Lab Manual for Digital Signal Controller

Lab1: Implement Timer and LED Driver and blink LED D3 on explorer 16 board using Microchip Code Configurator (MCC)

Pre-Requisite Needed:

System: A system with 16 GB RAM, Windows 10 Machine.
MPLABX – 6.20 Ver
Compiler - XC DSC 3.10 Ver

Purpose:

Upon finishing the lab, you will acquire a comprehensive understanding of how to utilize the MPLABX Integrated Development Environment (IDE) and the Microchip Code Configurator (MCC). The user will learn how to create a project, navigate the MCC User interface, and utilize MCC to develop a UART driver.

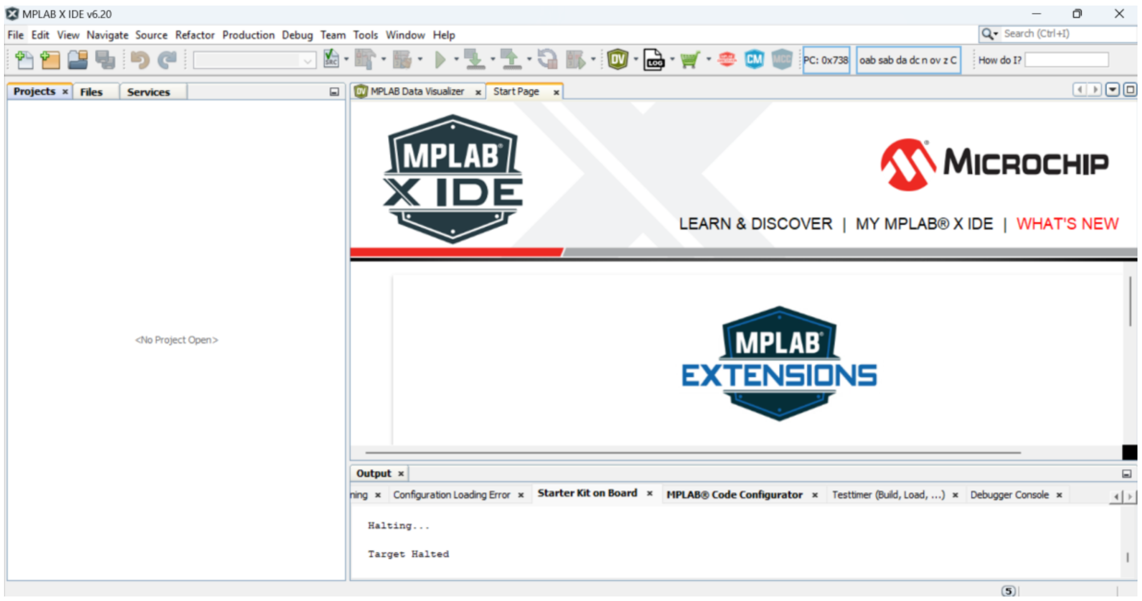
Overview:

The objective of the practical session is to familiarize the user with the Microchip Digital Signal Controller, as well as the Explorer 16 board and its interfaces. This guide aims to assist the user in comprehending the process of configuring the Timer in MCC and exploring the diverse applications accessible on the MPLABX tool.

Procedure:

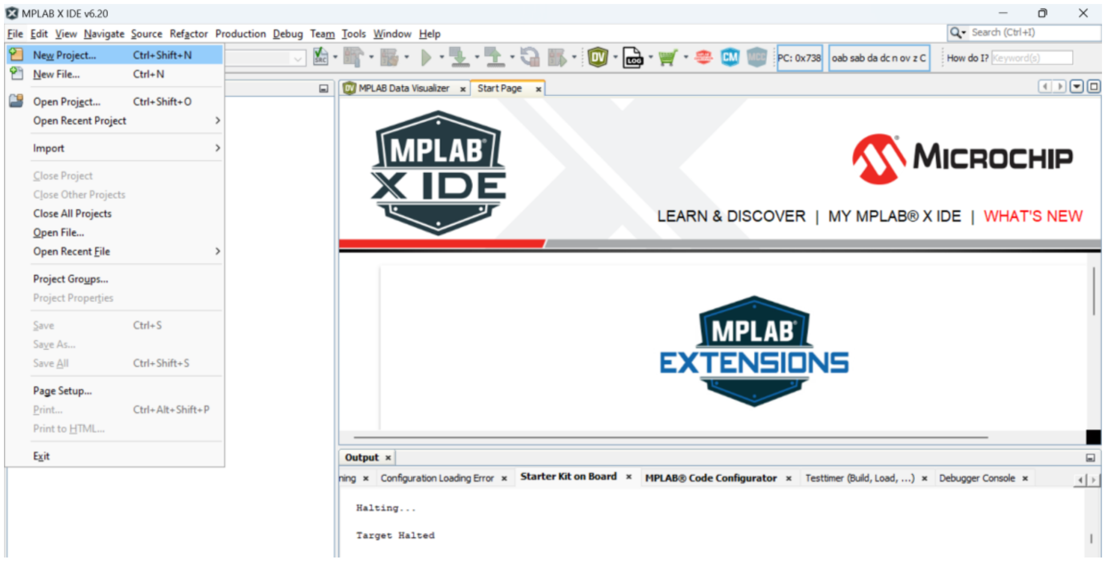
The laboratory experiment should be conducted by following the subsequent procedures.

STEP 1: Click on the MPLABX IDE which is available on Desktop. Once MPLABX windows opens you can see the below Figure.



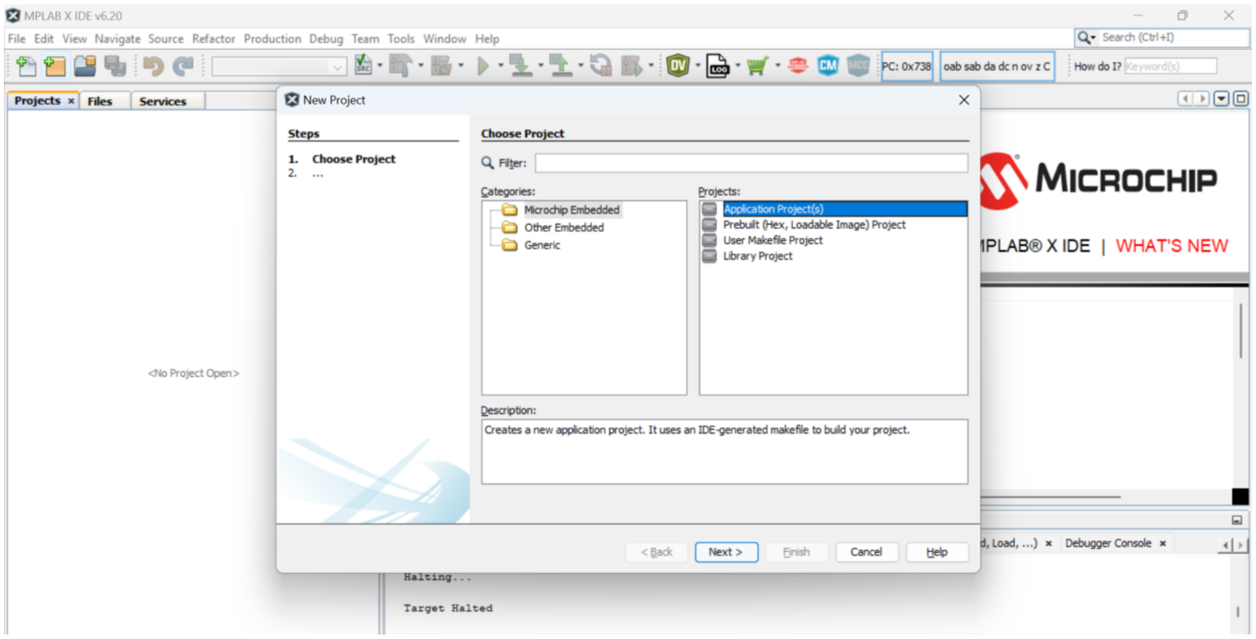
```
Halting...

Target Halted
```

STEP 2: Create a new project by clicking on "New" on the window.



```
Halting...

Target Halted
```

STEP 3: Once you click the new project option, the below figure will be displayed. Choose the application project option as shown below.


MPLAB X IDE v6.20
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Projects Files Services
<No Project Open>

New Project

Steps
1. Choose Project
2. ...

Choose Project

Filter:

Categories:
- Microchip Embedded
- Other Embedded
- Generic

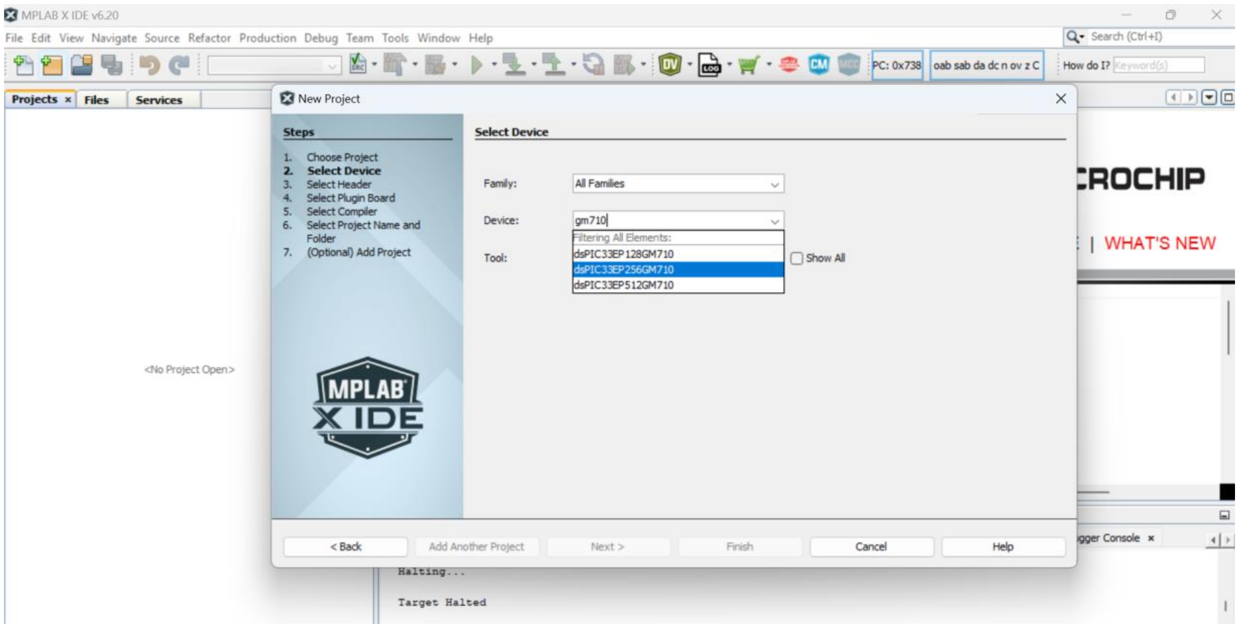
Projects:
- Application Project(s)
- Prebuilt (Hex, Loadable Image) Project
- User Makefile Project
- Library Project

Description:
Creates a new application project. It uses an IDE-generated makefile to build your project.

< Back | Next > | Finish | Cancel | Help

```
Halting...

Target Halted
```

Step 4: Choose the appropriate device you are working on. Here type dspic33ep512GM710 in the device option, and choose the mentioned device.


MPLAB X IDE v6.20
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Projects Files Services
<No Project Open>
MPLAB X IDE

New Project

Steps
1. Choose Project
2. **Select Device**
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. Select Project Name and Folder
7. (Optional) Add Project

Select Device

Family: All Families

Device: gm710

Filtering All Elements:
- dsPIC33EP128GM710
- dsPIC33EP256GM710
- dsPIC33EP512GM710

Tool: Show All

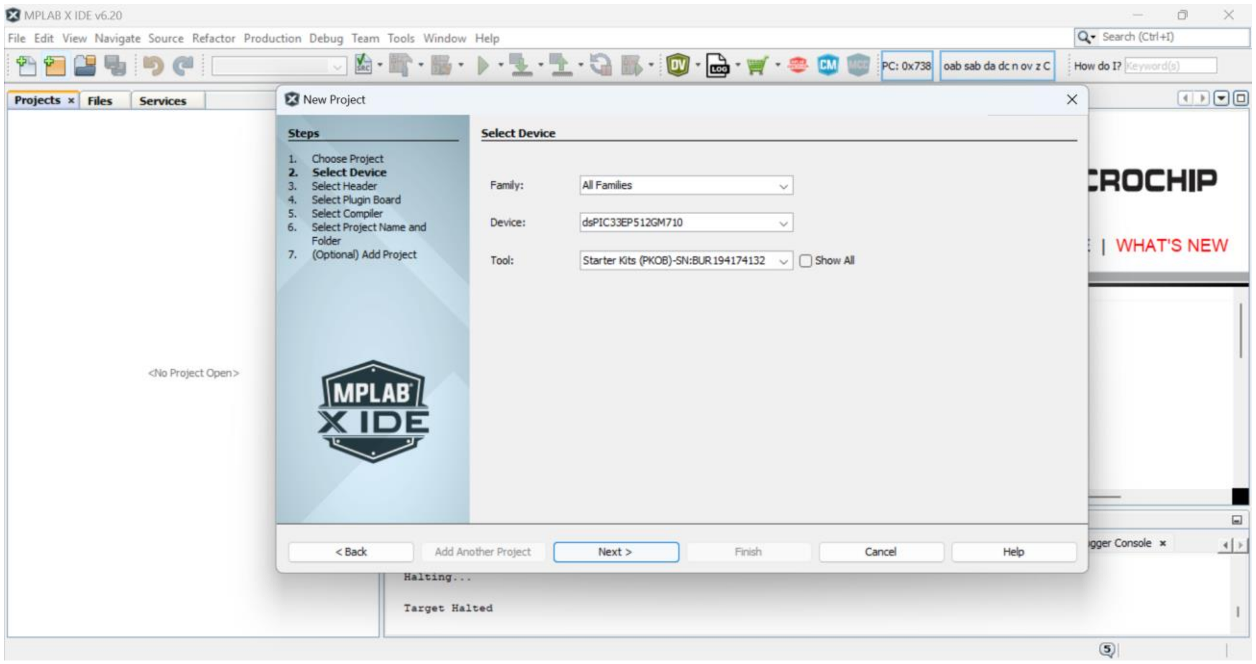
< Back | Add Another Project | Next > | Finish | Cancel | Help

```
Halting...

Target Halted
```

STEP 5: Once you have selected the device, proceed to the next step and choose "Debugger" as indicated below. The debugger option will only be accessible when the board is connected.

Alternatively, you can select the simulation option to work on. Subsequently, proceed to select the "Next" option.

.



Steps
1. Choose Project
2. **Select Device**
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. Select Project Name and Folder
7. (Optional) Add Project

Select Device

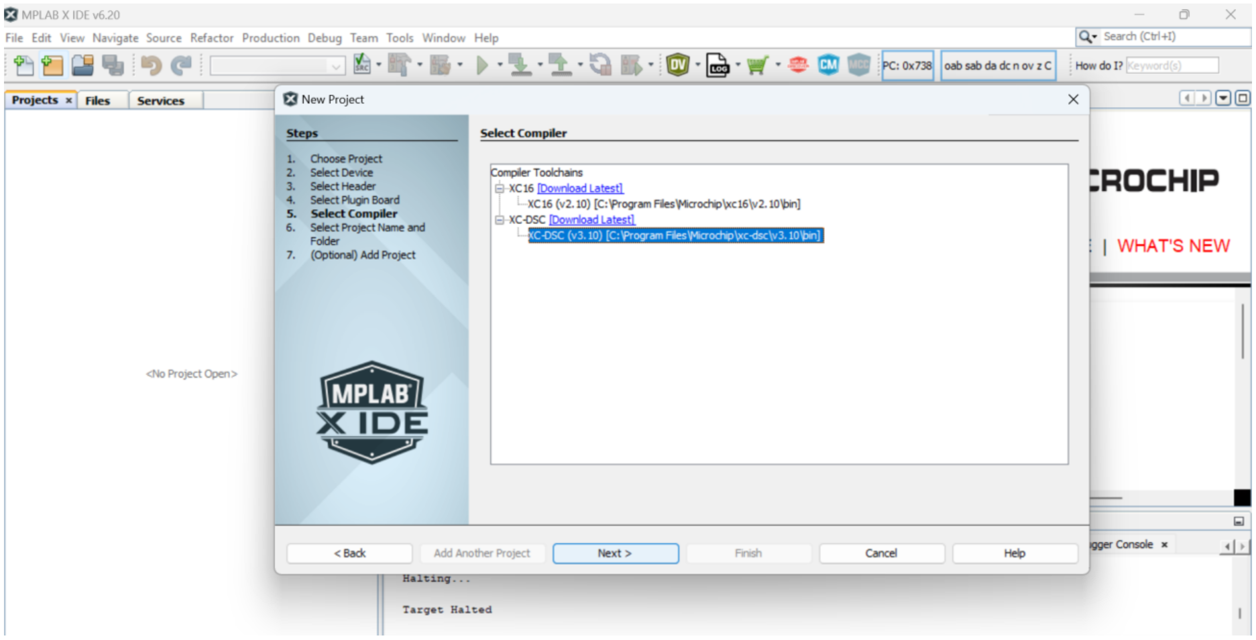
Family: All Families

Device: dsPIC33EP512GM710

Tool: Starter Kits (PKOB)-SN:BUR194174132 ☐ Show All

```
Halting...
Target Halted
```

STEP 6: In this step choose the compiler. Here you can choose XC16 compiler for PIC controllers and XC DSC for Digital Signal Controllers. Since dspic33ep512GM710 is Digital Controller, choose XC DSC compiler and press next.



Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. **Select Compiler**
6. Select Project Name and Folder
7. (Optional) Add Project

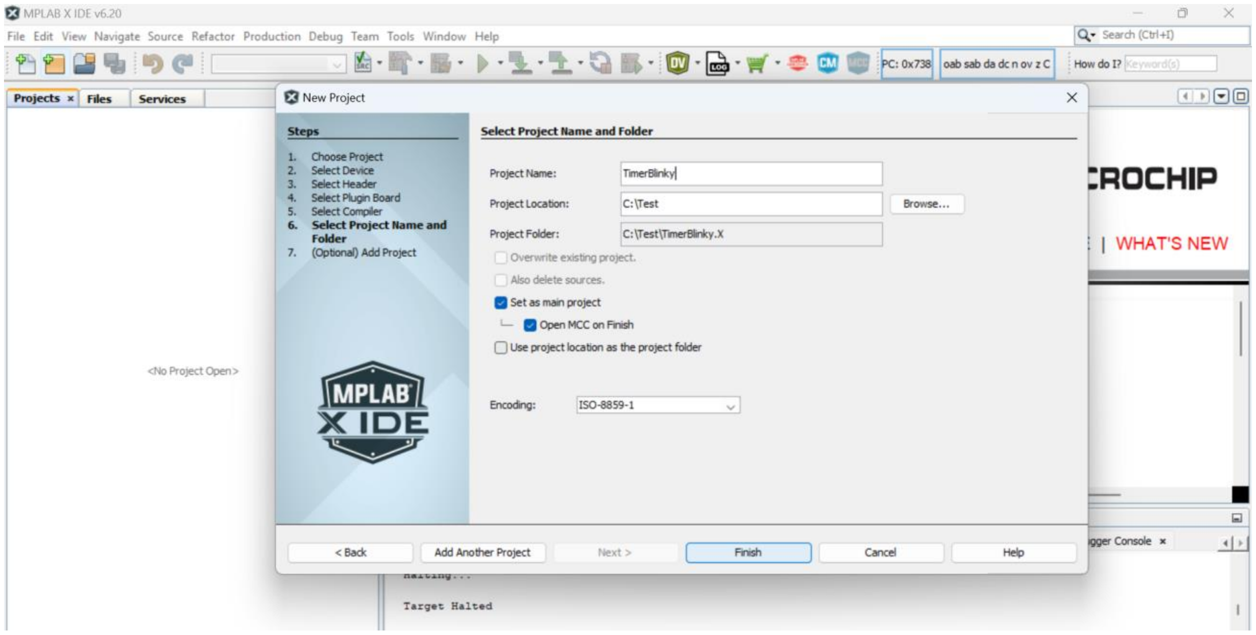
Select Compiler

Compiler Toolchains
- XC16 [Download Latest]
  - XC16 (v2.10) [C:\Program Files\Microchip\xc16\v2.10\bin]
- XC-DSC [Download Latest]
  - XC-DSC (v3.10) [C:\Program Files\Microchip\xc-dsc\v3.10\bin]

```
Halting...
Target Halted
```

STEP 7: When selecting the project path, ensure that you choose the correct folder. It is expected that you have already established a folder, enabling you to make a selection at this time.

.



Steps

1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. **Select Project Name and Folder**
7. (Optional) Add Project

Select Project Name and Folder

Project Name: TimerBlinky

Project Location: C:\Test

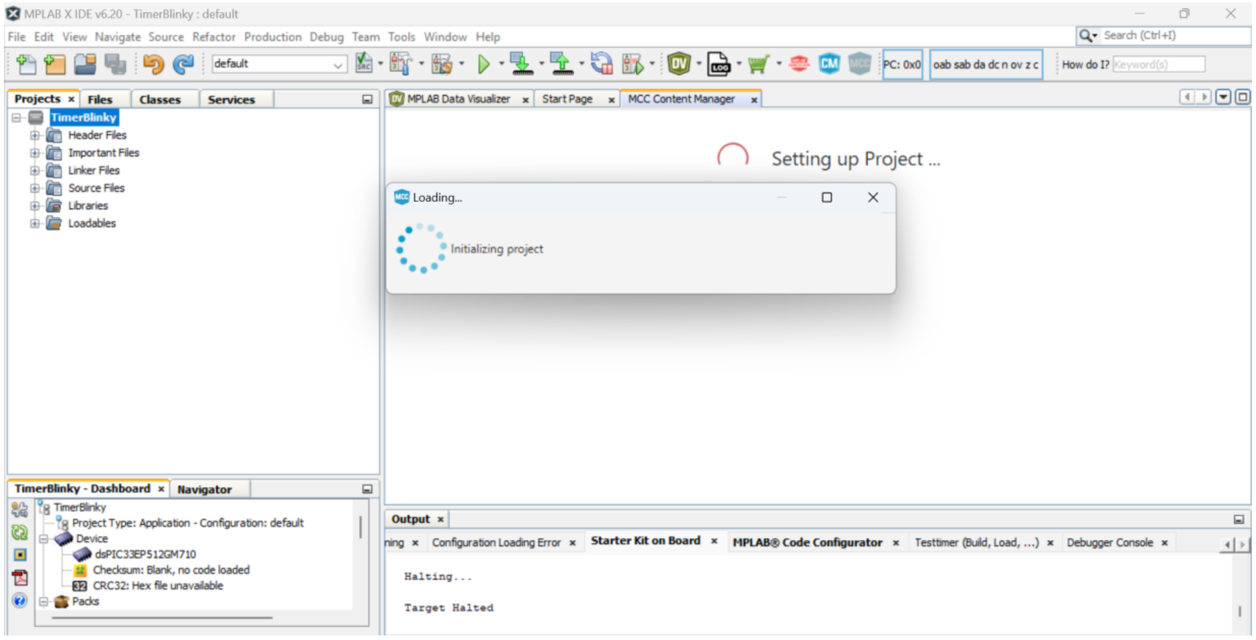
Project Folder: C:\Test\TimerBlinky.X

- [ ] Overwrite existing project.
- [ ] Also delete sources.
- [x] Set as main project
  - [x] Open MCC on Finish
- [ ] Use project location as the project folder

Encoding: ISO-8859-1

```
Halting...

Target Halted
```

After this click Finish and the IDE will create a new project for you.

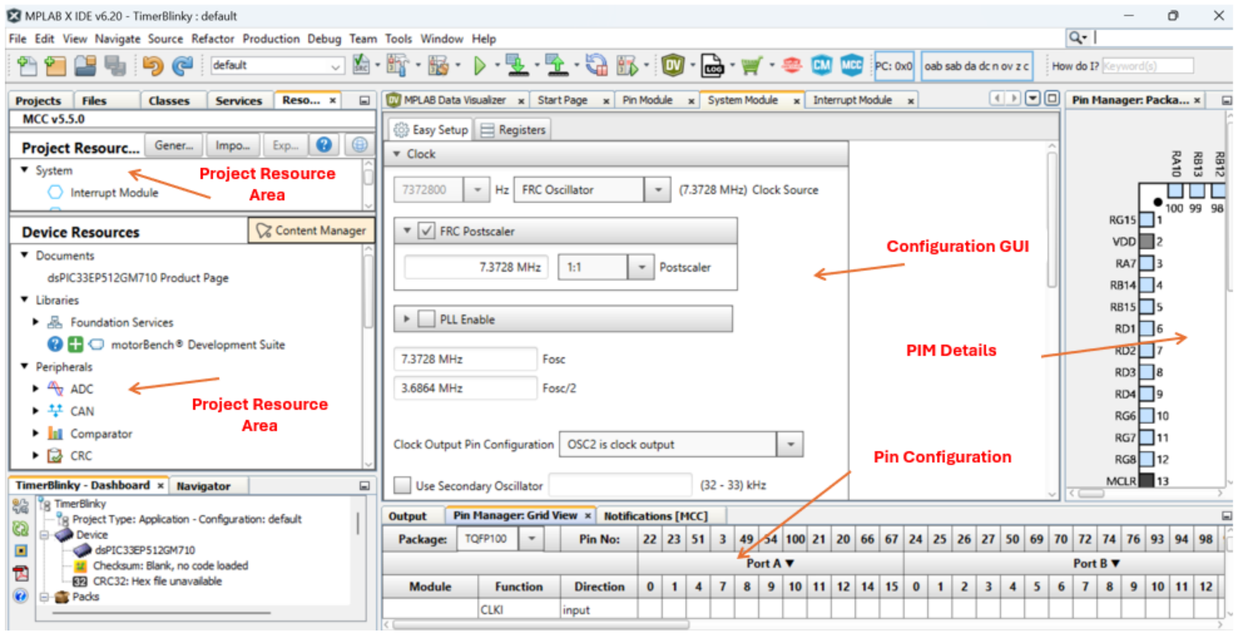

MPLAB X IDE v6.20 - TimerBlinky : default
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Projects Files Classes Services
MPLAB Data Visualizer
Start Page
MCC Content Manager
Setting up Project ...
Loading...
Initializing project
TimerBlinky - Dashboard
Navigator
Output
Configuration Loading Error
Starter Kit on Board
MPLAB® Code Configurator
Testtimer (Build, Load, ...)
Debugger Console

TimerBlinky
- Header Files
- Important Files
- Linker Files
- Source Files
- Libraries
- Loadables

TimerBlinky
- Project Type: Application - Configuration: default
- Device
  - dsPIC33EP512GM710
  - Checksum: Blank, no code loaded
  - CRC32: Hex file unavailable
- Packs

```
Halting...

Target Halted
```

STEP 8: Once the project is created, you can see the below Screen shot in the IDE. Please explore each of the labels mentioned in the screen shot.



Clock: 7372800 Hz, FRC Oscillator (7.3728 MHz) Clock Source

FRC Postscaler: 7.3728 MHz, 1:1 Postscaler

PLL Enable

7.3728 MHz Fosc

3.6864 MHz Fosc/2

Clock Output Pin Configuration: OSC2 is clock output

Use Secondary Oscillator (32 - 33) kHz

TimerBlinky
Project Type: Application - Configuration: default
Device
dsPIC33EP512GM710
Checksum: Blank, no code loaded
CRC32: Hex file unavailable
Packs

Package: TQFP100

Pin No:	22	23	51	3	49	54	100	21	20	66	67	24	25	26	27	50	69	70	72	74	76	93	94	98
	Port A ▼											Port B ▼												
Module / Function / Direction	0	1	4	7	8	9	10	11	12	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12
CLKI input																								

Pin	Name
1	RG15
2	VDD
3	RA7
4	RB14
5	RB15
6	RD1
7	RD2
8	RD3
9	RD4
10	RG6
11	RG7
12	RG8
13	MCLR
100	RA10
99	RB13
98	RB12

Project resources window will show you all the folders and files available under the project.

- It also clearly segregates between source and header files folder. Also, it generates the main.c file for application.

Resource Management Window

- Will list Peripherals that are selected for this project
- **The system** window will list the system setup settings.
- **Device Resource** window will list down all the available peripherals for the given device.

Pin Manager Package View

- This gives an overview of the pins of the chosen controller.
- It gives the appropriate pin mapping details.

Output Window

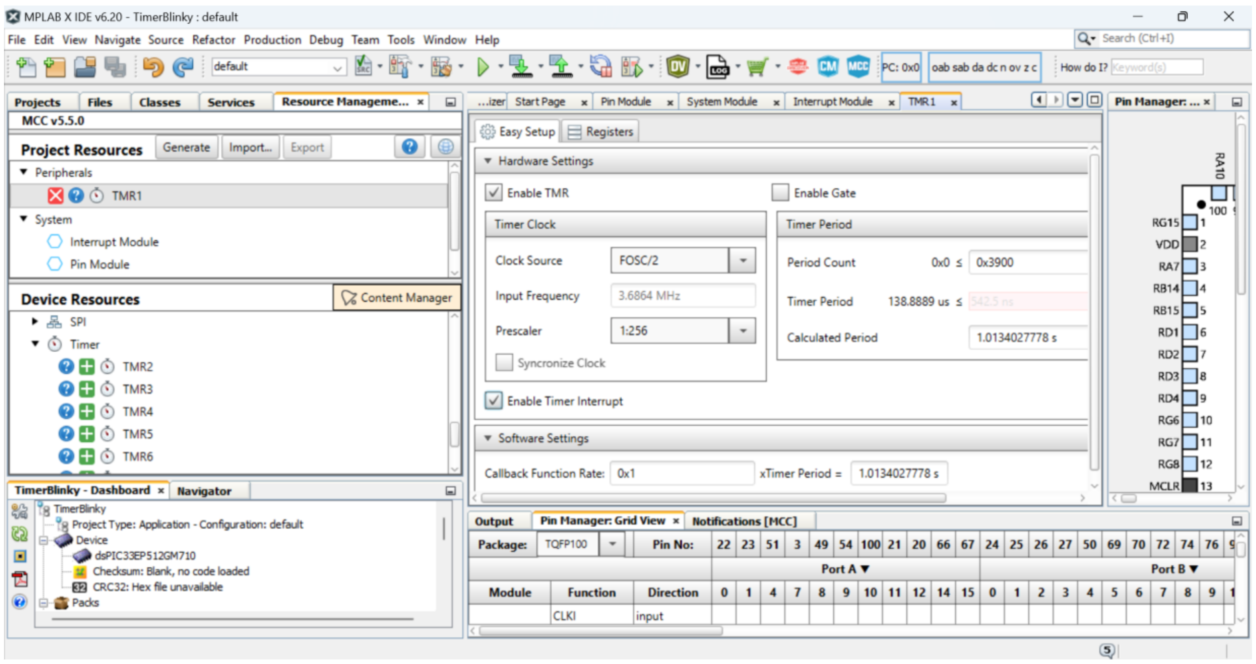
- This gives message details of all the process that is happening during compilation, output, error messages etc.

Pin Manager Grid View

- This gives the various pins details of the selected module.

These are the major details about MCC. The remaining thing as we get on to the exercise will be explained.

STEP 9: In the Device resource window, navigate downwards to locate the Timer module. If you click on it, additional timer modules will be displayed. Select TMR1 in that case. Once you make a selection, the tmr1 module project resources will be displayed in the Peripheral Section.


MPLAB X IDE v6.20 - TimerBlinky : default
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Projects Files Classes Services Resource Manageme...
MCC v5.5.0
Project Resources Generate Import... Export
Peripherals
TMR1
System
Interrupt Module
Pin Module
Device Resources Content Manager
SPI
Timer
TMR2
TMR3
TMR4
TMR5
TMR6
TimerBlinky - Dashboard Navigator
TimerBlinky
Project Type: Application - Configuration: default
Device
dsPIC33EP512GM710
Checksum: Blank, no code loaded
CRC32: Hex file unavailable
Packs
Start Page Pin Module System Module Interrupt Module TMR1
Easy Setup Registers
Pin Manager
Output Pin Manager: Grid View Notifications [MCC]
Package: TQFP100 Pin No:
Port A Port B
Module Function Direction
CLKI input

Hardware Settings

- [✓] Enable TMR
- [ ] Enable Gate

Timer Clock

Clock Source	FOSC/2
Input Frequency	3.6864 MHz
Prescaler	1:256

- [ ] Syncronize Clock

Timer Period

Period Count	0x0 ≤ 0x3900
Timer Period	138.8889 us ≤ 542.5 ns
Calculated Period	1.0134027778 s

- [✓] Enable Timer Interrupt

Software Settings

Callback Function Rate: 0x1 xTimer Period = 1.0134027778 s

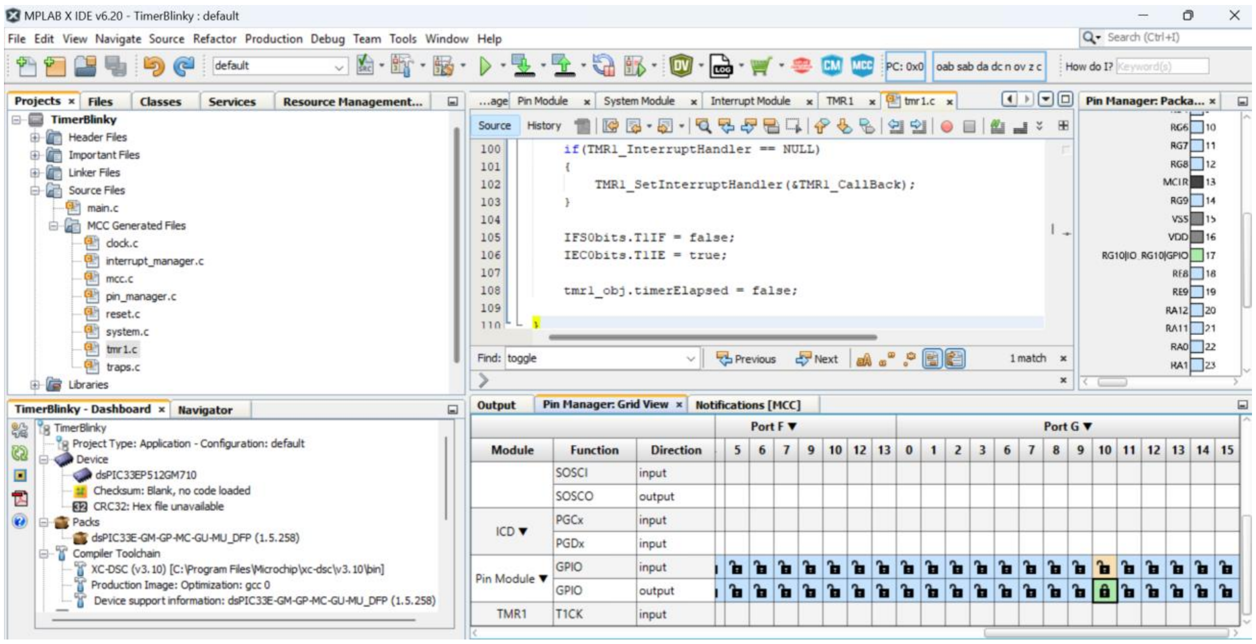
STEP 10: On the right-hand side, in the easy setup window all the timer setup options are visible. Since we are creating a 1s timer, choose the prescaler value to 1:256.

Subsequently, assign a numerical value to represent the count of periods. Modifying the count value triggers the calculation of the period, which is then shown in the designated calculated period box. A delay of 1 second will be displayed, and it should be acceptable. Otherwise, provide the appropriate value.

STEP 11: Here in the screen shot a value 0x3900 is given, which calculates to approximately 1s time, which is what required.

STEP 12: In this experiment we need to blink LED D3 for 1s. Need to choose the D3 LED Pin. So refer the circuit diagram of the board https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/BoardDesignFiles/Explorer_16_32_Schematics_R6_3.pdf

As per the circuit diagram Pin 17 is connected to LED D3. When you see the Pin 17 in Pin Manager Window, it is mapped to RG10 pin. So, in the Pin Manager Grid choose RG10 as output as shown.

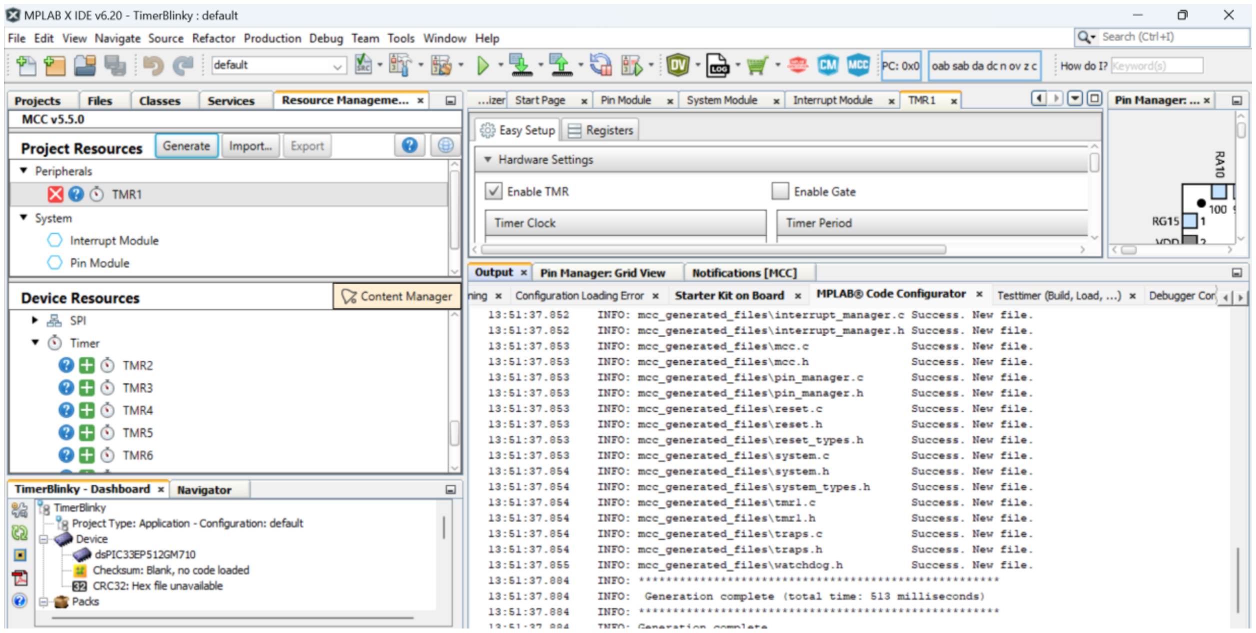


```
if(TMR1_InterruptHandler == NULL)
{
    TMR1_SetInterruptHandler(&TMR1_CallBack);
}

IFS0bits.T1IF = false;
IEC0bits.T1IE = true;

tmr1_obj.timerElapsed = false;
}
```

Find: toggle — 1 match

			Port F ▼							Port G ▼													
Module	Function	Direction	5	6	7	9	10	12	13	0	1	2	3	6	7	8	9	10	11	12	13	14	15
	SOSCI	input																					
	SOSCO	output																					
ICD ▼	PGCx	input																					
	PGDx	input																					
Pin Module ▼	GPIO	input	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓
	GPIO	output	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔒	🔓	🔓	🔓	🔓	🔓
TMR1	T1CK	input																					

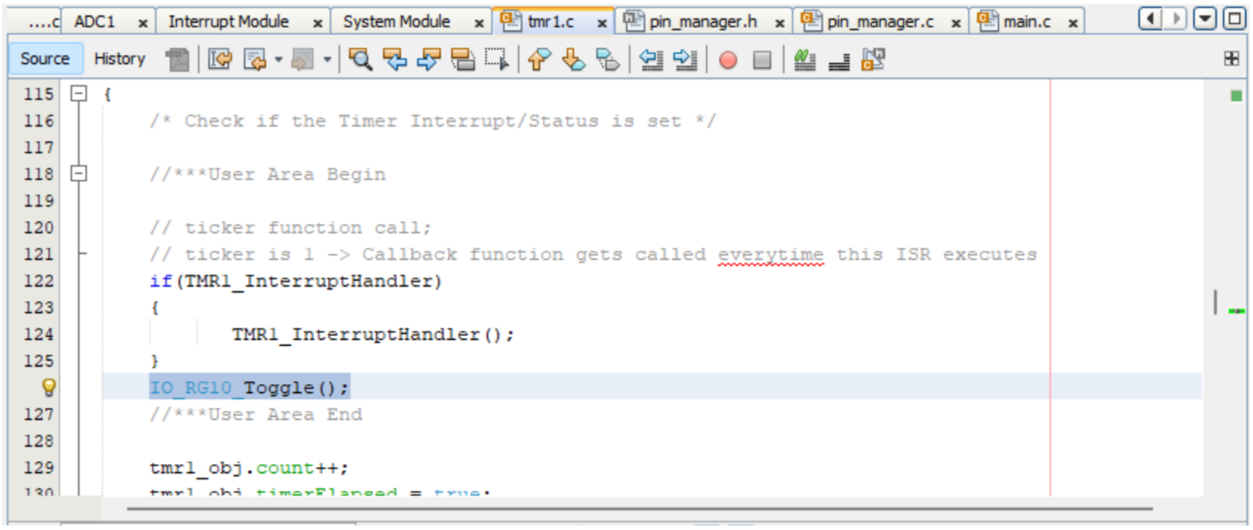
STEP 13: Now generate the code by clicking "generate" button in the Project Resource space. It will generate the code as shown below.



```
13:51:37.852    INFO: mcc_generated_files\interrupt_manager.c Success. New file.
13:51:37.852    INFO: mcc_generated_files\interrupt_manager.h Success. New file.
13:51:37.853    INFO: mcc_generated_files\mcc.c               Success. New file.
13:51:37.853    INFO: mcc_generated_files\mcc.h               Success. New file.
13:51:37.853    INFO: mcc_generated_files\pin_manager.c       Success. New file.
13:51:37.853    INFO: mcc_generated_files\pin_manager.h       Success. New file.
13:51:37.853    INFO: mcc_generated_files\reset.c             Success. New file.
13:51:37.853    INFO: mcc_generated_files\reset.h             Success. New file.
13:51:37.853    INFO: mcc_generated_files\reset_types.h       Success. New file.
13:51:37.853    INFO: mcc_generated_files\system.c            Success. New file.
13:51:37.854    INFO: mcc_generated_files\system.h            Success. New file.
13:51:37.854    INFO: mcc_generated_files\system_types.h      Success. New file.
13:51:37.854    INFO: mcc_generated_files\tmr1.c              Success. New file.
13:51:37.854    INFO: mcc_generated_files\tmr1.h              Success. New file.
13:51:37.854    INFO: mcc_generated_files\traps.c             Success. New file.
13:51:37.854    INFO: mcc_generated_files\traps.h             Success. New file.
13:51:37.855    INFO: mcc_generated_files\watchdog.h          Success. New file.
13:51:37.884    INFO: *****************************************************
13:51:37.884    INFO:  Generation complete (total time: 513 milliseconds)
13:51:37.884    INFO: *****************************************************
13:51:37.884    INFO: Generation complete
```

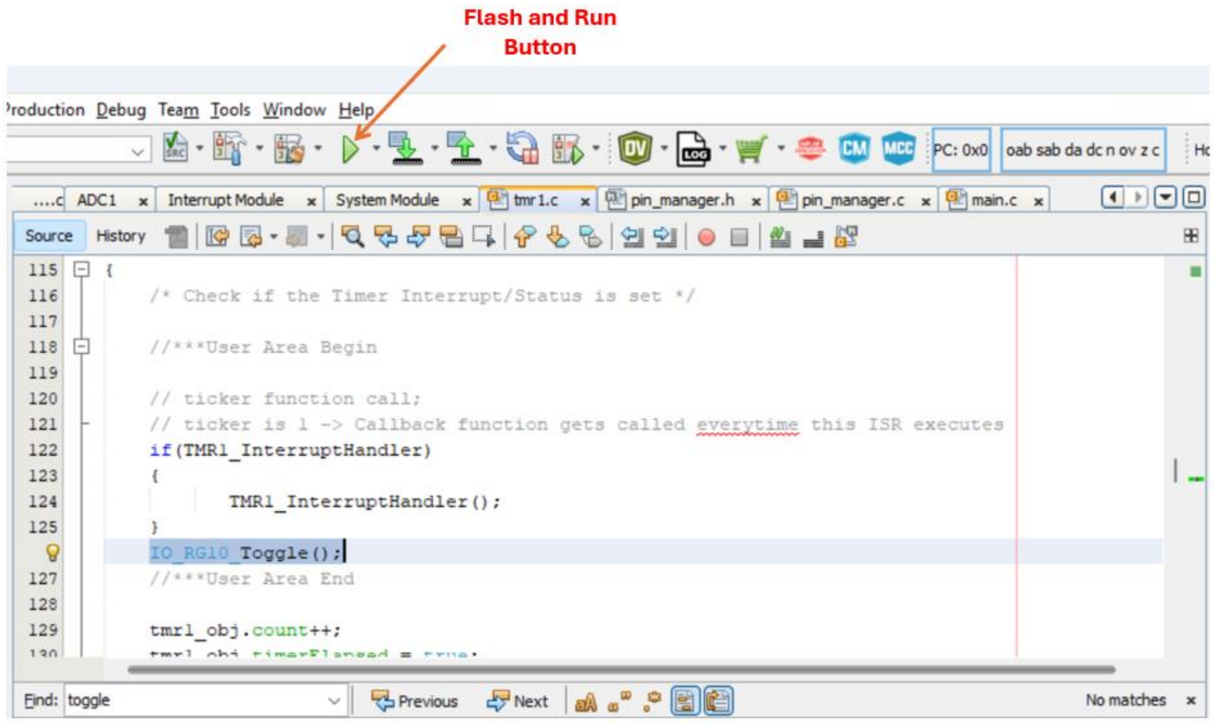
Step 14: Now choose the project window, in the mcc_generated folder open tmr1.c. In that consider the _T1Interrupt(), navigate to " //User Area End" and the macro IO_RG10_Toggle(); (

This macro can be searched in pin_manager.h file). Also, Please Open the pin_manager.h file and look for the above macro and add it in tmr1.c



```
{
    /* Check if the Timer Interrupt/Status is set */

    //***User Area Begin

    // ticker function call;
    // ticker is 1 -> Callback function gets called everytime this ISR executes
    if(TMR1_InterruptHandler)
    {
        TMR1_InterruptHandler();
    }
    IO_RG10_Toggle();
    //***User Area End

    tmr1_obj.count++;
```

Step 15: Now run the program, compile the program and flash the program by clicking the Green button.



```
{
    /* Check if the Timer Interrupt/Status is set */

    //***User Area Begin

    // ticker function call;
    // ticker is 1 -> Callback function gets called everytime this ISR executes
    if(TMR1_InterruptHandler)
    {
        TMR1_InterruptHandler();
    }
    IO_RG10_Toggle();
    //***User Area End

    tmr1_obj.count++;
```

STEP 16: After programming onto flash, now you should be able to see the LED D3 is blinking.

Lab2: Implement UART Driver and Application to Print a Character on the Terminal using Microchip Code Configurator (MCC)

Purpose:

You will have a comprehensive understanding of the MPLABX IDE and MCC once you have finished the lab. Additionally, the user will be familiar with the process of creating a project, navigating the MCC user interface, and collaborating with MCC to develop a UART driver.

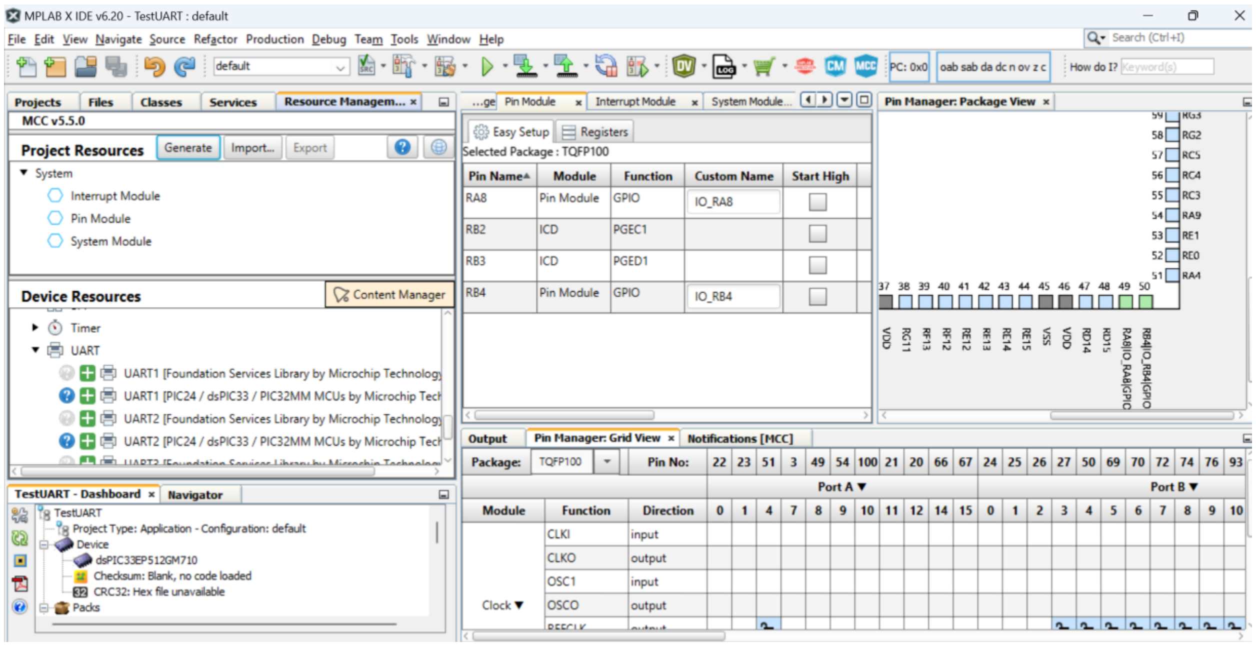
Overview:

The objective of the hands-on session is to acquaint the user with the user interfaces of the Microchip Digital Signal Controller and Explorer 16 board. This will assist the user in comprehending the process of configuring the UART in MCC and in exploring the diverse applications that are available on the MPLABX tool.

Procedure

STEP1 – STEP 8 the procedure remains same as LAB 1 activity. Please follow the same.

STEP 9: Navigate through Device resources options and choose the UART1 peripheral as shown in the below figure.


MPLAB X IDE v6.20 - TestUART : default
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Projects Files Classes Services Resource Managem...
MCC v5.5.0
Generate Import... Export
Content Manager
Pin Module Interrupt Module System Module...
Easy Setup Registers
Pin Manager: Package View
Output Pin Manager: Grid View Notifications [MCC]

Project Resources
- System
  - Interrupt Module
  - Pin Module
  - System Module

Device Resources
- Timer
- UART
  - UART1 [Foundation Services Library by Microchip Technology
  - UART1 [PIC24 / dsPIC33 / PIC32MM MCUs by Microchip Tech
  - UART2 [Foundation Services Library by Microchip Technology
  - UART2 [PIC24 / dsPIC33 / PIC32MM MCUs by Microchip Tech

TestUART - Dashboard
- TestUART
  - Project Type: Application - Configuration: default
  - Device
    - dsPIC33EP512GM710
    - Checksum: Blank, no code loaded
    - CRC32: Hex file unavailable
  - Packs

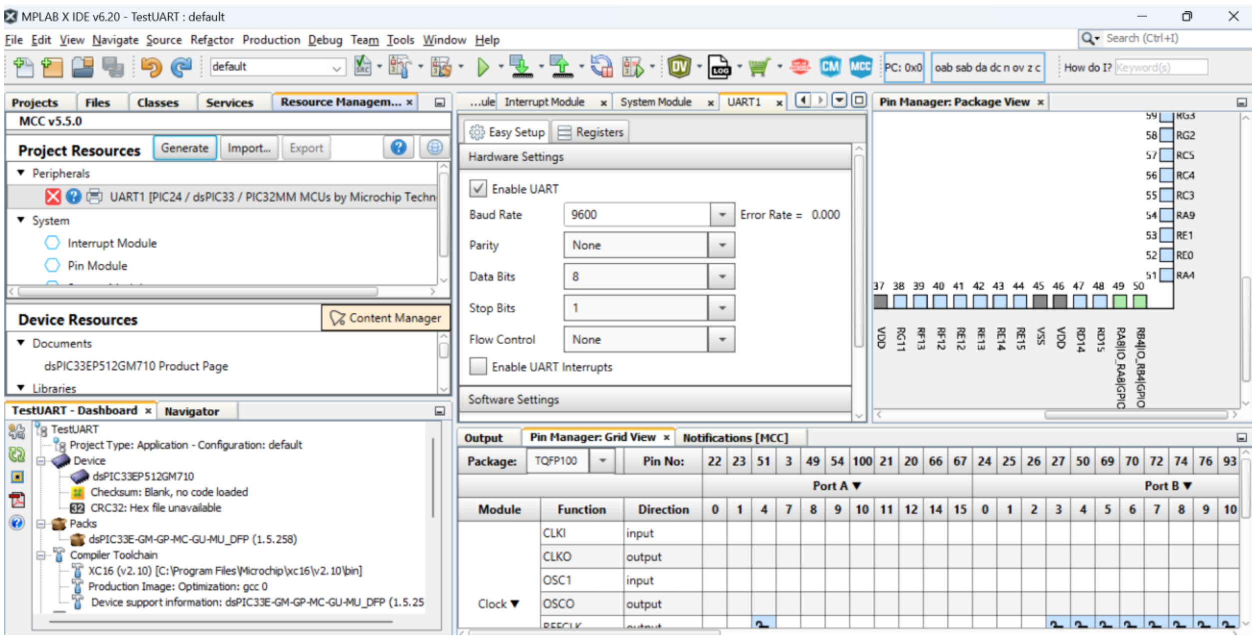
Selected Package : TQFP100

Pin Name	Module	Function	Custom Name	Start High
RA8	Pin Module	GPIO	IO_RA8	☐
RB2	ICD	PGEC1		☐
RB3	ICD	PGED1		☐
RB4	Pin Module	GPIO	IO_RB4	☐

Package: TQFP100

Module	Function	Direction
Clock	CLKI	input
	CLKO	output
	OSC1	input
	OSCO	output

STEP 10: Choose UART1 peripheral as shown and, in the figure, and Enable UART as shown in the figure.


MPLAB X IDE v6.20 - TestUART : default

MCC v5.5.0

Project Resources
- Peripherals
  - UART1 [PIC24 / dsPIC33 / PIC32MM MCUs by Microchip Techn
- System
  - Interrupt Module
  - Pin Module

Device Resources
- Documents
  - dsPIC33EP512GM710 Product Page
- Libraries

TestUART - Dashboard
- TestUART
  - Project Type: Application - Configuration: default
  - Device
    - dsPIC33EP512GM710
    - Checksum: Blank, no code loaded
    - CRC32: Hex file unavailable
  - Packs
    - dsPIC33E-GM-GP-MC-GU-MU_DFP (1.5.258)
  - Compiler Toolchain
    - XC16 (v2.10) [C:\Program Files\Microchip\xc16\v2.10\bin]
    - Production Image: Optimization: gcc 0
    - Device support information: dsPIC33E-GM-GP-MC-GU-MU_DFP (1.5.25

UART1 — Easy Setup — Hardware Settings

Enable UART	✓	
Baud Rate	9600	Error Rate = 0.000
Parity	None	
Data Bits	8	
Stop Bits	1	
Flow Control	None	
Enable UART Interrupts	☐	

Software Settings

Pin Manager: Grid View — Package: TQFP100

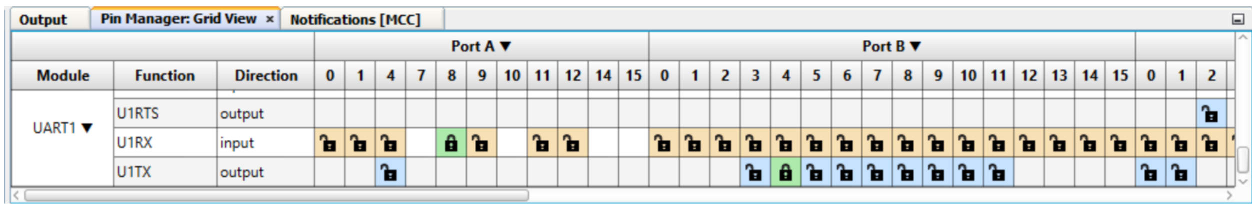
Module	Function	Direction	Pin No: 22	23	51	3	49	54	100	21	20	66	67	24	25	26	27	50	69	70	72	74	76	93
			Port A ▼											Port B ▼										
			0	1	4	7	8	9	10	11	12	14	15	0	1	2	3	4	5	6	7	8	9	10
Clock ▼	CLKI	input																						
	CLKO	output																						
	OSC1	input																						
	OSCO	output																						

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Projects Files Classes Services Resource Managem...
Generate Import... Export
Content Manager
Interrupt Module System Module UART1
Easy Setup Registers
Output Pin Manager: Grid View Notifications [MCC]
Pin Manager: Package View
37 38 39 40 41 42 43 44 45 46 47 48 49 50
VDD RG11 RF13 RF12 RE12 RE13 RE14 RE15 VSS VDD RD14 RD15 RA8|IO_RA8|GPIO RB4|IO_RB4|GPIO
58 RG2 57 RC5 56 RC4 55 RC3 54 RA9 53 RE1 52 RE0 51 RA4

STEP 11: Please check the hardware circuit and identify the Rx and Tx pin connections. The circuit diagram can be found in the below path. https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/BoardDesignFiles/Explorer_16_32_Schematics_R6_3.pdf

As per the circuit Pin 49 is connected to Rx and Pin 50 is connected to Tx.

When you see Pin **49 and Pin 50** in Pin Manager Window, it is mapped to **RA8** pin and **RB4** respectively. So, in the Pin Manager Grid choose **RA8 and RB4** from UART pin selections as output. Now generate the code.


			Port A ▼											Port B ▼																		
Module	**Function**	**Direction**	0	1	4	7	8	9	10	11	12	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2
UART1 ▼	U1RTS	output																														🔓
	U1RX	input	🔓	🔓	🔓		🔒	🔓		🔓	🔓			🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓	🔓
	U1TX	output			🔓												🔓	🔒	🔓	🔓	🔓	🔓	🔓	🔓	🔓					🔓	🔓	

STEP 12: In the generated code, Now click on main.c and open the file. Pls refer the Family Reference Manual of dspic series of controller. It can be found in the below path: https://ww1.microchip.com/downloads/en/DeviceDoc/70000582e.pdf

Goto example 5.1 on page 18. In that it can be observed that a delay of 105us needs to be introduced after the enabling of UART. It can be found inside the program in the comment.

```
/* Wait at least 105 microseconds (1/9600) before sending first char */
```

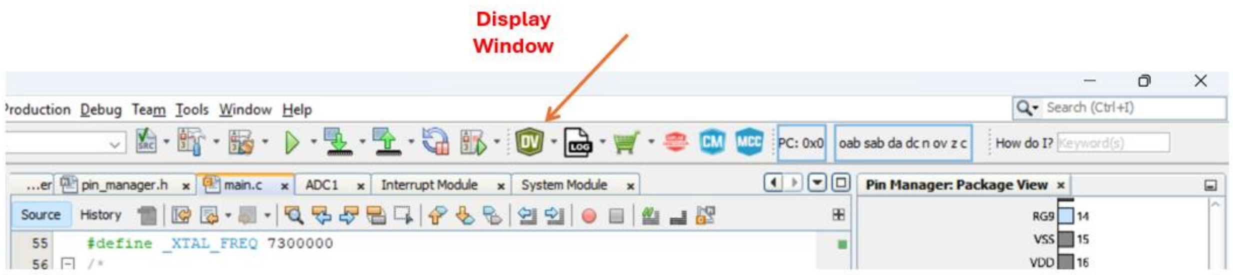
So, in the main program after initialization, please introduce a delay using a for loop as shown below.

```
unsigned int i;
while (i<600) {
   Nop(); // 105 us delay
   i++;
   }
```

And in the infinite loop please add the below code.

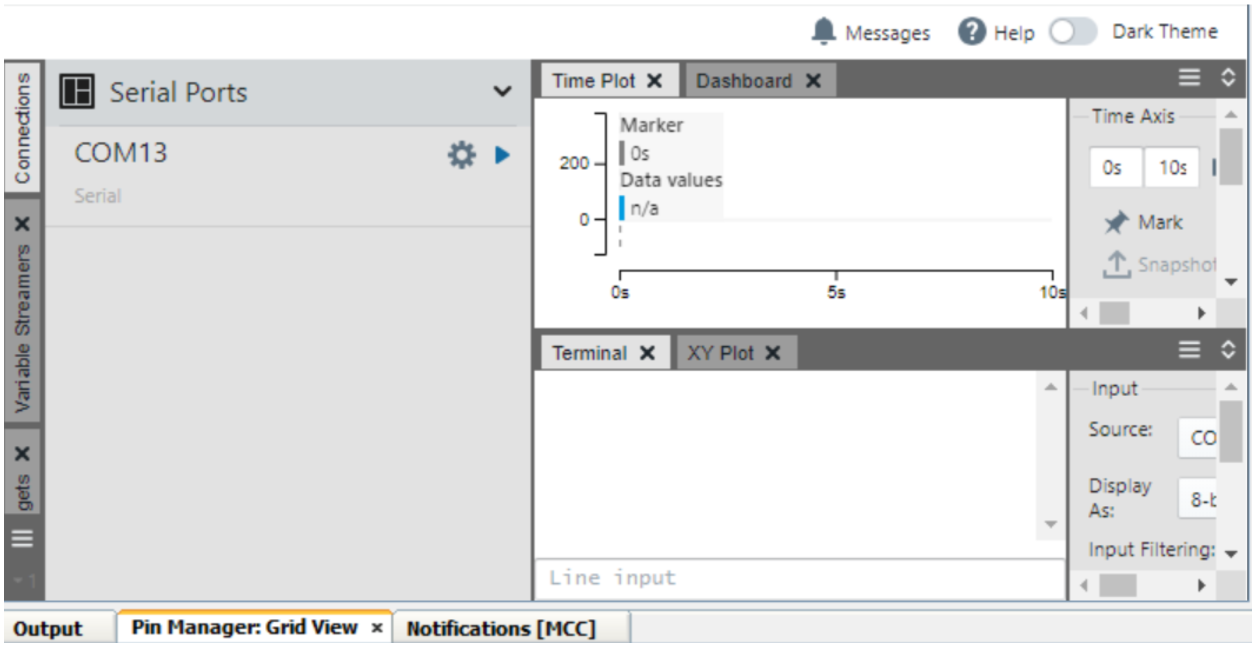
```
if(UART1_IsTxReady() == 1)
       {
        UART1_Write('a');
       }
```

STEP 13: Now click on main.c and open the file. Pls refer the Family Reference Manual of dspic series of controller. It can be found in the below path:


Display Window
Production Debug Team Tools Window Help
Search (Ctrl+I)
PC: 0x0
oab sab da dc n ov z c
How do I? Keyword(s)
...er pin_manager.h main.c ADC1 Interrupt Module System Module
Pin Manager: Package View
Source History
RG9 14
VSS 15
VDD 16

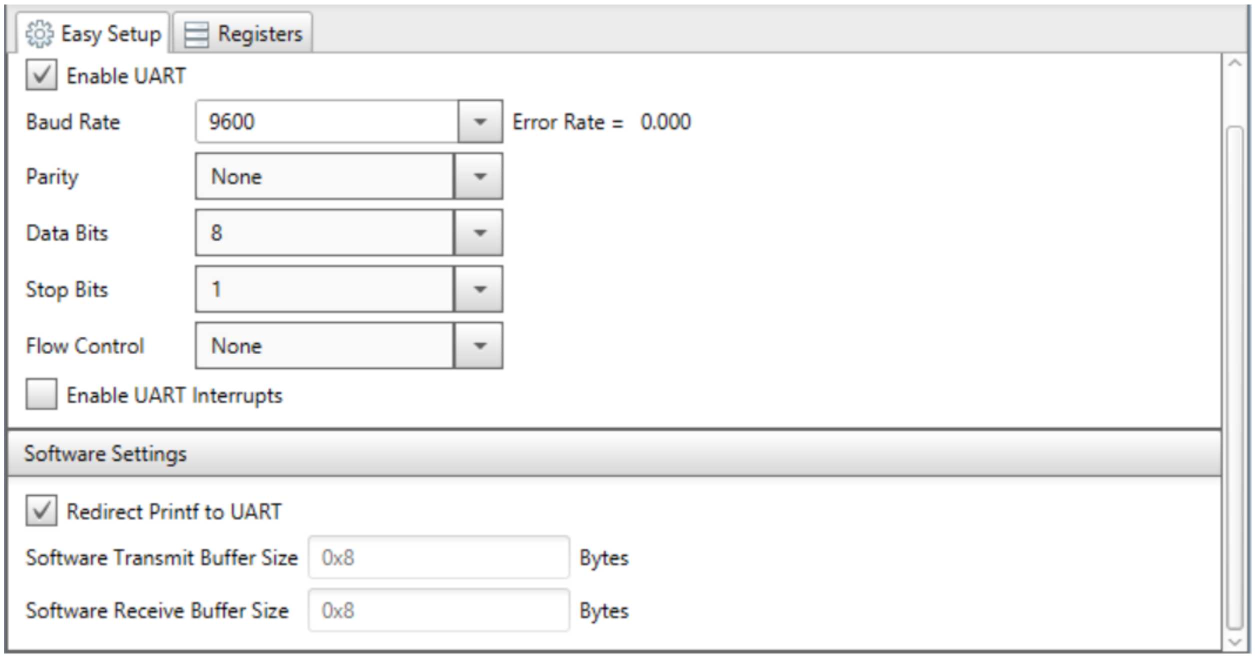
```
#define _XTAL_FREQ 7300000
/*
```

Once the Display window is opened, the screen shot will be displayed as illustrated below. It will automatically display the COM PORT that is connected. To commence the capture of the display on the COM13, click the blue-colored capture button. The terminal should display the characters 'a' and 'a' scrolling. This is the anticipated result.



Next Exercise:

1. Enable UART " Redirect Print Option to UART" and try printing the UART prints.



- [✓] Enable UART

Baud Rate	9600	Error Rate = 0.000
Parity	None	
Data Bits	8	
Stop Bits	1	
Flow Control	None	

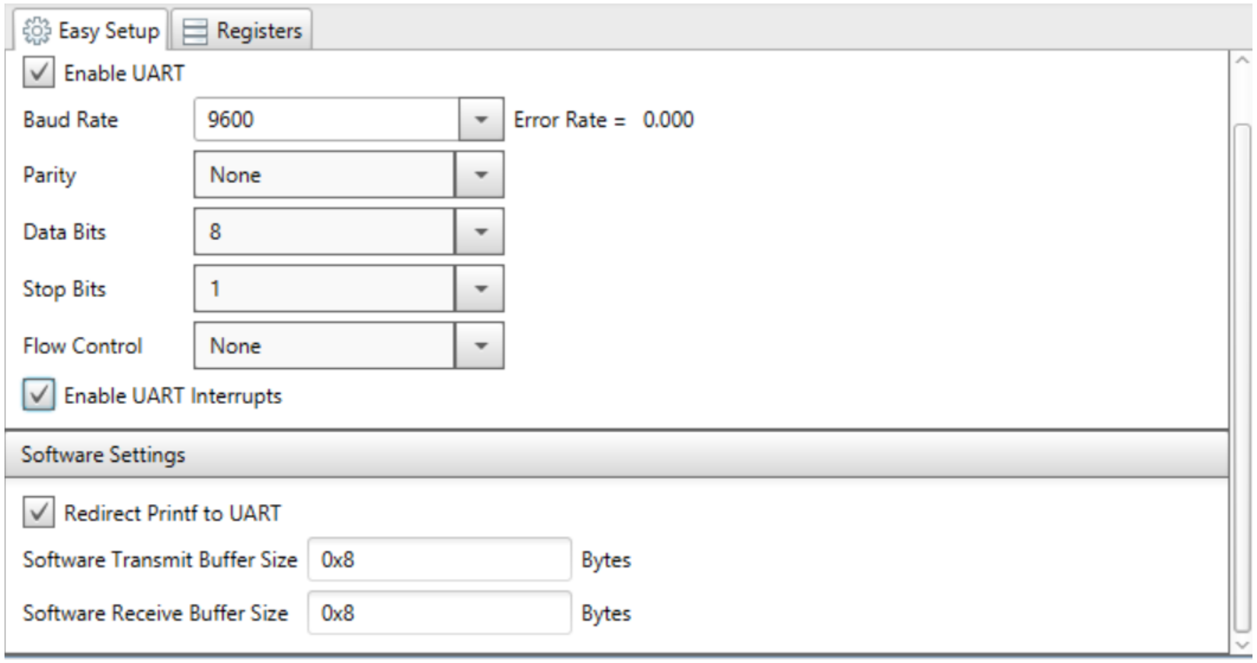
- [ ] Enable UART Interrupts

Software Settings

- [✓] Redirect Printf to UART

Software Transmit Buffer Size	0x8	Bytes
Software Receive Buffer Size	0x8	Bytes

2. "Enable UART Interrupts " and try printing the UART prints, after regenerate the code.


Easy Setup | Registers

Enable UART [✓]

Baud Rate: 9600 — Error Rate = 0.000

Parity: None

Data Bits: 8

Stop Bits: 1

Flow Control: None

Enable UART Interrupts [✓]

Software Settings

Redirect Printf to UART [✓]

Software Transmit Buffer Size: 0x8 Bytes

Software Receive Buffer Size: 0x8 Bytes

Lab3: Implement Switch interface and accordingly and LED Blink if a switch is pressed using MCC.

Purpose:

You will have a comprehensive understanding of the MPLABX IDE and MCC once you have finished the lab. Additionally, the user will be familiar with the process of creating a project, navigating the MCC user interface, and collaborating with MCC to develop a switch and LED driver.

Overview:

The objective of the hands-on session is to acquaint the user with the user interfaces of the Microchip Digital Signal Controller and Explorer 16 board. This will assist the user in comprehending the process of configuring the switch in MCC and in exploring the diverse applications that are available on the MPLABX tool.

Procedure

STEP1 – STEP 8 the procedure remains same as LAB 1 and LAB2 activity. Please follow the same.

STEP 9: Please check the hardware circuit and identify the Switch connection and corresponding LED pin connections. The circuit diagram can be found in the below path. https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/BoardDesignFiles/Explorer_16_32_Schematics_R6_3.pdf

As per the circuit **Pin 83** is connected to **Switch S3** and **Pin 17** is connected to **LED D3**.

When you see **Pin 83 and Pin 17** in Pin Manager Window, it is mapped to **RD6** pin and **RG10** respectively. So, in the Pin Manager Grid choose **RD6 and RG10** from GPIO pin selections as input and output respectively.

STEP 10: Now generate the code.

STEP 11: In the generated code, open pin_manager.h from the header file section in mcc_generated files. In that identify the macros

“IO_RD6_GetValue()” , “IO_RG10_SetHigh()” and “IO_RG10_SetLow” Copy those macros.

STEP 12: Now open main.c file and add the identified macros in below code in the infinite while loop .

```
if(IO_RD6_GetValue()) {
    IO_RG10_SetHigh();
 } else {
    IO_RG10_SetLow();
    trigger_adc = true;
```

Please check the output by pressing Switch S3 and ensure the LED D3 is ‘ON’. This completes the experiment.