powering the board connect carefully the LCD screen !!!
- Build, check warnings and errors, plug the target and program it.
- The LED is blinking. The LCD displays a menu. By clicking on SW2 the speed can be set using the potentiometer, it will take effect after clicking back SW2. The SW3 is for allowing Brightness setup, it will take effect after clicking back SW3.



LAB3

PART I Send message via UART

I.1 Introduction

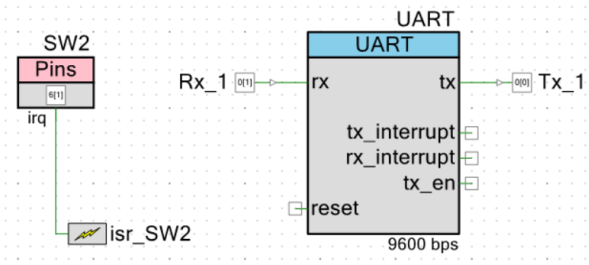
I.1.1 Overview

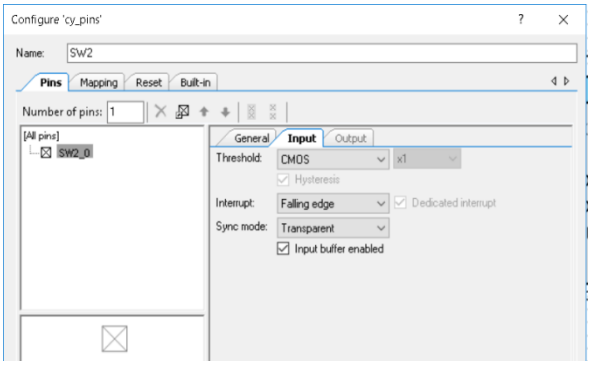
The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***Sending a message from PSoC to PC via UART***. The message must be sent by clicking to a button.

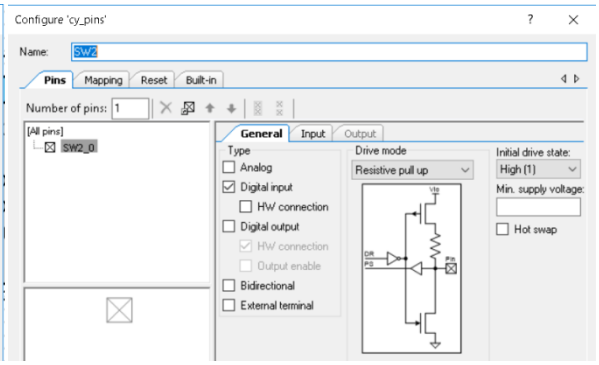
I.1.2 Required Instrumentation

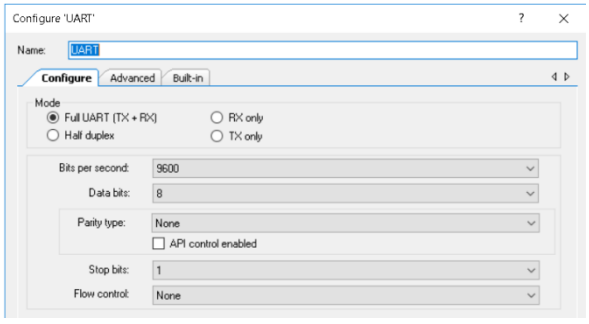
- CY8CKIT-050 is a PSoC 5LP development kit from CYPRESS based on CY8C5868-AXI-LP035 chip.
- UART-USB cable or RS232-USB cable
- 3 wire jumpers male-male.

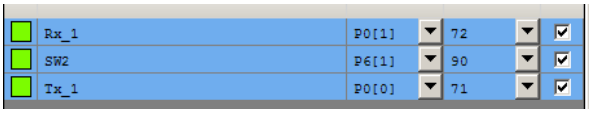
I.2 Hardware configuration











I.3 Software configuration

```
#include<project.h>

CY_ISR(SW2_Handler)
{
    SW2_ClearInterrupt();
    UART_PutString("Azul !");
}

int main()
{
    CyGlobalIntEnable;

    isr_SW2_StartEx( SW2_Handler );

    UART_Start();

    for(;;)
    {

    }
}
```

I.4 Hardware connections:

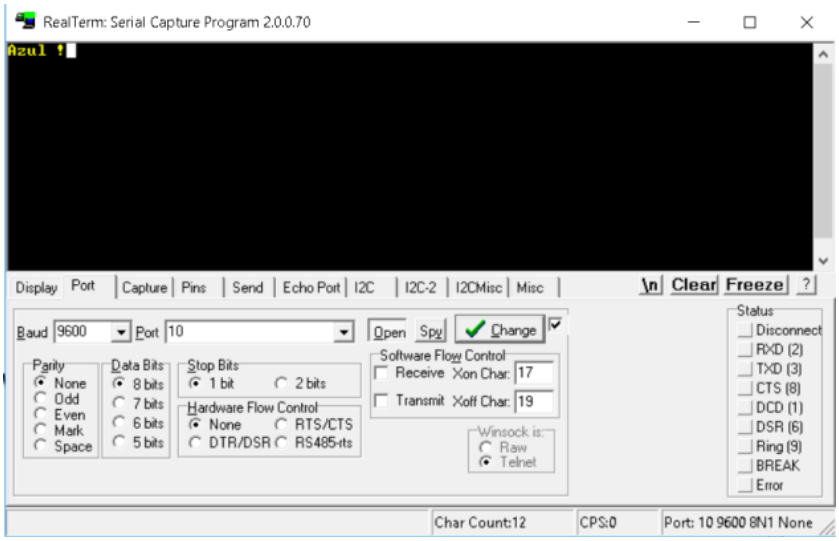
1. Connect P0_0 (Tx of PSOC) pin to Rx pin of the USB-UART cable adapter.
2. Connect VSSD pin of PSOC to GND pin of the USB-UART cable adapter.

I.5 Installation of a Serial Terminal

You will need to install RealTerm (a serial terminal) from the link https://sourceforge.net/projects/realterm/files/

I.6 Configuration of RealTerm and test

1. Go to Port
2. Set the Baud to 9600
3. Set and open the appropriate COM port



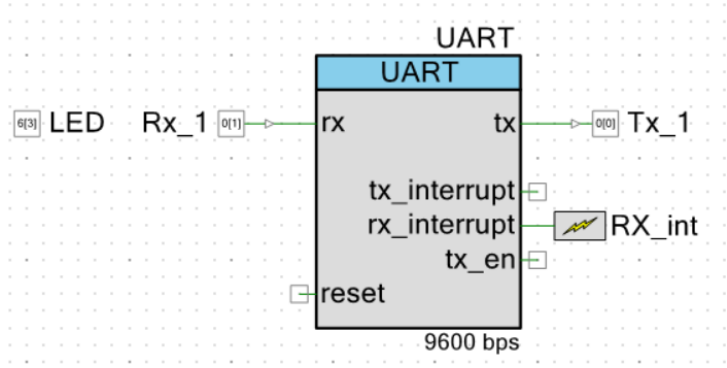
I.8 Exercise1

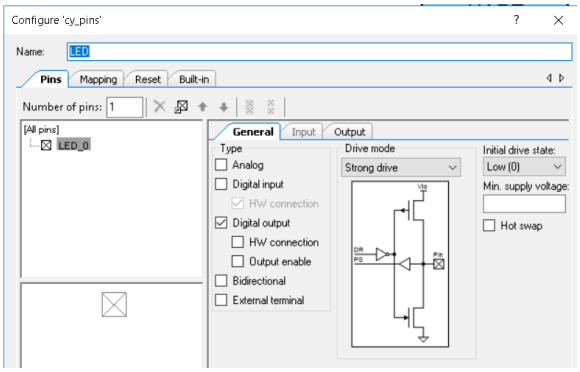
When a message is sent a LED must light in the board during 1s.

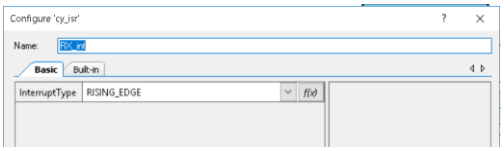
PART II Lighting LED via UART (Electronics of Things application example)

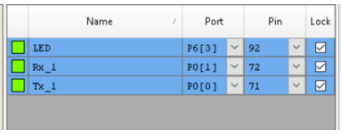
The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***Controlling a LED on PC***. The LED must light by sending '1' to the PSoC via UART and turn off by sending '0'. The PSoC must send back the state of the LED to the PC.

II.1 Hardware configuration







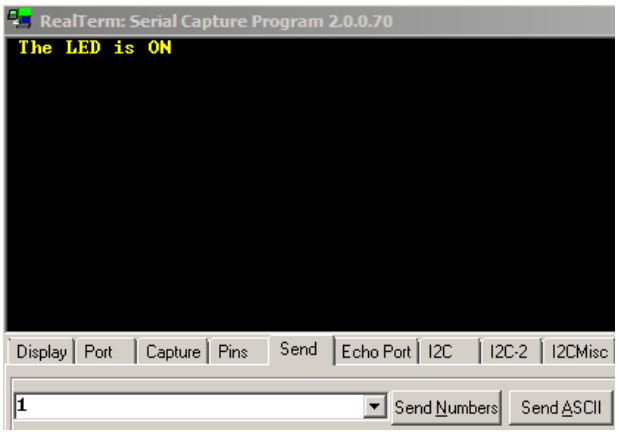


II.2 Software configuration

```
#include<project.h>

char data;

CY_ISR(isrRX)
    {
        data = UART_GetChar();
    }

int main()
{

    CyGlobalIntEnable;
    UART_Start();
    RX_int_StartEx(isrRX);

for(;;)
        {
            if (data == '1')
            {
                LED_Write(1);
                UART_PutString(" The LED is ON ");

                data = 0;
            }

            if (data == '0')
            {
                LED_Write(0);
                UART_PutString(" The LED is OFF ");
                data = 0;
            }
        }
}
```

II.3 Hardware connections and test

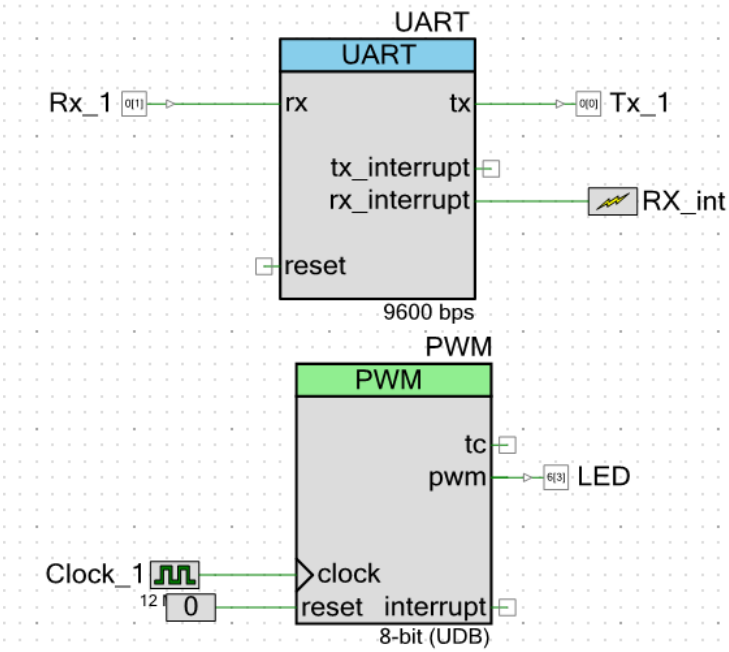
1. Connect P0_0 (Tx of PSOC) pin to Rx pin of the USB-UART cable adapter.
2. Connect P0_1 (Rx of PSOC) pin to Tx pin of the USB-UART cable adapter.
3. Connect VSSD pin of PSOC to GND pin of the USB-UART cable adapter.
4. On RealTerm go to send, write 1 and Send ASCII.

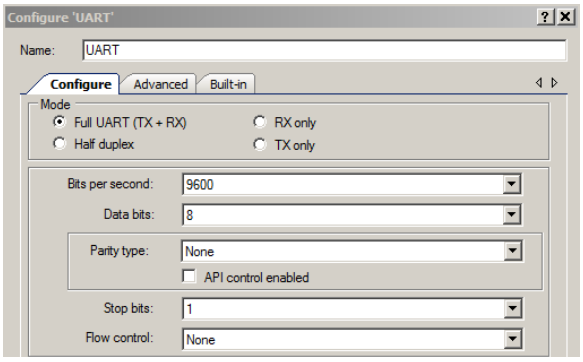


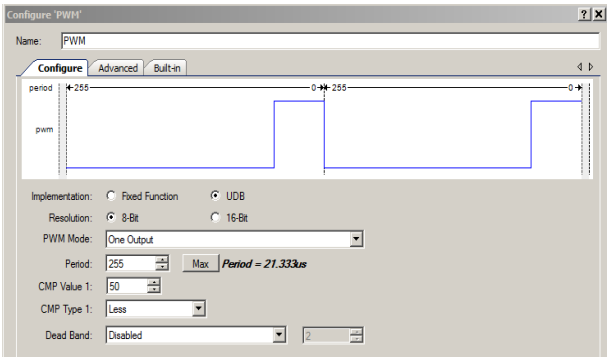
PARTIII Controlling LED blinking speed via UART

The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***Controlling the speed of LED blinking on PC***.

III.1 Hardware configuration







III.2 Software configuration

```
#include<project.h>

char speedval;

CY_ISR(isrRX)
    {
        speedval = UART_GetChar();
    }

int main()
{
    CyGlobalIntEnable;

    UART_Start();
    RX_int_StartEx(isrRX);

    UART_PutString(" Set the speed of the LED Blinking ");


    for(;;)

    {
        PWM_Start();
        PWM_WritePeriod(255);
        PWM_WriteCompare(200);
        CyDelay(speedval);

        PWM_Stop();
        CyDelay(speedval);

    }
}
```

III.3 Test

Set the desired speed of the LED blinking (from 0 to 255) and click Send Numbers

III.4 Exercise2

Set the blinking Speed and the Brightness of a LED by Electronics of Things

LAB4
Data Logger for Analog Sensor Data

PART I Built-in Potentiometer

I.1 Introduction
I.1.1 Overview

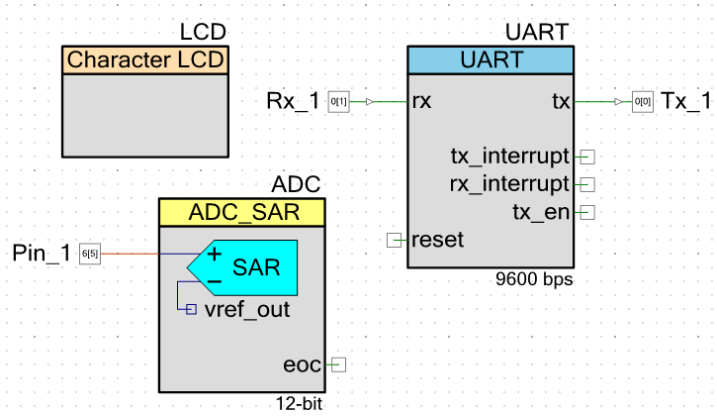
The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***displaying the built-in Potentiometer values***. The values must be displayed on a Terminal Emulator and LCD

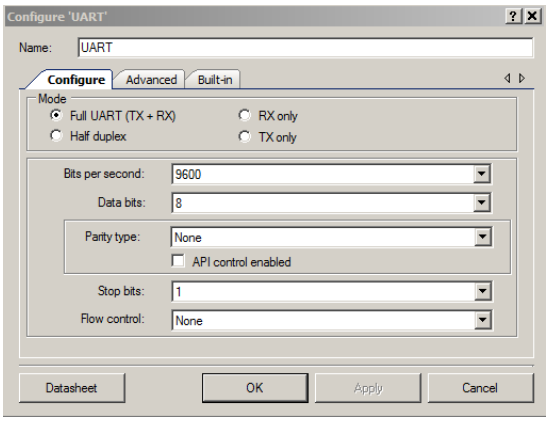
I.1.2 Required Instrumentation

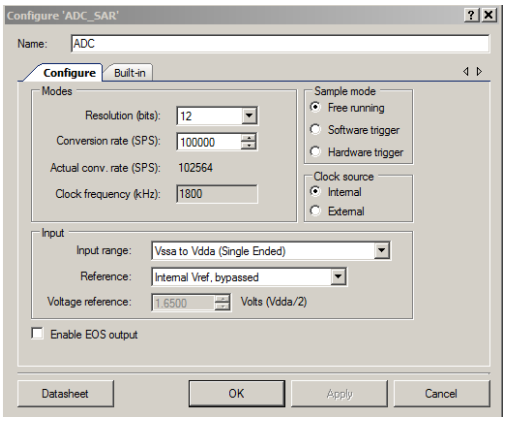
- CY8CKIT-050 is a PSoC 5LP development kit from CYPRESS based on CY8C5868-AXI-LP035 chip.
- 2x16 LCD Character Display.
- UART-USB cable or RS232-USB cable
- 2 wire jumpers male-male.

I.2 Hardware configuration
I.2.1 Components Configuration

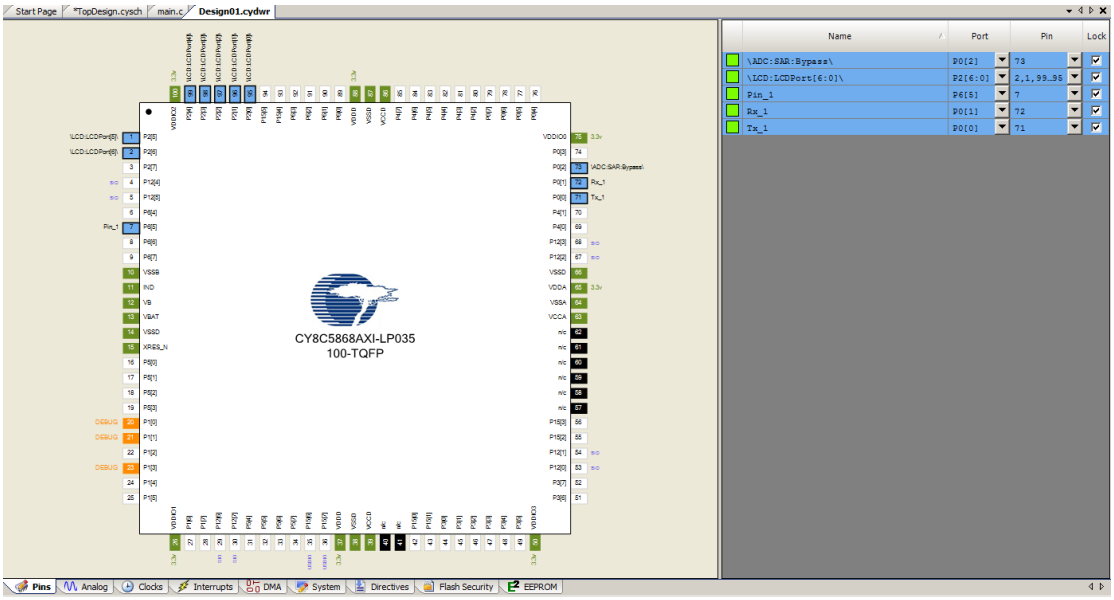
We need the following components: Analog Pin, SAR ADC, UART and Character LCD.





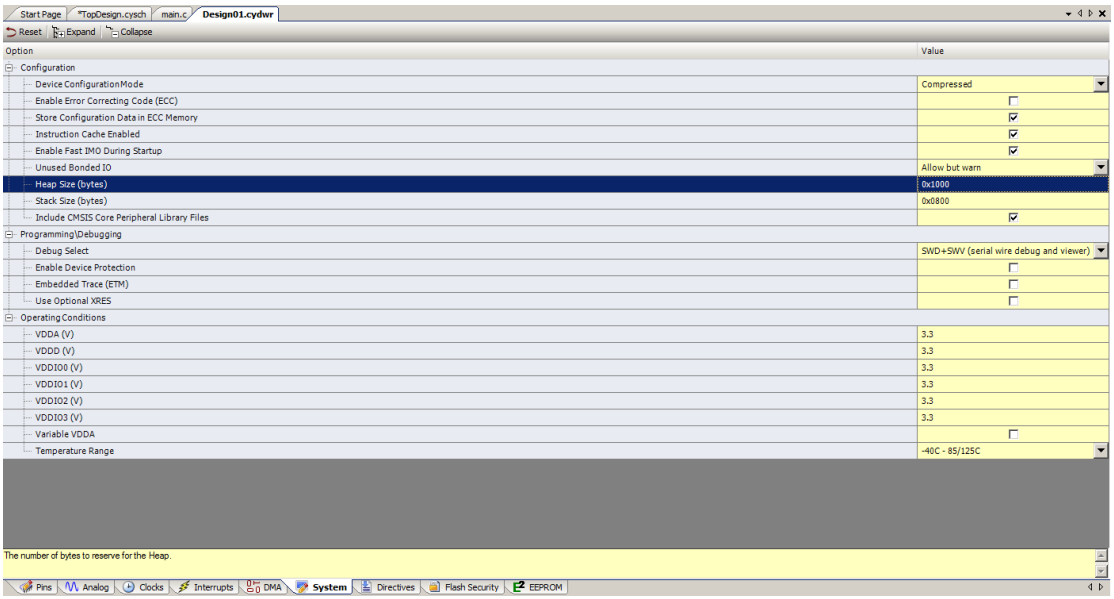


I.2.2 Pins Assignment



I.2.3 System Configuration

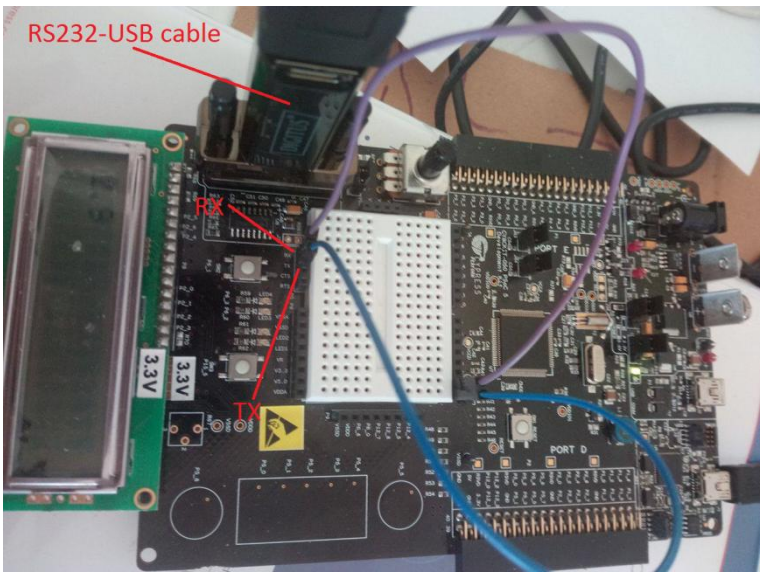
The power supply voltages should be set to 3.3 V
The Heap Size has to be set to 0x1000 bytes



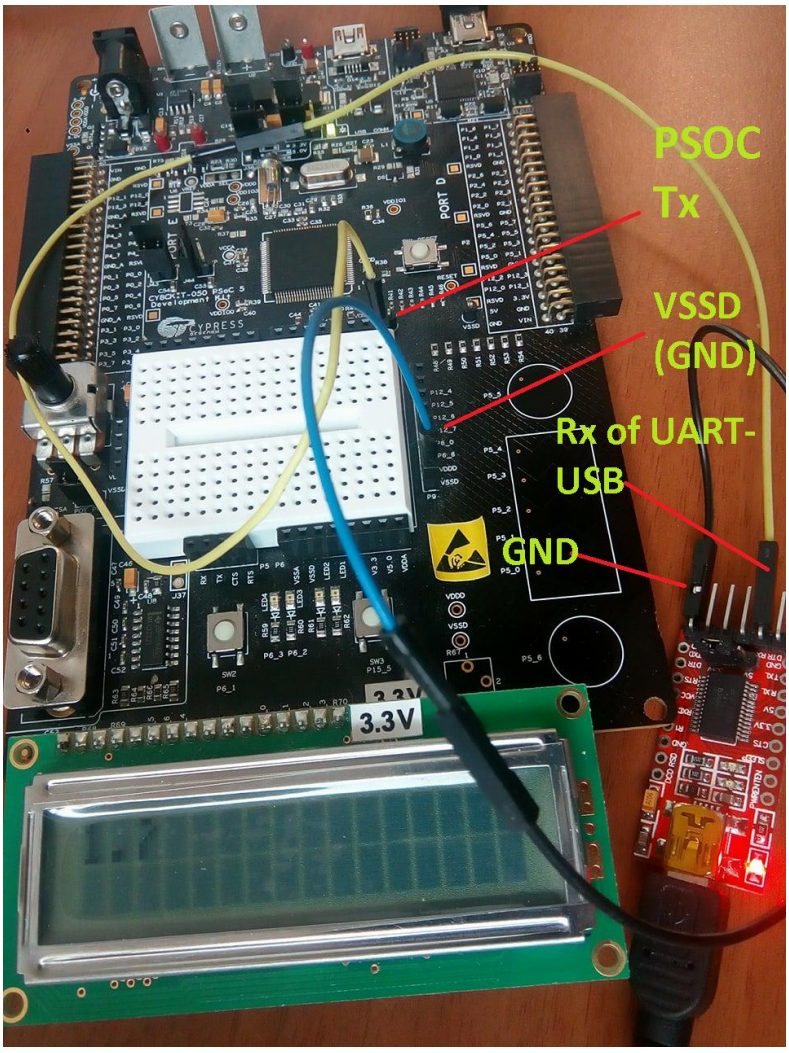
I.3 Software configuration

```c
#include <project.h>
#include <stdio.h>

#if defined (__GNUC__)
    asm (".global _printf_float");
#endif

int main()
{
    int16 result1 = 0;
    float res1 = 0;
    char8 resultStr1[16];
    LCD_Start();
    LCD_ClearDisplay();
    ADC_Start();
    ADC_StartConvert();
    UART_Start();

    for(;;)
    {
        ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
        result1 = ADC_GetResult16();

        res1 = ADC_CountsTo_Volts(result1);

        CyDelay(25);
        LCD_ClearDisplay();
        LCD_Position(0u,0u);

        sprintf((char *)resultStr1,"%1.1f",res1);

        LCD_PrintString(resultStr1);
        UART_PutChar(13);
        UART_PutChar(10);
        UART_PutString(resultStr1);
        CyDelay(100);
    }
}
```

I.4 Hardware connections
I.4.1 using RS232-USB cable

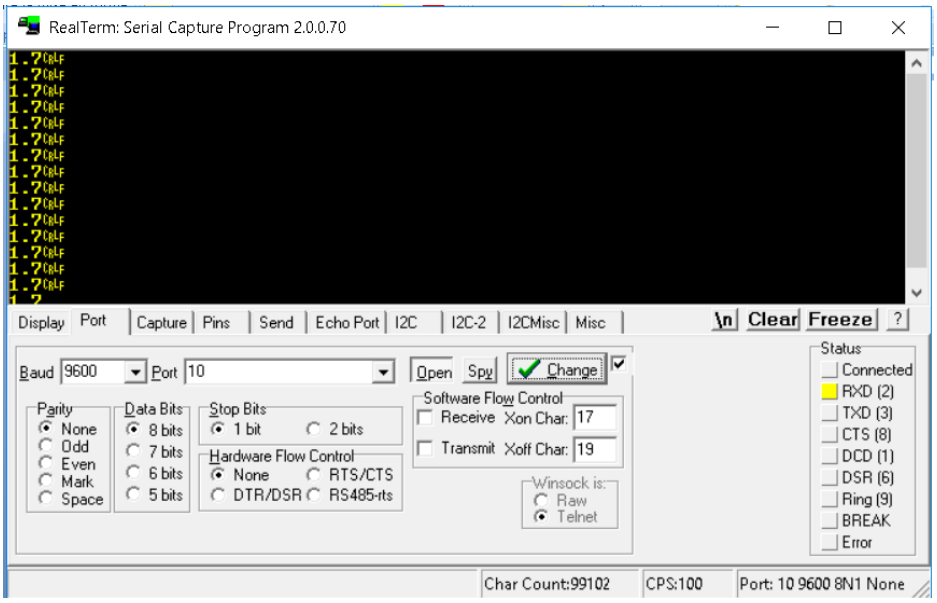


I.4.2 using UST-USB Adapter



I.5 Hardware Connexions and Results

- Before powering the board connect correctly the LCD.
- Connect the board from J1 to PC via USB, and the RS232-USB cable to PC.
- Identify which COM port has been assigned to the previously plugged cable, if the driver have not been previously installed you should wait until it is automatically done by Windows.
- Connect P0_0 (Tx of PSOC) pin to Rx pin of the USB-UART cable adapter.
- Connect VSSD pin of PSOC to GND pin of the USB-UART cable adapter.
- Launch any Serial Terminal software. Configure it as 9600 baud, 8 data bits, 1 stop bit and no parity. Then connect to previously identified COM port.
- By turning the potentiometer you can vary the voltage from 0 to 3.3V.
- The result is displayed in the Terminal and LCD at the same time.



PART II External Potentiometer (Goniometer)

II.1 Introduction

II.1.1 Overview

The purpose is to build an application using CY8CKIT-050 PSoC development kit for ***displaying and storing the built-in Potentiometer Positions***. The position must be displayed on a Terminal Emulator, LCD and the stored data must be plotted on MATLAB.

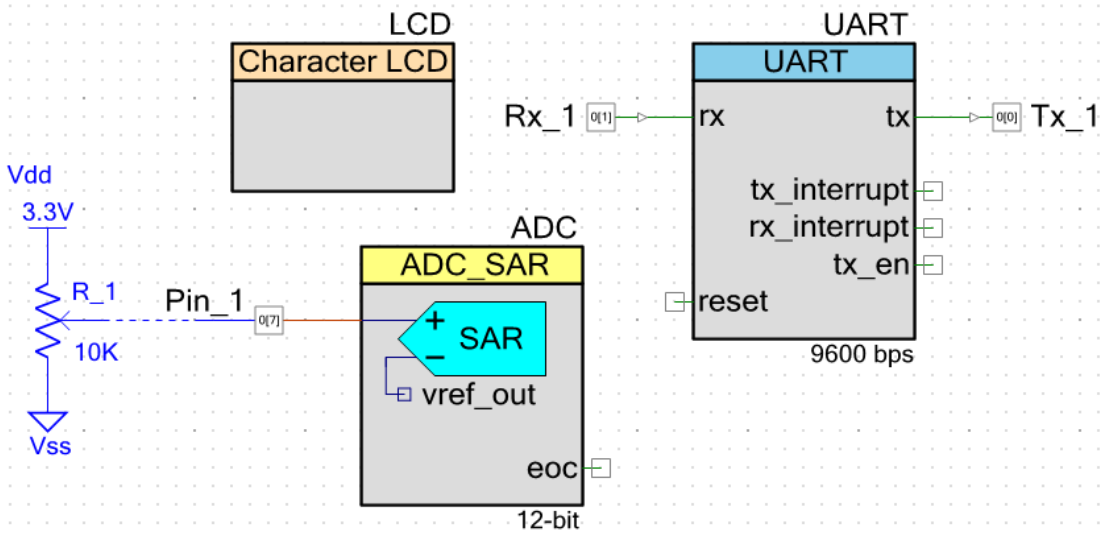
II.1.2 Required Instrumentation

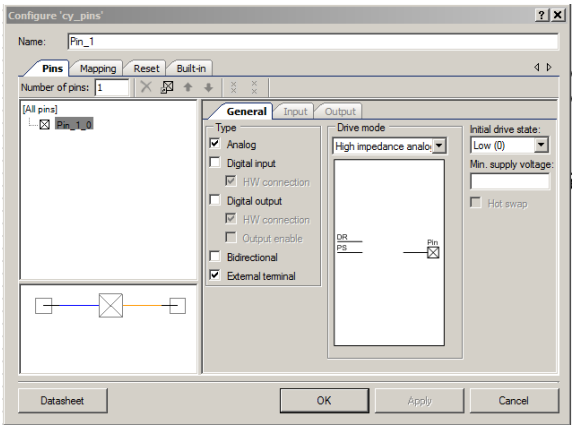
- CY8CKIT-050 is a PSoC 5LP development kit from CYPRESS based on CY8C5868-AXI-LP035 chip.
- 2x16 LCD Character Display.
- UART-USB cable or RS232-USB cable.
- 2 wire jumpers male-male.
- Trimmer Potentiometer.

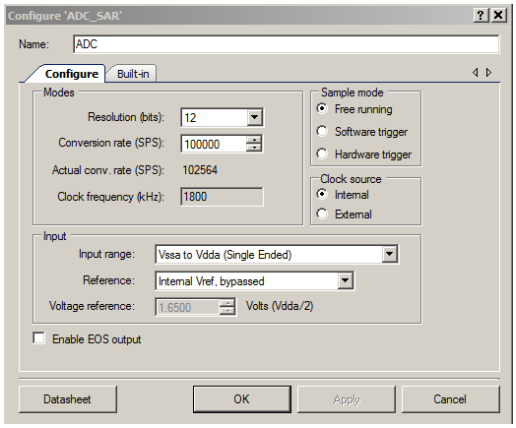
II.2 Hardware configuration

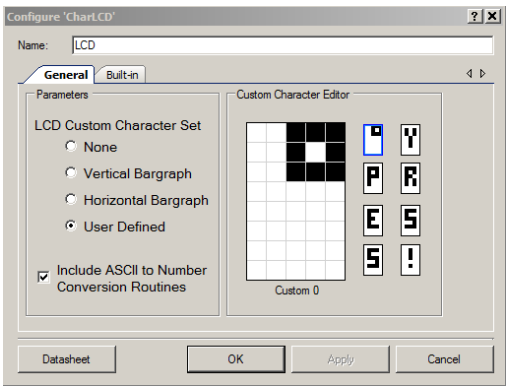
II.2.1 Components Configuration

We need these components: Analog Pin, SAR ADC, UART and Character LCD.
We need also Off-Chip components: Potentiometer, Ground, Power.

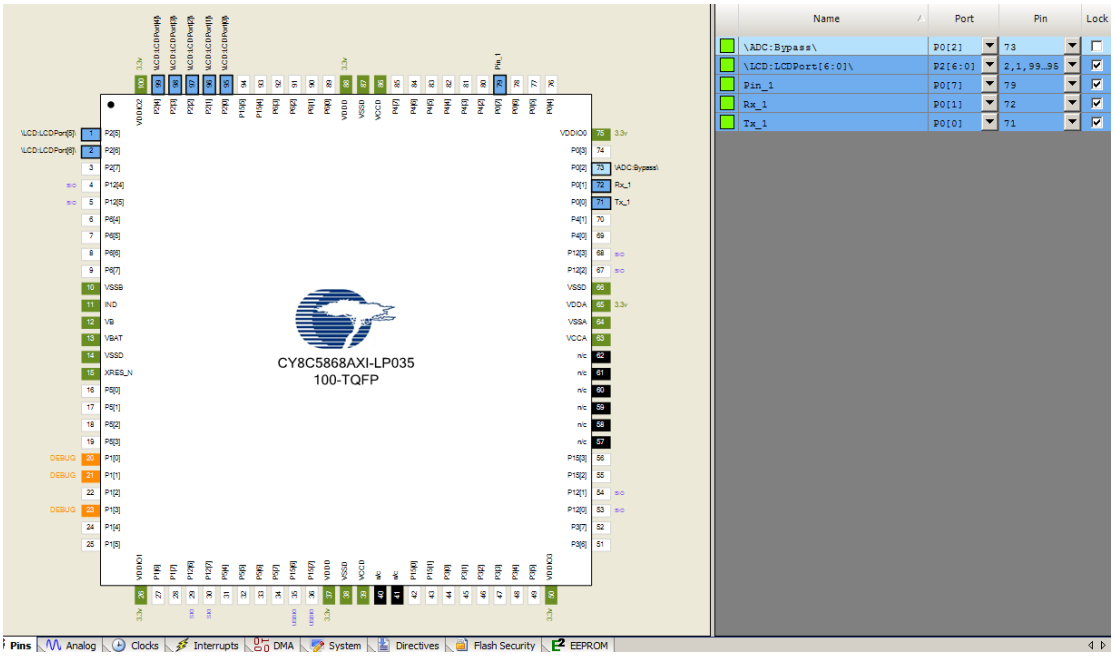

LCD
Character LCD
UART
UART
Rx_1
0[1]
rx
tx
0[0]
Tx_1
tx_interrupt
rx_interrupt
tx_en
reset
9600 bps
Vdd
3.3V
R_1
10K
Pin_1
0[7]
Vss
ADC
ADC_SAR
SAR
vref_out
eoc
12-bit


Configure 'cy_pins'
Name:
Pin_1
Pins
Mapping
Reset
Built-in
Number of pins:
General
Input
Output
Type
Analog
Digital input
Digital output
Bidirectional
External terminal
Drive mode
Initial drive state:
Min. supply voltage:
Hot swap
Datasheet
OK
Apply
Cancel


Configure 'ADC_SAR'
Name:
ADC
Configure
Built-in
Modes
Resolution (bits):
12
Conversion rate (SPS):
100000
Actual conv. rate (SPS):
102564
Clock frequency (kHz):
1800
Sample mode
Free running
Software trigger
Hardware trigger
Clock source
Internal
External
Input
Input range:
Vssa to Vdda (Single Ended)
Reference:
Internal Vref, bypassed
Voltage reference:
1.6500
Volts (Vdda/2)
Enable EOS output
Datasheet
OK
Apply
Cancel

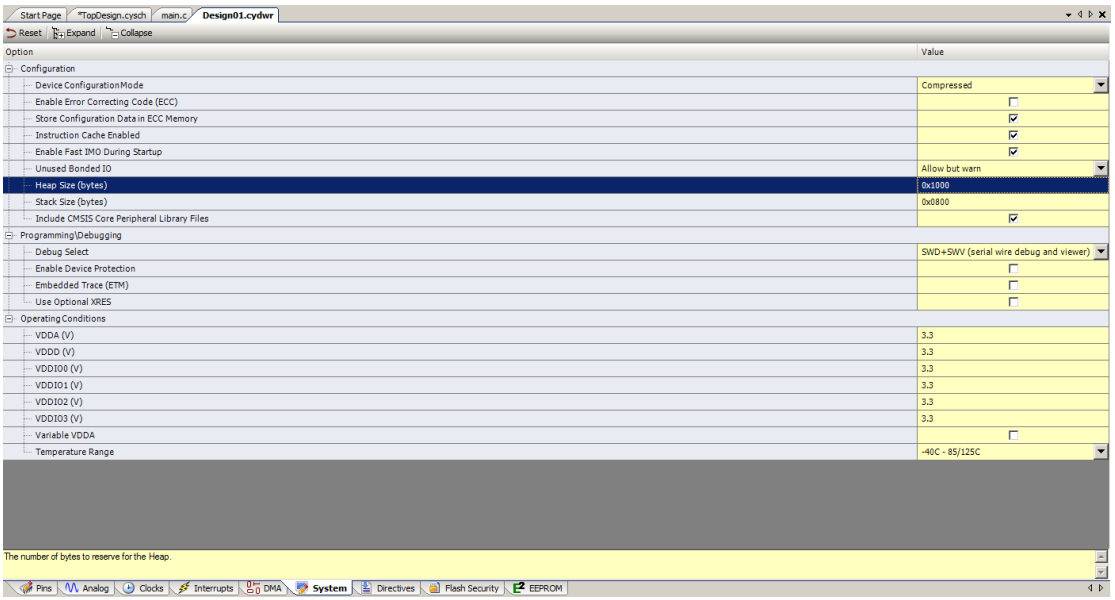

Configure 'CharLCD'
Name:
LCD
General
Built-in
Parameters
LCD Custom Character Set
None
Vertical Bargraph
Horizontal Bargraph
User Defined
Include ASCII to Number Conversion Routines
Custom Character Editor
Custom 0
Datasheet
OK
Apply
Cancel

II.2.2 Pins Assignment



II.2.3 System Configuration

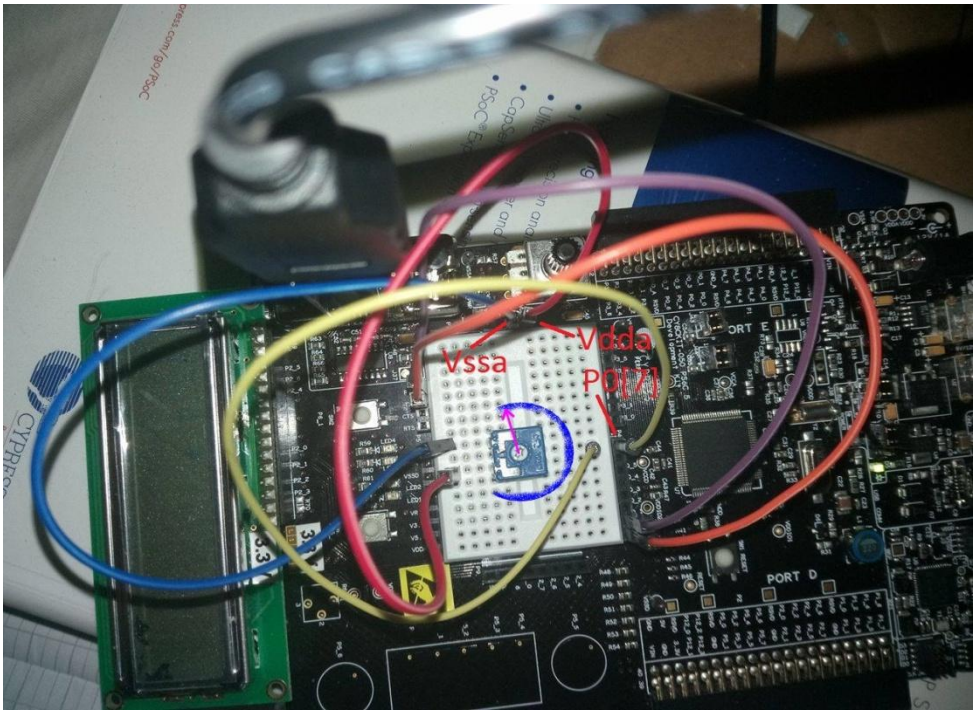
The power supply voltages should be set to 3.3 V
The Head Size has to be set to 0x1000 bytes



II.3 Software configuration

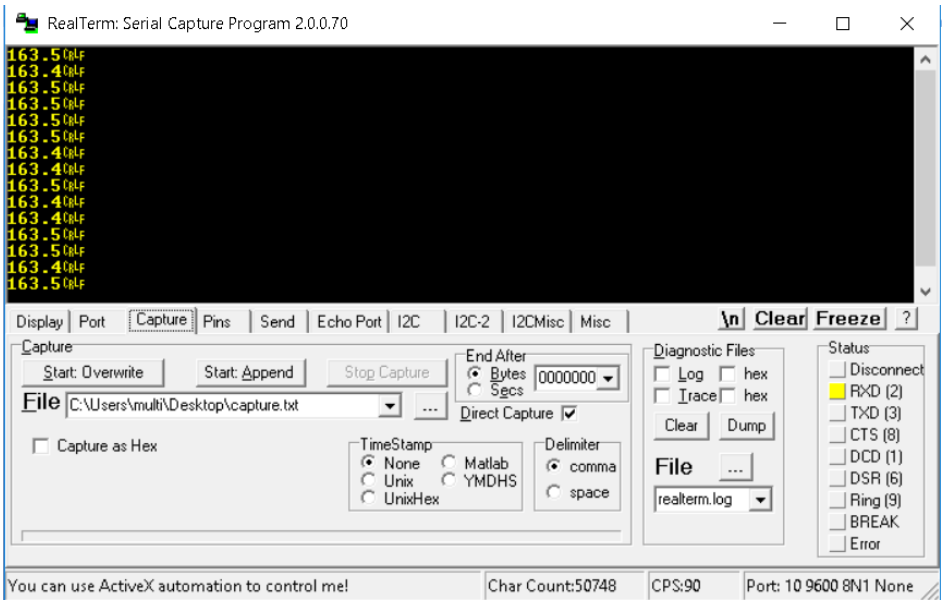
```
#include <project.h>
#include <stdio.h>
#if defined (__GNUC__)
    asm (".global _printf_float");
#endif

int main()
{
    int16 result1 = 0;
    float res1 = 0;
    char8 resultStr1[16];
    LCD_Start();
    LCD_LoadCustomFonts(LCD_customFonts);
    LCD_ClearDisplay();
    ADC_Start();
    ADC_StartConvert();
    UART_Start();

    for(;;)
    {
            ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
            result1 = ADC_GetResult16();
            res1 = ADC_CountsTo_Volts(result1);
            res1 = res1 * 220 / 3.3;
            CyDelay(25);
            LCD_ClearDisplay();
            LCD_Position(0u,0u);
            sprintf((char *)resultStr1,"%.1f",res1);
            LCD_PrintString(resultStr1);
            LCD_Position(0u, 5u);
            LCD_PutChar(LCD_CUSTOM_0);
            UART_PutString(resultStr1);
            UART_PutChar(13);
            UART_PutChar(10);
            CyDelay(100);
        }
}
```

II.4 Hardware connections



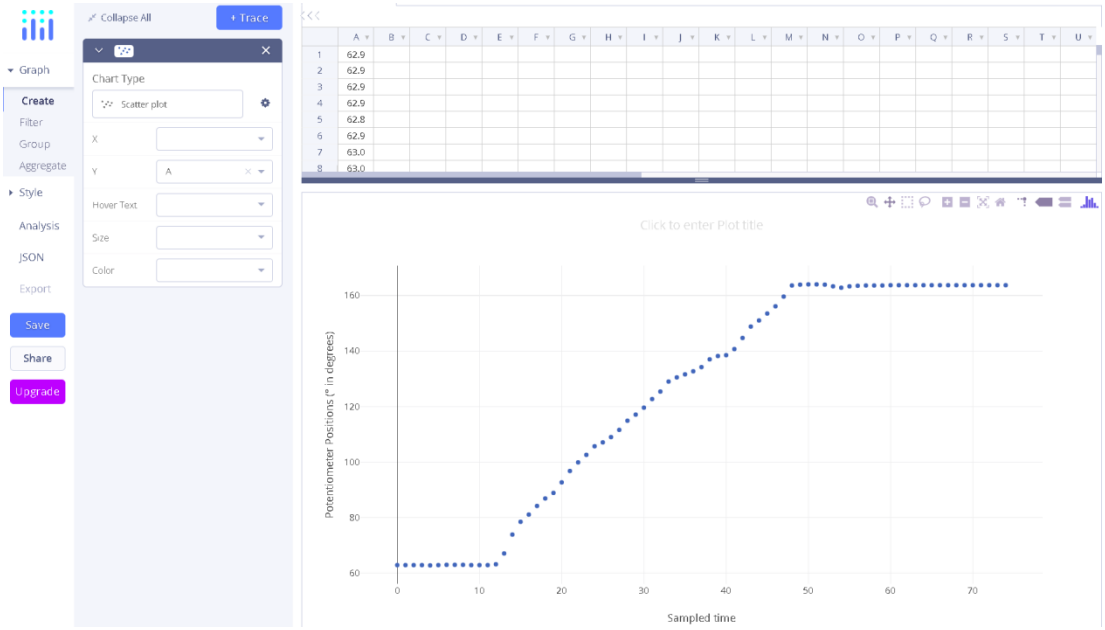
II.5 Results

- Before powering the board connect correctly the LCD.
- Connect the board from J1 to PC via USB, and the RS232-USB cable to PC.
- Identify which COM port has been assigned to the previously plugged cable, if the driver have not been previously installed you should wait until it is automatically done by Windows.
- Connect the two wire jumpers in order to link P0[0] to TX and P0[1] to RX.
- Launch Realterm software. Configure it as 9600 baud, 8 data bits, 1 stop bit and no parity. Open COM port. Go to Capture tab, Change file destination, Click on "Start".
- By turning the potentiometer you can vary the position from 0 to 220 °.
- The result is displayed in the Terminal and LCD at the same time.
- When you finish the measurement, click on "Stop Capture" then the log file is ready as txt in the destination folder.



II.6 Data Plot

- Use any online free graph maker (ex: https://plot.ly/create/)
- Open the previously created log file
- Copy the data and past in a column (or click import data on the interface).



III Exercise

Instead of external potentiometer, use internal one.

LAB5
State Machine

1 Introduction
1.1 Overview

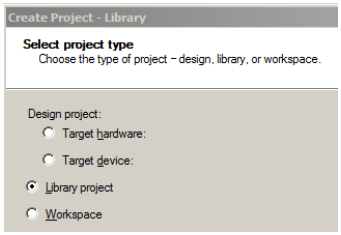
The purpose is to build an application using CY8CKIT-050 PSoC development kit for creating a ***manual LED chase using a state machine***. Two buttons have to be assigned for controlling the direction of the LED chase.

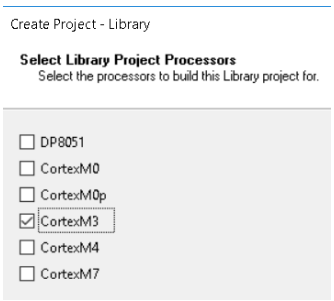
1.2 Required Instrumentation

- CY8CKIT-050 is a PSoC 5LP development kit from CYPRESS based on CY8C5868-AXI-LP035 chip.
- 4 LEDs.
- 4 wire jumpers male-male.

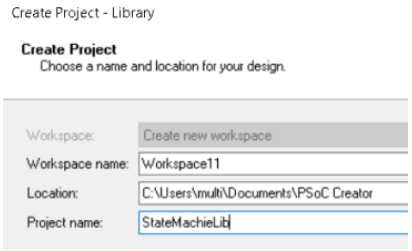
2 Library Creation

- Go to File-new-project-library project

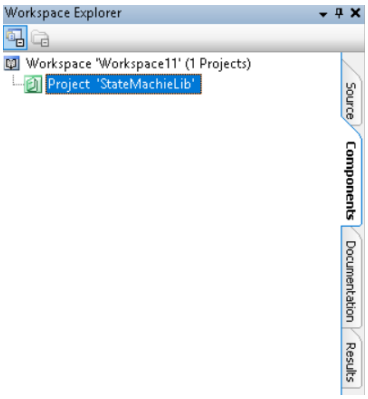




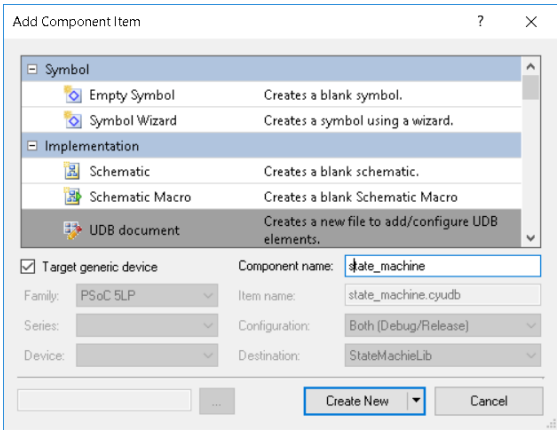
- Name the project: StateMachineLib



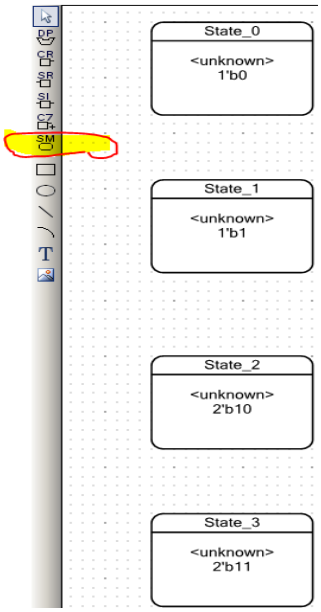
Go to Workspace explorer-Components tab



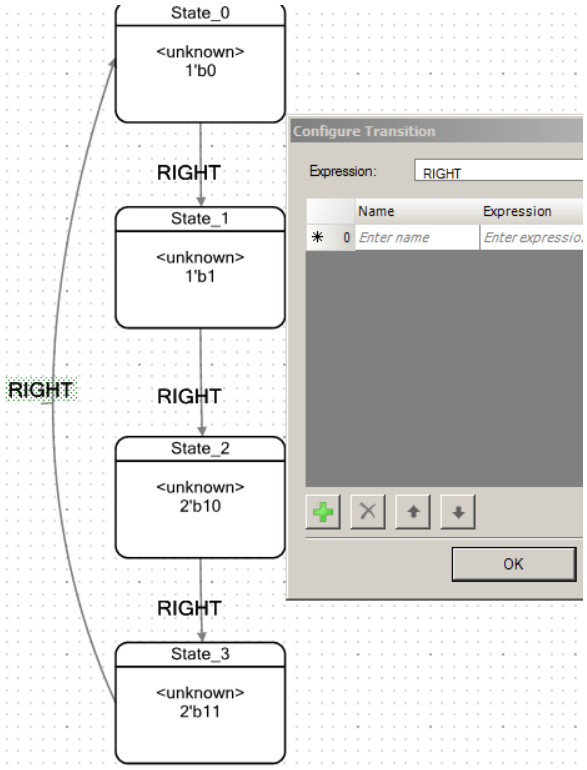
- Right click on Project “StateMachineLib”, add component item, UDB document



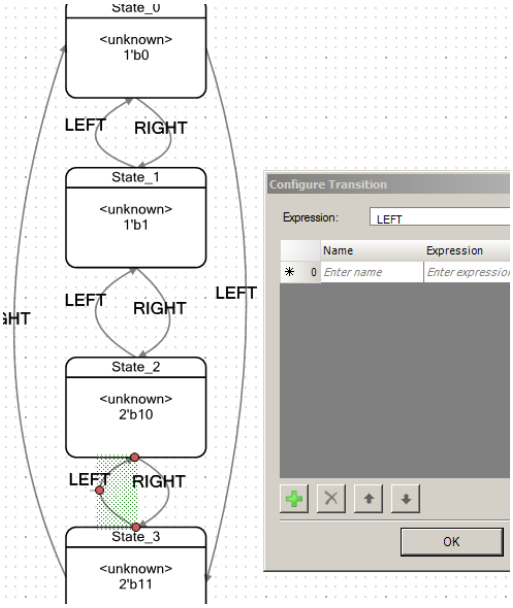
- Drag and drop SM component (4 times).



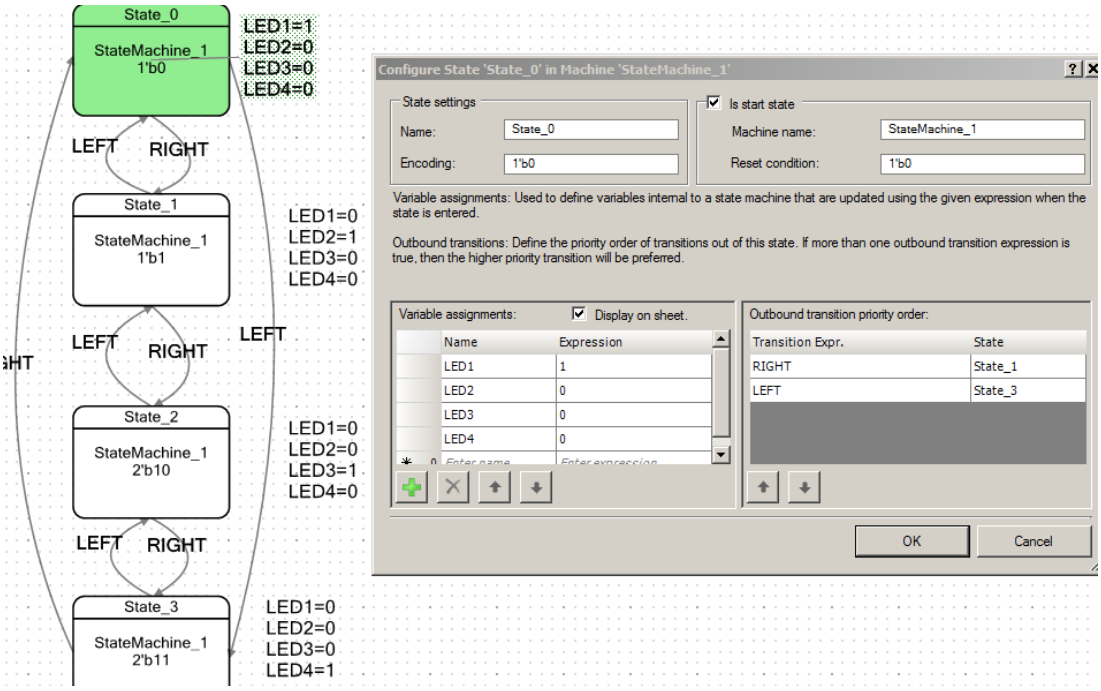
- Wire a connection from the bottom of each component to the top of the next one.



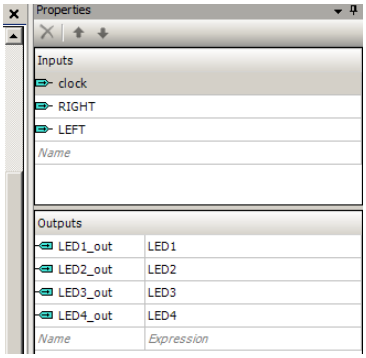
- Wire a connection from the top of each component to the top of the next one.



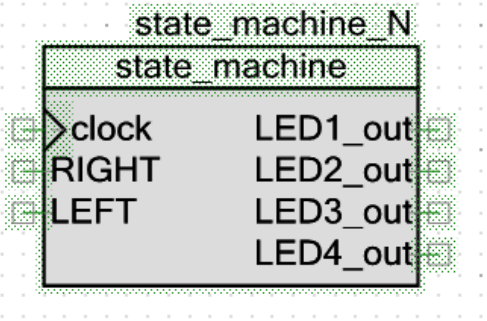
- Double click on the components for variable assignments and to set the first component as start state.



- Set a Name to each the input. Set a Name and Expression to each output.



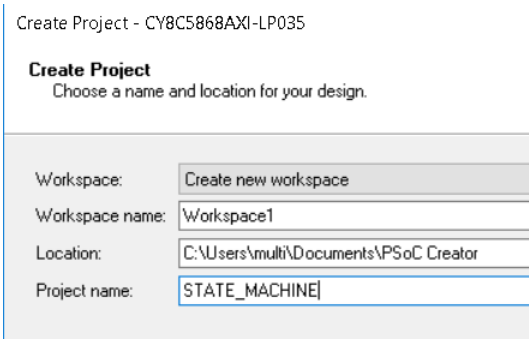
- Right-click on the blank side of the schema then select Generate Symbol.



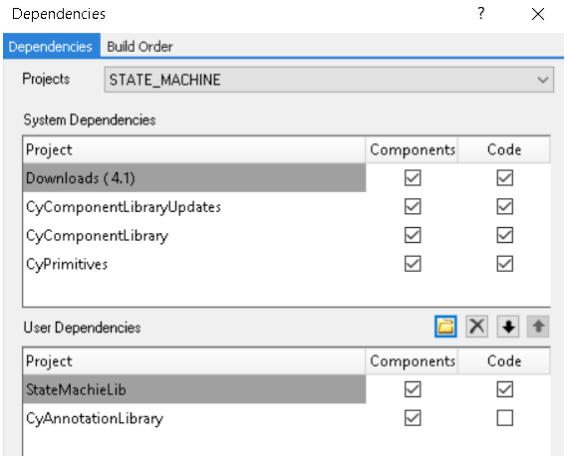
3 Hardware Configuration

3.1 Project Creation

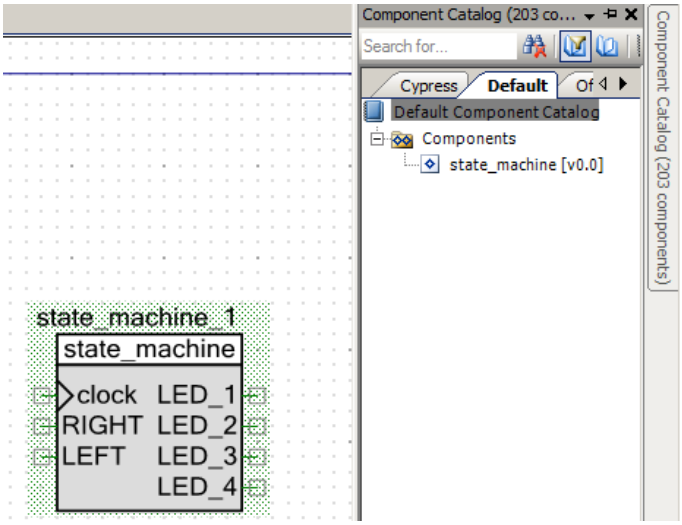
- Close the current PSOC Creator project window.
- Open a new PSOC Creator window then create a new project.



- On Workspace Explorer under source tab right click on project "STATE_MACHINE" choose Dependencies then click on New Entry.
- Localize your library previously created on a .cylib folder then add it.

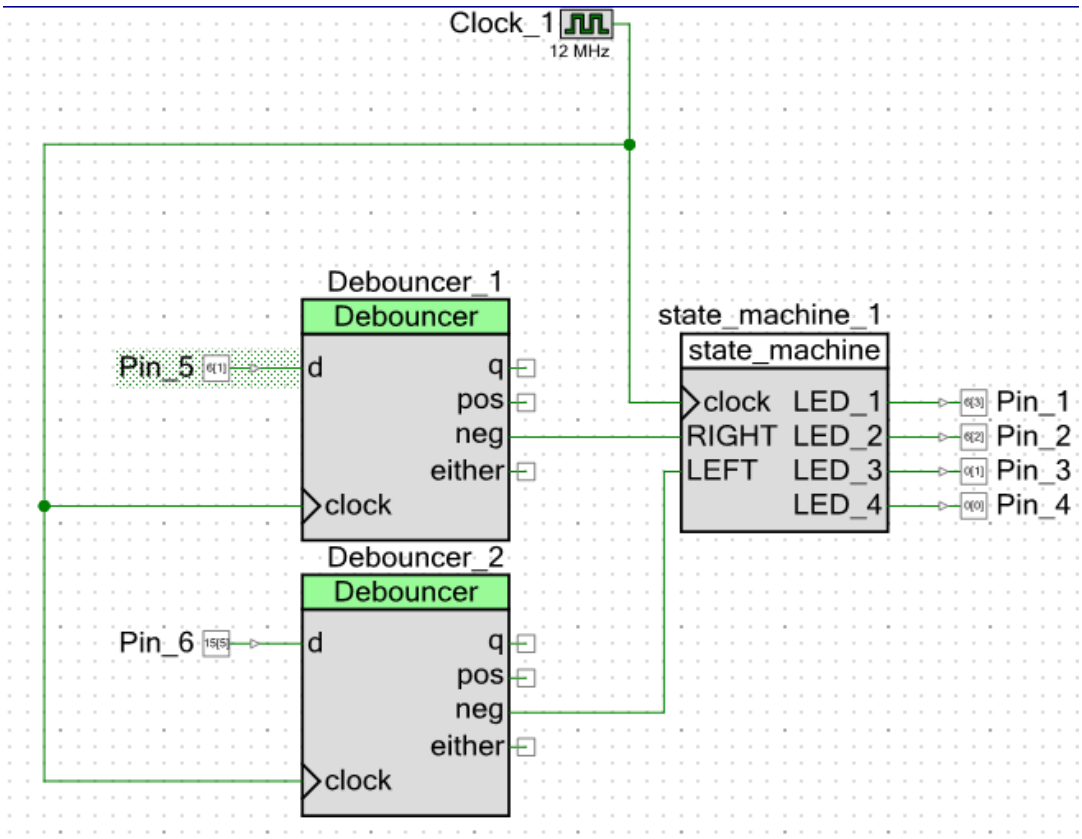


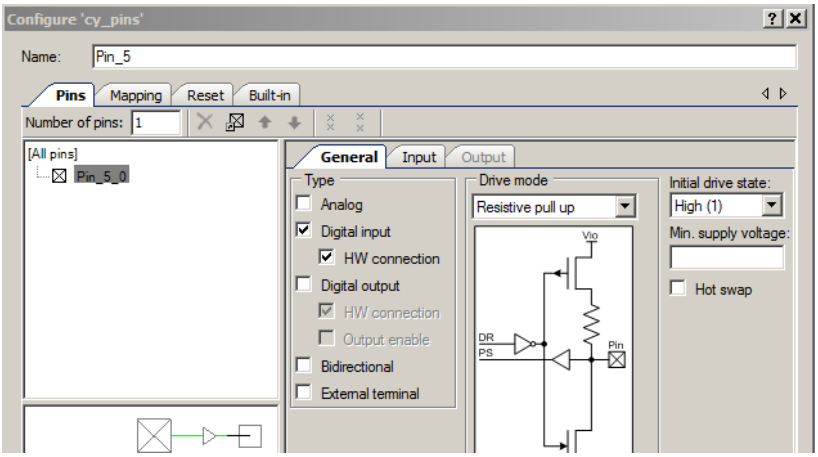
- Go to Component Catalog, under Default tab drag and drop to your schematic the **state_machine** component already created.

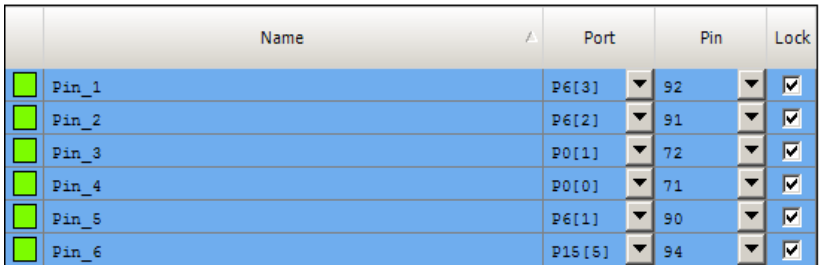


3.2 Components Configuration

- We need the following components: state_machine, Clock, 2 Debouncers, 2 Digital Input Pins and 4 Digital Output Pins.







	Name	Port	Pin	Lock
	Pin_1	P6[3]	92	☑
	Pin_2	P6[2]	91	☑
	Pin_3	P0[1]	72	☑
	Pin_4	P0[0]	71	☑
	Pin_5	P6[1]	90	☑
	Pin_6	P15[5]	94	☑

4 Exercise

Instead of built-in LEDs use externals connected on breadboard.

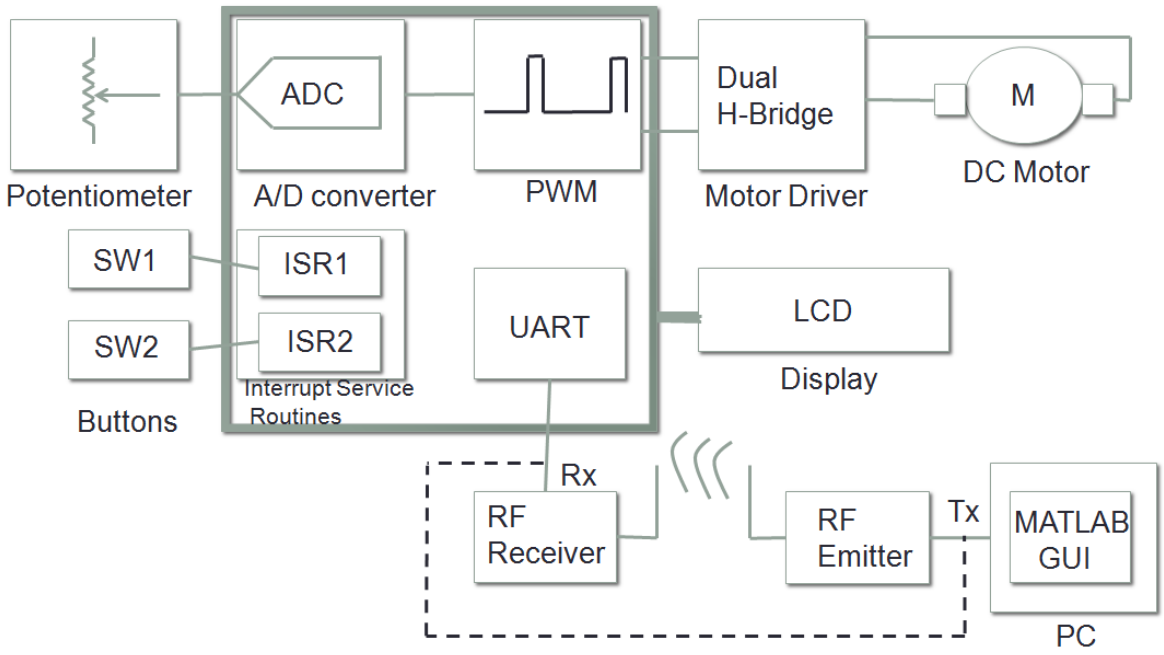
Mini-Project
Remotely Controled DC Motor (PSOC5 Design)

1 Overview

The mini-project consists of designing and implementing a remotely controled DC motor using CY8CKIT-050 PSoC development kit and some discret components. An implemented GUI (Graphical User Interface) on PC is used to for sending commands via RF modules to PSOC in order to control the speed and the direction of the DC motor. Two buttons and a potentiometer have the same purpose. The LCD is used as HMI (Human-Machine Interface) to display the purpose of each button and the speed of the motor (in RPM).

2 Required Instrumentation

- CY8CKIT-050 is a PSoC 5LP development kit from CYPRESS based on CY8C5868-AXI-LP035 chip.
- DC motor
- Motor Driver
- 2x16 LCD Character Display.
- RF Emitter and RF Receiver modules
- USB-UART adapter
- Wires



Remotely Controled DC Motor (PSOC5 Design)