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

This LABBOOK is available in print and as an electronic book (PDF format).

Text and diagrams from this book may be reproduced in their entirety and used for non-profit academic purposes, provided that a clear reference to the original source is made in all derivative documents. This reference should be of the following form:

Ahmed Bouzid, The PSoC 5LP LABBOOK, First Edition, University of Miskolc, 2018.

Requests to use content from this book for other than non-profit academic purposes should be made to qgebouzid@uni-miskolc.hu

This book may not be reproduced in its original form and sold by any unauthorized third party.

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 underdimension ned 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

ACKN PREFA TABL	OWLEDGEM ACE E OF CONTE	IENTS NTS	ii iii iv
LAB1			1
	PART I	Blinking LED	1
	PART II	Blinking LED with PWM component	3
	PART III	Controlling LED (Hardware Design)	5
	PART IV	Controlling LED (Hardware/Software Design)	6
	PART V	Controlling a LED with an interrupt	9
LAB2			11
	PART I	Maximal value indicator (with comparator)	11
	PART II	Maximal value indicator (with ADC)	13
	PART III	Changing Speed and Brightness of Blinking LED	16
LAB3			21
	PART I	Send message via UART	21
	PART II	Lighting LED via UART	23
	PARTIII	Controlling LED blinking speed via UART	25
LAB4		Data Logger for Analog Sensor Data	27
	PART I	Built-in Potentiometer	27
	PART II	External Potentiometer (Goniometer)	31
LAB5		State Machine	37
Mini-F	Project	Remotely Controled DC Motor (PSOC5 Design)	44

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.

Create Project - CY8C58 Select project type Choose the type of p	68AXI-LP035 roject - design, library, or workspace.
Design project:	
○ Target module:	\frown
Target <u>d</u> evice:	PSoC 5LP V Last used CY8C58684XI-LP035
 Library project 	
○ Workspace	

- Choose Empty Schematic and click next



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

Create Project - CY8C5868AXI-LP035
Create Project
Choose a name and location for your design.
Workspace:
Create new workspace

Holkspace.	create new workspace
Workspace name:	Workspace01
Location:	C:\Users\xxxxx\Documents\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 $\pm 5\%$ tolerance

Configure 'cy_clo	Configure 'cy_clock'						
Name: Clock							
Basic A	dvanced Built-in						
Clock type:	New O Existing						
Source:	<auto></auto>						
Specify:	Frequency: 10 Hz 🗸						
	✓ Tolerance: - 5% + 5%						

- 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]

Workspace Explorer (1 project)	≁ 4 ×	Start Page TopDesign.cysch Blinking_LED.cydwr					•	4 Þ 🗙
📲 🕞 🔯 Workspace "Workspace05" (1 Projects)	~	1		Name	Port	5	Pin	Lock
Project 'Blinking_LED' [CY8C5868AXI-LP035] Sopplesign.cysch	Son	• 1000	<u> </u>	LED	P6[3] ~	92	~	
Pesign Wide Resources (Blinking_LED.cydwr)	8	2 PAD 3 PAD 3 PAD	10068 5. 53- 1983 14. 1983 13.					

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.

- 4 FX Competer Calls PWM Copress Contraction Contrac	
PWM 1	g (1 of 202 comp
Cuprect Bentasco PWM 1	🦓 🔟 🛈
PWM 1	
PWM_1	o o nont Catalo d
PWM 1	ponenceatalog
PWM_1	
PWM_1	ions
	VM [v3.30]
PWM L PWM	
· · · · · · · · · · · · · · · · · · ·	
pwm1	
pwini	
nwm2	
prinz	
reset interrupt	
S-bit (LIDB)	
0-0ii (000)	

- 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 $\pm 5\%$ tolerance

Co	Configure 'cy_clock'							
N	lame: Clock	1						
	Basic Ad	vanced Built-in	1					
	Clock type:	New	 Existing 					
	Source:	<auto></auto>						
	Specify:	Frequency:	100 Hz ~					
		☑ Tolerance: •	5% + 5%					

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

Configure 'PWM'		?	×
Name: PWM			
Configure	dvanced Built-In		4 ⊳
period ¥-99			-0-#
Implementation:	Fixed Function IDB		
Resolution:	● 8-Bit ○ 16-Bit		
PWM Mode:	One Output V		
Period:	99 Aax Period = 1s		
CMP Value 1:	5 🔹		
CMP Type 1:	Less ~		
Dead Band:	Disabled v 2		

- 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]

Workspace Explorer (1 project)	- # X / 3	Start Page *TopDesign.cysch	*BlinkingPWM.cydwr				-	4 Þ 🗙
4 G		4	8		Name /	Port	Din	Lock
*Workspace 'Workspace04' (1 Projects)	^				ivarrie /	Port	PIN	LOCK
Project 'Blinking_LED_PWM' [CY8C5868AXI-LP035]	S)	• 8	*********		LED	P6[3] ~	92 🗸	
TopDesign.cysch	The second se	1 120		VEC 420 2 1.0-				
Pins	H	2 920		P801 34				
		1 100		Het 1				
	PWN	4						
		.						
PWN	v I							
	to							
	10	<u> </u>						
	pwm) .					
	P							
Clock 1 DD Clock								
Interest interest	terrupt	+						
0	IT (UDB))						
								

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

1 6[3] LED BUTTON [[1]

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.

Workspace Explorer (1 project) - 7 X	Start Page TopDesign.cysch Control_LED_HW.cydwr			•	4 Þ 🗙
Sec. 1997		Name	Port	Pin	Lock
Workspace 'Workspace05' (1 Projects)	4 9 2 4	i danie	1 on		LOCK
🖻 🔁 Project 'Control_LED_HW' [CY8C5868AXI-LP035] اير		BUTTON	P6[1] ~	90 🗸	
- TopDesign.cysch		LED	P6[3] ~	92 🗸	
Pins 0	2 Page Page Page Page Page Page Page Page				

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 ${\bf Strong}$ Drive mode, uncheck HW connection and the initial drive state must be set to ${\bf Low}$



- 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
1 #	include "project.h"
2	
3 i	int main(void)
4 🖂 {	
5	CyGlobalIntEnable;
6	
7	int a=0 ;
8	for(;;)
9 🖻	{
10	if (BUTTON_Read() == 0)
11 🗎	(
12	if (a == 0)
13	{
14	LED_Write(1);
15	a = 1;
16	
17 -	}
18	eise
19-	(
20	LED_@FICE(0);
22	a - 0,
23	3
24	,
25	}
26 3	,
,	

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
    1 #include "project.h"
    2
    3
      int main(void)
    4 🖂 {
           CyGlobalIntEnable;
    5
    6
    7
           int a=0 ;
           for(;;)
    9
            {
                if (BUTTON_Read() == 0)
   10
   11
                {
   12
                    if (a == 0)
   13
                    {
   14
                        LED_Write(1);
   15
                        CyDelay(200);
   16
                         a = 1;
   17
                    }
   18
                    else
   19
                    {
                        LED Write(0);
  20
                        CyDelay(200);
  21
                        a = 0;
  22
  23
                    )
  24
               ÷
  25
           )
  26 }
```

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**.

Configure 'cy_pins'		? ×	Configure 'cy pins' ? X
Name: DV2 Pins Mapping Reset Built Number of pins: 1 × 50 [4] pins] - 23 BW2_0	 in General Input Output Type Analog Digital input HW connection Oighal output HW connection Output enable Bidirectional External terminal 	d ▷ Initial drive state: High (1) ∨ Min. supply voltage: Hot swap	Name: SW2 Pins Mapping Reset Bulk-In Number of pins: 1 X General Imput Output Threshold: CMOS Witheredis Input Input bulfer enabled

- 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 $\pm 5\%$ 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:



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.

Confiç	gure 'Comp'	
Name	e: Comp	
	Configure Built-in	
- 1	Hysteresis	
	O Enable	Disable
- 9	Speed O Ultra Low Power O Slow	Fast
- 1	PowerDownOverride O Enable	Disable
1	Polarity Non-Inverting	O Inverting
-	Svnc	
	O Normal	Bypass
	Vdda V. V4 OV	
	Vddd out Vssd	

- 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.

onfigure	'ADC_SAR'		
Name:	ADC		
Con	figure Built-in		
- Mod (Ac	es Resolution (bits): Conversion rate (SPS): ctual conv. rate (SPS): Clock frequency (kHz):	8 100000 ÷ <i>UNKNOWN RATE</i> 1200	Sample mode Free running Software trigger Hardware trigger Clock source Internal External
Inpu	t		
	Input range: Vss	a to Vdda (Single Ended)	\sim
	Reference: Inte	rnal Vref	\sim
Vo	ltage reference: 2.50	00 🗘 Volts (Vdda	/2)
🗆 Er	able EOS output		
<u> </u>			
Pota	r 655	ADC ADC_SAR	LC Character LC
		u	
		eoc	EI3 LED
		8-bit	

- 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.

Workspace Explorer 🗸 🕈 🗙	Start Page *TopDesign.cysch *ADC_LED.cydwr main.c					- 4 Þ ×
	****		Name /	Port	Pin	Lock
"Workspace WorkspaceU/ (1Projects) "		t	\LCD:LCDPort[6:0]\	P2[6:0]	2,1,99_95	~ 🗹
TopDesign.cysch Design Wide Resources (ADC_LED.cydwr)	• 8 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	* • • • • • • • • • • • • • • • • • • •	LED	P6[3]	92	~ 🗹
Pins Q		voors 🔤 🕬	Potar	P6[5]	7	< <p></p>
- Clocks	2 500 100 100 100 100 100 100 100 100 100	10 10 10 10 10 10 10 10 10 10 10 10 10 1				
Pia DMA	3 Pigs	Prop. 10				
- System	Pair 🚺 Mgt	P(1 8				

II.3 Software configuration

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



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.

onfigure 'cy_pins' lame: Potar			? >
Pins Mapping Reset Buil	t-in		4
[All pins]	General Input	Output Drive mode	Initial drive state:
	Analog Digital input HW connection Digital output	High impedance analo V	Low (0) V Min. supply voltag
	HW connection Uutput enable Bidirectional	<u>Pa</u> Pin	_ Hot swap
	External terminal		

- 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.

Configure 'ADC_SAR'

Name: ADC Configure Built-in		
Modes Resolution (b Conversion rate (SI Actual conv. rate (SI Clock, frequency (ki	aks: 8 ✓ PS: 100000 ♀ PS: UNKNOWN RATE Hz: 1200	Sample mode Free running Software trigger Hardware trigger Clock source Internal External
Input Input range: Reference: Voltage reference:	Vssa to Vdda (Single Ended) Internal Vref 2.5000	~

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

Configure 'PWM'		?	×
Name: PWM		_	_
Configure	Advanced Built-in		4 ⊳
period ¥-255 — pwm			-0 -#
Implementation:	○ Fixed Function		
Resolution:	8-Bit 0 16-Bit		
PWM Mode:	One Output 🗸		
Period:	255 A Max Period = 21.333us		
CMP Value 1:	50 🗢		
CMP Type 1:	Less V		
Dead Band:	Disabled V 2		

- 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:

```
#include<project.h>
 1
 2
 з
    unsigned char speedval = 250;
    unsigned char brightval = 250;
 4
 5
    CY_ISR(SpeedSW2_Handler)
 6
 7 🗖 (
 8
        SpeedSW2 ClearInterrupt();
 9
        CyDelay(200);
10
         while (SpeedSW2_Read() == 1)
11
12 🗄 {
13
14
         ADC_StartConvert();
         ADC IsEndConversion(ADC WAIT FOR RESULT);
15
16
         speedval = ADC_GetResult8();
17
18
         LCD_ClearDisplay();
19
        LCD Position(Ou, Ou);
20
        LCD_PrintString("Blinking Speed");
21
         LCD_Position(1u, Ou);
22
        LCD PrintNumber (speedval);
23
         CyDelay(10);
24
        }
25
   L}
26
    CY_ISR(BrightSW3_Handler)
27
28 🖂 (
29
         BrightSW3 ClearInterrupt();
30
        CyDelay(200);
31
32
         while (BrightSW3_Read() == 1)
33
         {
34
             ADC StartConvert();
             ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
35
             brightval = ADC_GetResult8();
36
37
38
             LCD ClearDisplay();
             LCD_Position(Ou, Ou);
39
40
             LCD_PrintString("Brightness");
41
             LCD Position(1u, Ou);
42
             LCD PrintNumber (brightval);
43
             CyDelay(10);
44
         }
45
   L3
46
    int main()
47 🖯 (
48
        CyGlobalIntEnable;
49
50
         Speed int StartEx(SpeedSW2 Handler);
51
         Bright_int_StartEx(BrightSW3_Handler);
52
53
         ADC Start();
54
         LCD_Start();
55
56
         for(;;)
57
         {
58
59
             PWM Start();
60
             PWM_WritePeriod(255);
61
             PWM_WriteCompare(brightval);
             LCD_Position(Ou, Ou);
62
63
             LCD_PrintString("SW2:Speed change");
             LCD_Position(1u, Ou);
64
65
             LCD PrintString("SW3:Brightness");
66
             CyDelay(speedval);
67
68
             PWM_Stop();
69
70
             CyDelay(speedval);
71
         }
72
   L }
73
```

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.



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



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

📲 RealTerm: Serial Capture Program 2.0.0.70		- 🗆	\times
Dista Pert Castrol Dis Lond Long Publics Lines Lines Lines Lines	20	lear Freezo	^
Baud 9600 Port 10 Image: Control for the point of the poi		Status Discor RXD (TXD (TXD (DCD (DSR (BREA Error	nnect 2) 3) 3) 1) 6) 9) K
Char Count 12	CPS-0 Por	+ 10 9600 9N1 N/	one /

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>
 1
 2
 3
    char data;
 4
 5
    CY_ISR(isrRX)
 6
7
        {
            data = UART_GetChar();
 8
        -}
 9
10 int main()
11 🗐 (
12
        CyGlobalIntEnable;
13
14
        UART Start();
        RX_int_StartEx(isrRX);
15
16
17
    for(;;)
18
             {
                 if (data == '1')
19
20
                 {
21
                     LED_Write(1);
22
                     UART_PutString(" The LED is ON ");
23
24
                     data = 0;
25
                 }
26
                 if (data == '0')
27
28
                 {
29
                     LED_Write(0);
30
                     UART PutString(" The LED is OFF ");
                     data = 0;
31
32
                 }
33
             )
34 - }
```

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.

📒 RealTerm: Serial Capture Prog	jram 2.0.0.70
The LED is ON	
Display Port Capture Pins 5	pena Echo Port 12C 12C-2 12CMisc
4	
l [⊥]	Send Numbers Send ASCI

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

	UART
	UART
Rx_1 ₪–	rx tx Tx_1
	tx_interrupt ⊟ rx_interrupt ────────────────────────────────────
	reset
	PWM
	PWM
	tc
	pwm
Clock_1	->clock
12 0	- <mark>reset interrupt</mark> - 8-bit (UDB)

Configure 'UART'	? X	Configure 'PWM'	? X
Name: UART	_	- Name: PWM	
		Configure Advanced Bult-In	4 ۵
Configure Advanced Built-in		0++2550++2550	-0-+
© Full LIART (TX + RX) C RX only			
O Half duplex O TX only		pwm	
Bits per second: 9600		Implementation: C Event Eurotion C UDB	
Data bits: 8		Bestivition:	
		PWM Mode: One Output	
Parity type: None		Period: 255 A Max Period = 21.333us	
API control enabled		CMP Value 1: 50 *	
Stop bits: 1		CMP Type 1: Less	
Bau sastel		Dead Band: Disabled	
Flow control: None			

LED	P6[3]	▼	92	•	
Rx_1	P0[1]	▼	72	•	
Tx_1	P0[0]	•	71	•	V

III.2 Software configuration



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.



Configure 'UART'	<u>? ×</u>	Configure 'ADC_SAR'	? X
Name: UART		Name: ADC	
Configure Advan	ced Built-in 4 b RX) C RX only C TX only	Configure Built-In Modes Resolution (bits): 12 Conversion rate (SPS): 100000 Conversion rate (SPS): 100000	4 Þ
Bits per second: Data bits: Parity type:	9600 V 8 V	Actual conv. rate (SPS): 102564 Clock frequency (kHz): 1800 Input Input Inpu	
Stop bits: Flow control:	API control enabled 1 Image: Control enabled None Image: Control enabled	Input range: Vsas to Vida (single Ended) Reference: Internal Wef, bypassed Voltage reference: 1.6500 Uotas (Vidda/2)	
Datasheet	OK Apply Cancel	Datasheet OK Apply C	ancel

I.2.2 Pins Assignment



I.2.3 System Configuration

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

Start Page *TopDesign.cysch main.c Design01.cydwr	× 4 Þ →
CReset 6+ Expand □ Collapse	
Option	Value
⊖- Configuration	
- Device Configuration Mode	Compressed
Enable Error Correcting Code (ECC)	
Store Configuration Data in ECC Memory	N
- Instruction Cache Enabled	N
Enable Fast IMO During Startup	N
Unused Bonded IO	Allow but warn
Heap Size (bytes)	0x1000
Stack Size (bytes)	0x0800
Include CMSIS Core Peripheral Library Files	V
⊖. Programming\Debugging	
- Debug Select	SWD+SWV (serial wire debug and viewer)
Enable Device Protection	
Embedded Trace (ETM)	
Use Optional XRES	
Operating Conditions	
··· VDDA (V)	3.3
VDDD (V)	3.3
VDDI00 (V)	3.3
VDDI01 (V)	3.3
VDD102 (V)	3.3
VDDI03 (V)	3.3
Variable VDDA	
Temperature Range	-40C - 85/125C
Variable VODA Temperature Range	
The number of bytes to reserve for the Heap.	

I.3 Software configuration

```
#include <project.h>
 1
 2
    #include <stdio.h>
 3
 4 #if defined (__GNUC__)
        asm (".global _printf_float");
 5
 6 #endif
 7
 8
    int main()
  ⊟ {
9
10
        int16 result1 = 0;
        float res1 = 0;
11
12
        char8 resultStr1[16];
        LCD_Start();
13
14
        LCD_ClearDisplay();
        ADC_Start();
15
16
        ADC StartConvert();
        UART_Start();
17
18
19
        for(;;)
20
        {
  Þ
                ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
21
22
                result1 = ADC_GetResult16();
23
24
                res1 = ADC_CountsTo_Volts(result1);
25
26
                CyDelay(25);
27
                LCD_ClearDisplay();
28
                LCD Position(Ou,Ou);
29
30
                sprintf((char *)resultStr1,"%1.1f",res1);
31
32
                LCD_PrintString(resultStr1);
                UART_PutChar(13);
33
34
                UART PutChar(10);
                UART_PutString(resultStr1);
35
36
                CyDelay(100);
37
            3
38 L }
```

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.

📲 RealTerm: Serial Capture Program 2.0.0.70	_	
1. 204F 1.	\n Clear	Freeze ?
Baud 9600 Port 10 Image: Change of the second secon		Status Connected RXD (2) TXD (3) CTS (8) DCD (1) DSR (6) Ring (9) BREAK Error
Char Count:99102 CPS	:100 Port: 10 9	600 8N1 None

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.











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

StartPage TopDesign.cysch main.c Design01.cydwr	× 4 ▷ ×
> Reset k+1 Expand '□ Collapse	
Option	Value
⊖ Configuration	
Device Configuration Mode	Compressed
Enable Error Correcting Code (ECC)	
Store Configuration Data in ECC Memory	N
Instruction Cache Enabled	N
Enable Fast IMO During Startup	N
Unused Bonded IO	Allow but warn
- Heap Size (bytes)	0x1000
Stack Size (bytes)	0x0800
Include CMSIS Core Peripheral Library Files	N
Programming\Debugging	
Debug Select	SWD+SWV (serial wire debug and viewer)
Enable Device Protection	
Embedded Trace (ETM)	
Use Optional XRES	
- Operating Conditions	
VDDA (V)	3.3
VDDD (V)	3.3
·· VDDI00 (V)	3.3
VDDI01(V)	3.3
VDDI02 (V)	3.3
··· VDDI03 (V)	3.3
Variable VDDA	
Temperature Range	-40C - 85/125C
W Pins //V, Analog Cocks 💉 Interrupts 🖧 UMA 🖓 System 🖆 Directives ڬ Flash Security 🚩 EEPROM	4 Þ

II.3 Software configuration

```
1 #include <project.h>
 2
   #include <stdio.h>
 3 # #if defined (__GNUC__)
        asm (".global _printf_float");
 4
 5 #endif
 6
    int main()
 7
8 🗔 {
 9
        int16 result1 = 0;
        float res1 = 0;
10
11
        char8 resultStr1[16];
        LCD_Start();
12
13
        LCD_LoadCustomFonts(LCD_customFonts);
        LCD_ClearDisplay();
14
        ADC_Start();
ADC_StartConvert();
15
16
        UART Start();
17
18
19
        for(;;)
20
        {
                ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
21
                result1 = ADC_GetResult16();
22
                res1 = ADC_CountsTo_Volts(result1);
23
                res1 = res1 * 220 / 3.3;
24
                CyDelay(25);
25
26
                LCD ClearDisplay();
27
                LCD_Position(Ou,Ou);
28
                sprintf((char *)resultStr1,"%.1f",res1);
29
                LCD_PrintString(resultStr1);
30
                LCD Position(Ou, 5u);
                LCD_PutChar(LCD_CUSTOM_0);
31
32
                UART_PutString(resultStr1);
                UART_PutChar(13);
33
34
                UART_PutChar(10);
35
                CyDelay(100);
36
37 L }
```

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.

늘 RealTerm: Serial Capture Program 2.0.0.70		_		×
163.5 GMF 163.5 GMF 163.5 GMF 163.5 GMF 163.5 GMF 163.4 GMF 163.4 GMF 163.4 GMF 163.4 GMF 163.4 GMF 163.5 GMF 163.5 GMF 163.5 GMF				<
Display Port Capture Disp Soud Eaks P	Port LICC 2 LICEMine L Mine L	Clear	Freeze	21
Display Port Capture Pins Send Echo F Capture Start: Append Start: Append Start File C:\Users\multi\Desktop\capture.txt Capture as Hex	Port 12C 12C-2 12CMisc Misc 12C op Capture Bytes 0000000 Direct Capture Cises Delimiter Cinx CiyMoHS Cispace File realterm.log	les hex hex hex ump	Status Disconr RXD (2) TXD (3) CTS (8) DCD (1) DSR (6) Ring (9) BREAK Error	?

II.6 Data Plot

- Use any online free graph maker (ex: <u>https://plot.ly/create/</u>)
- 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



DP8051
CortexM0
CortexM0p
CortexM3
CortexM4
CortexM7

- Name the project: StateMachineLib

Create Project - Library



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.

×	Properties	→ ₽
	× + +	
	Inputs	
	B≻ clock	
	■- RIGHT	
	B≻ LEFT	
	Name	
	Outputs	
	- LED1_out	LED1
	- LED2_out	LED2
	- LED3_out	LED3
	- LED4_out	LED4
	Name	Expression

- 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.

Create Project - CY8C5868AXI-LP035

Create Project Choose a name and location for your design.				
Workspace:	Create new workspace			
workspace.	Create new workspace			
Workspace name:	Workspace1			
Location:	C:\Users\multi\Documents\PSoC Creator			
Project name:	STATE_MACHINE			

- 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.

Dependencie	5		?	×
Dependencies	Build Order			
Projects	STATE_MACHINE			\sim
System Depe	ndencies			
Project		Components	Code	
Downloads	(4.1)		\checkmark	
CyCompon	entLibraryUpdates		\checkmark	
CyCompon	entLibrary	\checkmark	\checkmark	
CyPrimitive	5	\checkmark	\checkmark	
User Depend	encies		×	1
Project		Components	Code	
StateMachie	eLib		\checkmark	
CyAnnotati	onLibrary			

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

Component Catalog (203 co 🚽 🛱 🗙	8
Search for 🏦 🔟 🕡 🛽	mpon
Cypress Default Of 4 >	ent -
Default Component Catalog	្អ
Components	
state machine [v0.0]	2
	8
	1 <u>6</u>
	ğ
	ner
	lts)
state machine 1	
state_machine	
RIGHT LED_2	
ELEFT LED 3	

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)