EE4120/EE4121 Introduction to Embedded Systems & Microcontrollers
Rayi Giri Varshini - EE22BTECH11215

Lab-1:

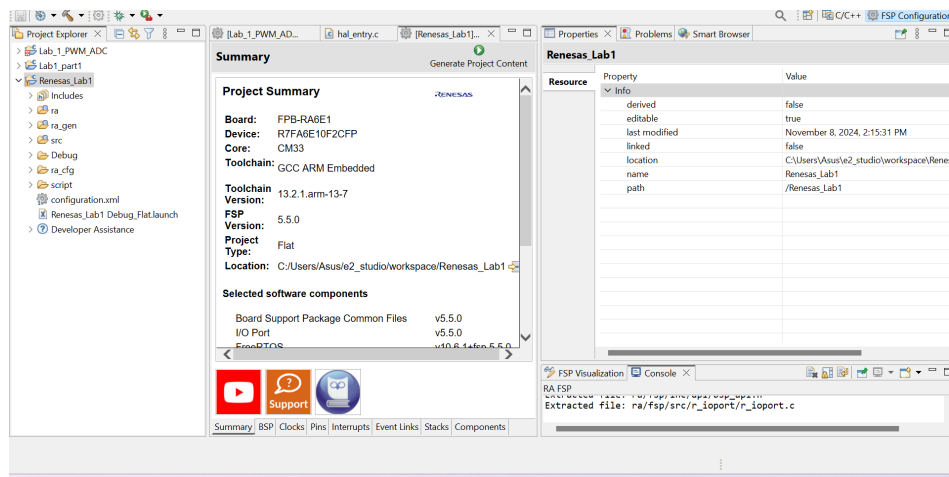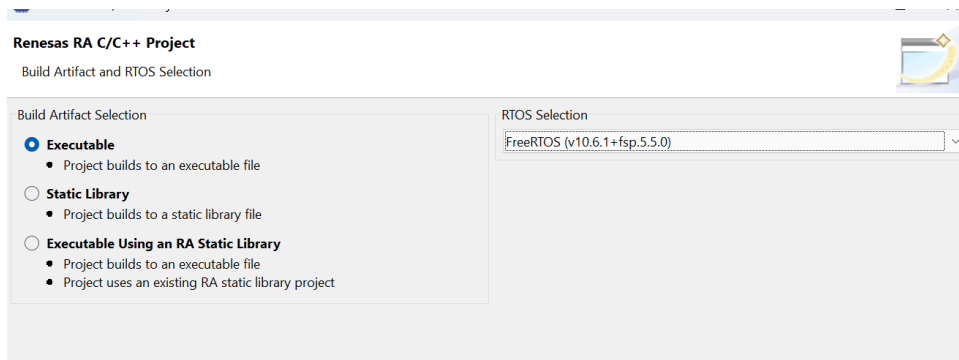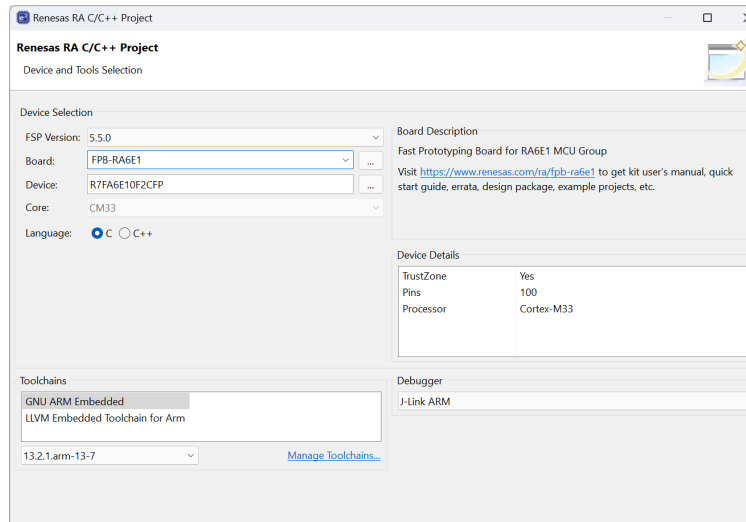Goal: Generating a sine PWM using Timer and ADC input for amplitude.
To generate a Pulse Width Modulated (PWM) signal where a sine wave modulates the duty cycle. The amplitude of the sine wave will be dynamically controlled by an ADC (Analog to Digital Converter) input, allowing real-time adjustment based on an external signal. The generated PWM signal is verified using an oscilloscope to ensure that both the modulation and amplitude control are functioning as intended.
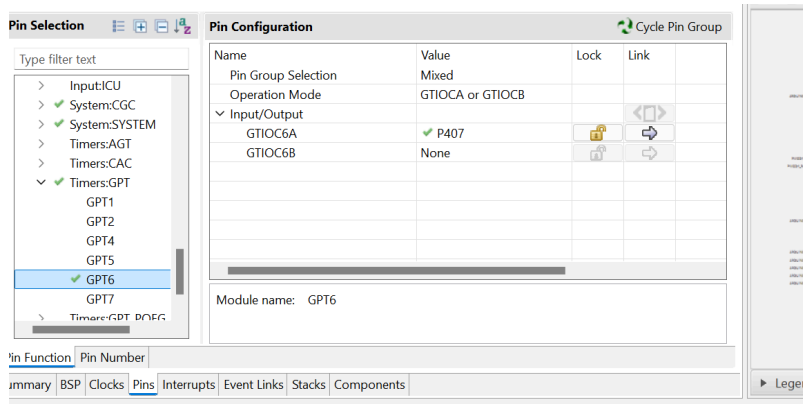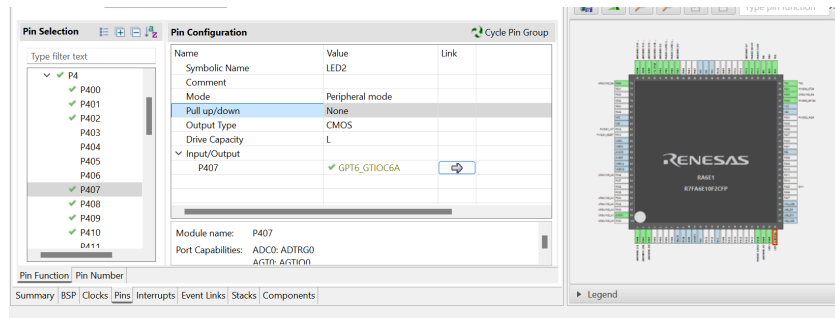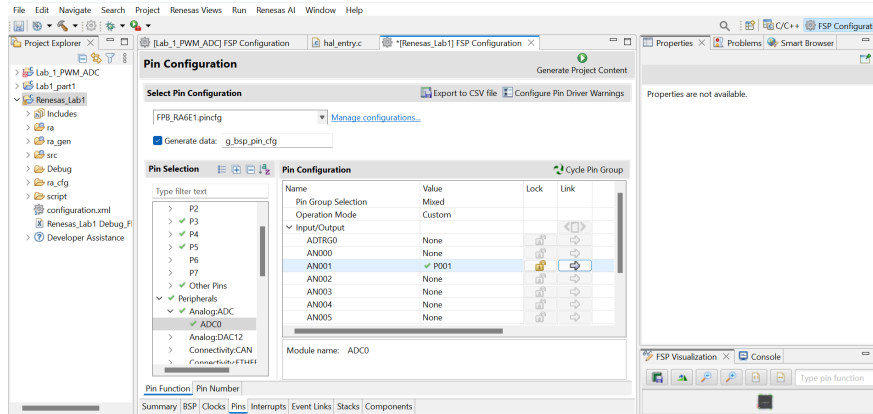
Board Used: Renesas FPB-RA6E1

PWM is a technique used to control the average voltage delivered to a load by varying the duty cycle. The duty cycle is the ratio of the high rime of the signal to tis total period. By modulating the duty cycle the output voltage can be controlled effectively. The ADC converts an external analog signal into a digital value. The digital value dynamically adjusts the amplitude of the sine wave. The General Purpose Timer (GPT) is used to generate the PWM signal.

A new project is created in e2studio with the bare metal minimal with Free RTOS template:

Renesas RA C/C++ Project

Device and Tools Selection

Device Selection

FSP Version: 5.5.0
Board: FPB-RA6E1
Device: R7FA6E10F2CFP
Core: CM33
Language: ● C ○ C++

Board Description
Fast Prototyping Board for RA6E1 MCU Group
Visit https://www.renesas.com/ra/fpb-ra6e1 to get kit user's manual, quick start guide, errata, design package, example projects, etc.

Device Details

TrustZone	Yes
Pins	100
Processor	Cortex-M33

Toolchains
GNU ARM Embedded
LLVM Embedded Toolchain for Arm

13.2.1.arm-13-7

Manage Toolchains...

Debugger
J-Link ARM

Renesas RA C/C++ Project

Build Artifact and RTOS Selection

Build Artifact Selection

● **Executable**
  • Project builds to an executable file

○ **Static Library**
  • Project builds to a static library file

○ **Executable Using an RA Static Library**
  • Project builds to an executable file
  • Project uses an existing RA static library project

RTOS Selection

FreeRTOS (v10.6.1+fsp.5.5.0)

Summary

Generate Project Content

Project Summary

RENESAS

Board: FPB-RA6E1
Device: R7FA6E10F2CFP
Core: CM33
Toolchain: GCC ARM Embedded
Toolchain Version: 13.2.1.arm-13-7
FSP Version: 5.5.0
Project Type: Flat
Location: C:/Users/Asus/e2_studio/workspace/Renesas_Lab1

Selected software components

Board Support Package Common Files	v5.5.0
I/O Port	v5.5.0
FreeRTOS	v10.6.1+fsp.5.5.0

Properties

Renesas_Lab1

Property	Value
∨ Info	
derived	false
editable	true
last modified	November 8, 2024, 2:15:31 PM
linked	false
location	C:\Users\Asus\e2_studio\workspace\Rene...
name	Renesas_Lab1
path	/Renesas_Lab1

FSP Visualization   Console

RA FSP
Extracted file: ra/fsp/inc/api/bsp_api.h
Extracted file: ra/fsp/src/r_ioport/r_ioport.c

The pins are properly set:

The timer (GPT peripheral) is configured to output PWM channel 6 on pin P407 with a PWM output of 16kHz.
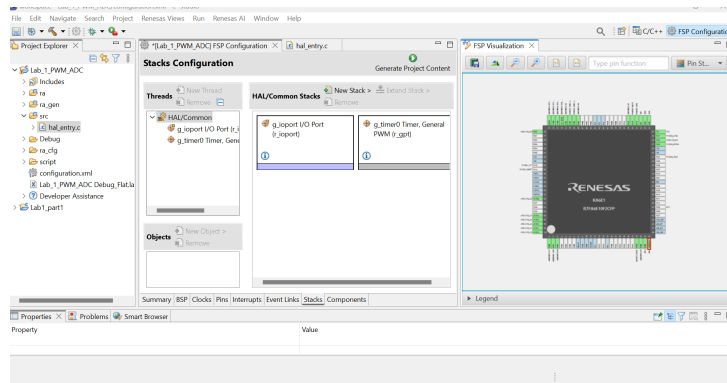
Settings	Property	Value
API Info	∨ Common	
	Parameter Checking	Default (BSP)
	Pin Output Support	Enabled
	Write Protect Enable	Disabled
	∨ Module g_timer0 Timer, General P	
	∨ General	
	> Compare Match	
	Name	g_timer0
	Channel	6
	Mode	Saw-wave PWM
	Period	16
	Period Unit	Kilohertz
	> Output	
	> Input	

∨ Output		
> Custom Waveform		
Duty Cycle Percent (only app	50	
GTIOCA Output Enabled	True	
GTIOCA Stop Level	Pin Level Low	
GTIOCB Output Enabled	False	
GTIOCB Stop Level	Pin Level Low	
> Input		
∨ Interrupts		
Callback	Timer_CB	
Overflow/Crest Interrupt Pri	Priority 3	
Capture/Compare match A	Disabled	
Capture/Compare match B	Disabled	
Underflow/Trough Interrupt	Disabled	
> Extra Features		
∨ Pins		
GTIOC6A	P407	
GTIOC6B	None	

Adding the ADC module (r_adc), setting channel 0 for input, and configuring it to read from pin P001.

Settings	Property	Value
API Info	∨ Common	
	Parameter Checking	Default (BSP)
	∨ Module g_adc0 ADC (r_adc)	
	∨ General	
	Name	g_adc0
	Unit	0
	Resolution	12-Bit
	Alignment	Right
	Clear after read	On
	Mode	Single Scan
	Double-trigger	Disabled
	∨ Input	
	∨ Channel Scan Mask (channe	
	Channel 0	☑
	Channel 1	☑
	Channel 2	☐
	Channel 3	☐
	Channel 4	☐
	Channel 5	☐
	Channel 6	☐
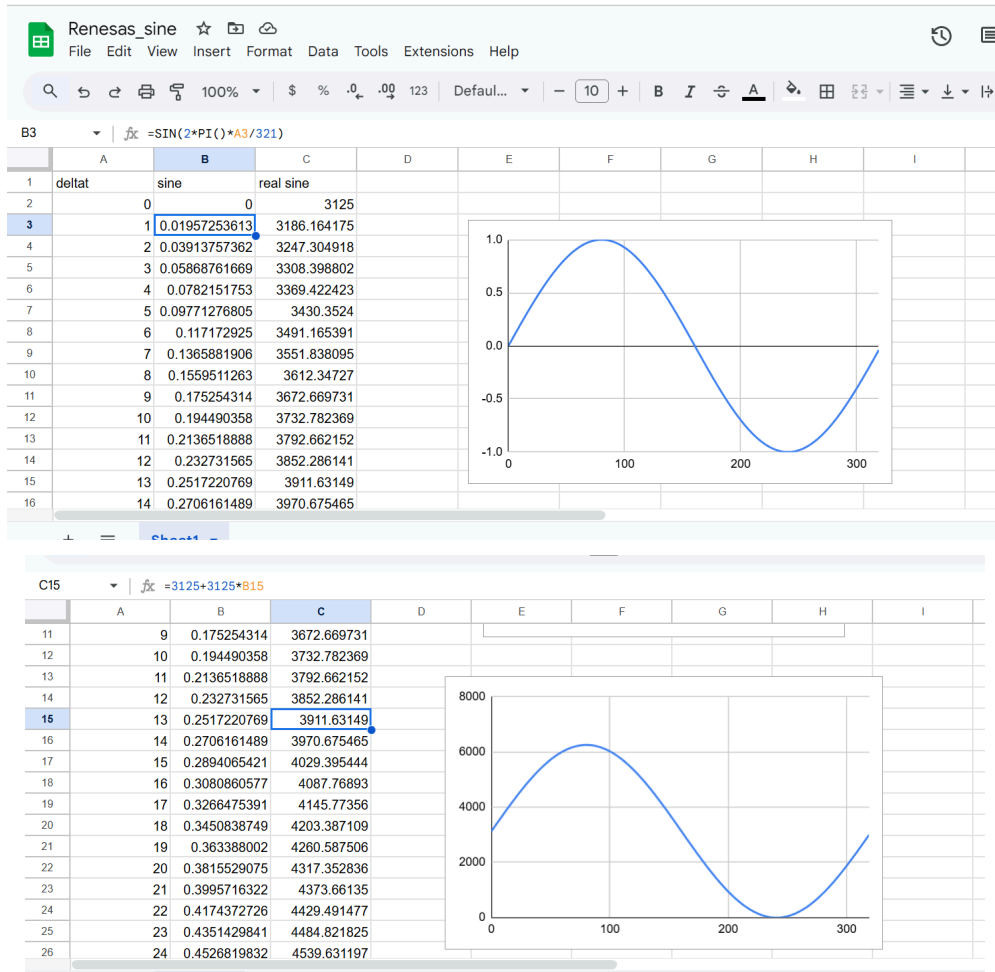
A sine wave is generated in the Excel sheet using the formula, and the real data sine is exported as an array to the e2studio.

Link to the excel sheet: 🟩 Renesas_sine

Sine == SIN(2*PI()*(A2/320))

Exported Data = INT(3125+3125*B2) =((period/2)+(period/2)*sine value)

Renesas_sine

File Edit View Insert Format Data Tools Extensions Help

B3 | fx =SIN(2*PI()*A3/321)

	A	B	C
1	deltat	sine	real sine
2	0	0	3125
3	1	0.01957253613	3186.164175
4	2	0.03913757362	3247.304918
5	3	0.05868761669	3308.398802
6	4	0.0782151753	3369.422423
7	5	0.09771276805	3430.3524
8	6	0.117172925	3491.165391
9	7	0.1365881906	3551.838095
10	8	0.1559511263	3612.34727
11	9	0.175254314	3672.669731
12	10	0.194490358	3732.782369
13	11	0.2136518888	3792.662152
14	12	0.232731565	3852.286141
15	13	0.2517220769	3911.63149
16	14	0.2706161489	3970.675465

C15 | fx =3125+3125*B15

	A	B	C
11	9	0.175254314	3672.669731
12	10	0.194490358	3732.782369
13	11	0.2136518888	3792.662152
14	12	0.232731565	3852.286141
15	13	0.2517220769	3911.63149
16	14	0.2706161489	3970.675465
17	15	0.2894065421	4029.395444
18	16	0.3080860577	4087.76893
19	17	0.3266475391	4145.77356
20	18	0.3450838749	4203.387109
21	19	0.363388002	4260.587506
22	20	0.3815529075	4317.352836
23	21	0.3995716322	4373.66135
24	22	0.4174372726	4429.491477
25	23	0.4351429841	4484.821825
26	24	0.4526819832	4539.631197

Imported .csv data file

Starting both timer and ADC with interrupts enabled. In the timer interrupt service routine (ISR), values from the sine wave array are assigned to the duty cycle in a linear loop.

The code for ADC_Thread_entry.c:


```c
#include <ADC_Thread.h>
#define period 0x989680
#define halfperiod (period/2)
/* New Thread entry function */
int sine[] = {3125, 3186, 3247, 3308, 3369, 3430, 3491, 3551,
;uint16_t ADCDATA,ptr;
uint32_t value;
float gain;

void ADC_Thread_entry(void *pvParameters)
{
    FSP_PARAMETER_NOT_USED (pvParameters);
    R_ADC_Open(&g_adc0_ctrl, &g_adc0_cfg);
    R_ADC_ScanCfg(&g_adc0_ctrl, &g_adc0_channel_cfg);

    R_GPT_Open(&g_timer0_ctrl, &g_timer0_cfg);
    R_GPT_Enable(&g_timer0_ctrl);
    R_GPT_Start(&g_timer0_ctrl);


    /* TODO: add your own code here */
    while (1)
    {
        R_ADC_ScanStart(&g_adc0_ctrl);
        vTaskDelay (1);
    }
}
```



```c
    /* TODO: add your own code here */
    while (1)
    {
        R_ADC_ScanStart(&g_adc0_ctrl);
        vTaskDelay (1);//every one second amplitude is calcula
    }
}
//Callback function
void ADC_CB(adc_callback_args_t *p_args)
{
    /*ADD Code here*/
    if(p_args->event == ADC_EVENT_SCAN_COMPLETE )
    {
        R_ADC_Read(&g_adc0_ctrl, 0, &ADCDATA);
        gain=(float)((ADCDATA*100)/4096);

    }
}
```

FSP Visualization   Console

The active editor element does not use this vi


```c
    }
}
//Callback function
void ADC_CB(adc_callback_args_t *p_args)
{
    /*ADD Code here*/
    if(p_args->event == ADC_EVENT_SCAN_COMPLETE )
    {
        R_ADC_Read(&g_adc0_ctrl, 0, &ADCDATA);
        gain=(float)((ADCDATA*100)/4096);

    }
}

//callback function for timer

void Timer_CB(timer_callback_args_t *p_args)
{
    if(ptr>318)
    {
        ptr=0;
    }
    /*Add code here*/
    if(p_args->event == TIMER_EVENT_CYCLE_END)
    {
        value=(uint32_t)((sine[ptr++])*(gain/100));
        R_GPT_DutyCycleSet(&g_timer0_ctrl, value, GPT_IO_PIN_G
    }
}
}
```

Building the code:

```
26                  }
27          }
28          //Callback function
29  ⊖ void ADC_CB(adc_callback_args_t *p_args)
30          {
31              /*ADD Code here*/
32  ⊖          if(p_args->event == ADC_EVENT_SCAN_COMPLE
33              {
34                  R_ADC_Read(&g_adc0_ctrl, 0, &ADCDATA)
35                  gain=(float)((ADCDATA*100)/4096);
36
37              }
38          }
39
40          //callback function for timer
41
42  ⊖ void Timer_CB(timer_callback_args_t *p_args)
43          {
44  ⊖          if(ptr>318)
45              {
46                  ptr=0;
47              }
48              /*Add code here*/
49  ⊖          if(p_args->event == TIMER_EVENT_CYCLE_END
50              {
51                  value=(uint32_t)((sine[ptr++])*((int)
52                  R_GPT_DutyCycleSet(&g_timer0_ctrl, va
53
54              }
55
56          }
57
58
59
```

Properties
Property	Value
∨ Info	
derived	false
editable	true
last modified	November 8, 2024, 3:19:00 PM
linked	false

CDT Build Console [Renesas_Lab1]
```
Extracting support files...
15:19:13 **** Incremental Build of configuration Debug for project
make -r -j4 all
Building file: ../src/ADC_Thread_entry.c
Building target: Renesas_Lab1.elf
arm-none-eabi-objcopy -O srec "Renesas_Lab1.elf"  "Renesas_Lab1.sre
arm-none-eabi-size --format=berkeley "Renesas_Lab1.elf"
   text    data     bss     dec     hex filename
  14416    1288    4664   20368    4f90 Renesas_Lab1.elf

15:19:14 Build Finished. 0 errors, 0 warnings. (took 1s.232ms)
```

Debugging:

Connecting it to the oscilloscope we get:
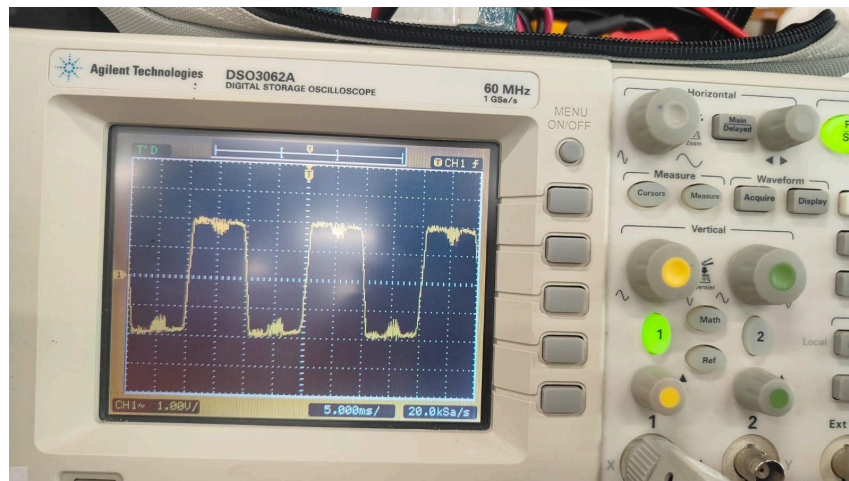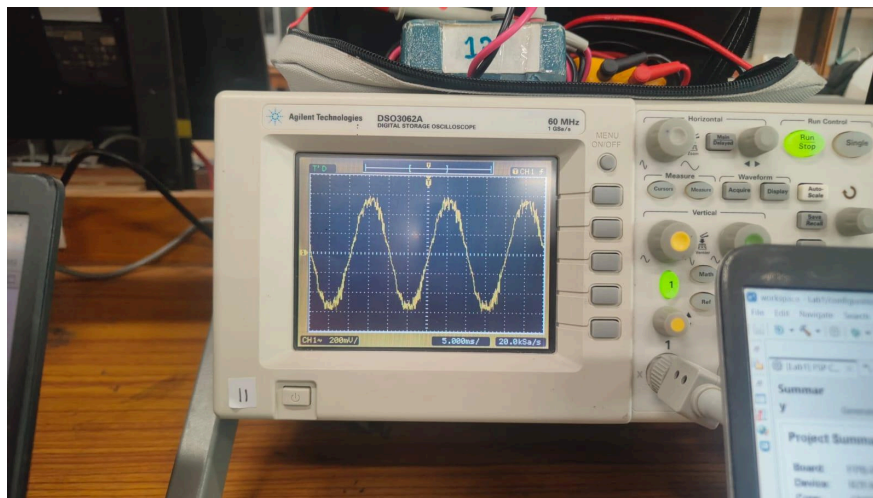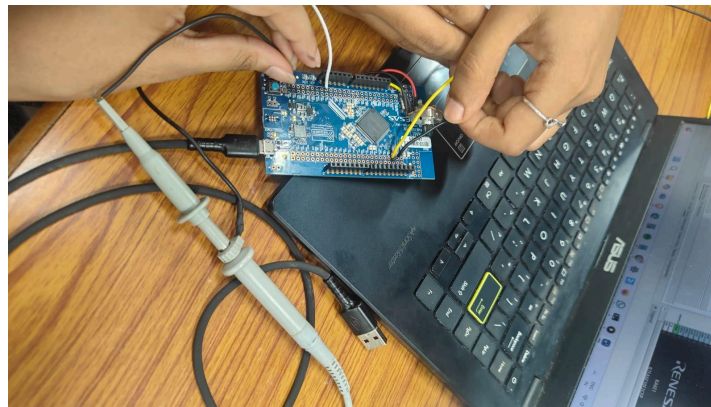
The objective was to generate a Pulse Width Modulated (PWM) signal whose duty cycle is modulated by a sine wave. The amplitude of the sine wave is dynamically controlled using an ADC input. The generated PWM is observed and verified using oscilloscope.

Lab - 2:

UART Loopback Communication:

The objective is to establish a UART loopback communication using the SCI9 interface, incorporating interrupt-driven data transmission and reception. The loopback setup will allow for data sent through the UART transmitter to be received by the UART receiver for testing purposes. Interrupt handling is employed to efficiently manage data flow without polling, ensuring minimal CPU intervention. The communication functionality and data integrity are verified using Segger Real-Time Terminal (RTT), providing a real-time view of transmitted and received data.

Following the similar set-up as Lab-1, UART SCI9 stack is added with SCI channel 0 pin connected.

Then the project content is genertared and Segger_RTT folder is copied to the src folder and the path to the compiler is added.
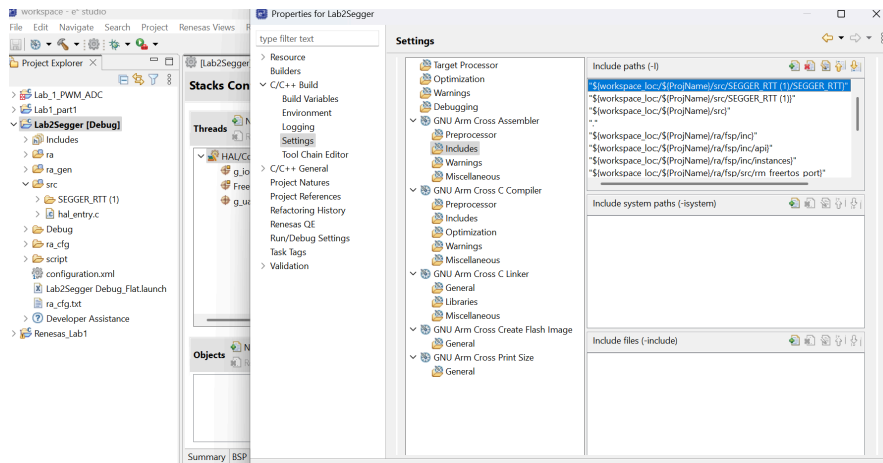
The code in hal_entry.c:


```c
#include "hal_data.h"
#include "SEGGER_RTT/SEGGER_RTT.h"

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
FSP_CPP_FOOTER

fsp_err_t status;
uint8_t write_buff[1];
uint8_t read_buff[1];
volatile uint8_t txflag, rxflag;
unsigned ret;

/* main() is generated by the RA Configuration editor and is used to
   *************************************************************************
void hal_entry(void)
{
    /* TODO: add your own code here */

    SEGGER_RTT_printf (0, "UART_LOOPBACK_attempt1\r\n");

//      __BKPT(0);

    status = R_SCI_UART_Open(&g_uart0_ctrl, &g_uart0_cfg);
    if(status!=FSP_SUCCESS)
    {
        __BKPT(0);
    }

    while(1)
    {
//      __BKPT(0);
        SEGGER_RTT_printf (0,"Enter char\r\n");
        while(SEGGER_RTT_Read(0, read_buff,1)<1);
```



```c
//          SEGGER_RTT_printf (0, "RTT_read_done\r\n");
        R_SCI_UART_Read(&g_uart0_ctrl, write_buff, 1);
//          SEGGER_RTT_printf (0, "UART_read_done\r\n");
        R_SCI_UART_Write(&g_uart0_ctrl, read_buff, 1);
//          SEGGER_RTT_printf (0, "UART_write_done\r\n");
        while(txflag!=1);
        txflag=0;
        while(rxflag!=1);
        rxflag=0;
//          SEGGER_RTT_printf (0, "Something %s\r\n", write_buff);
        SEGGER_RTT_printf (0, "\"%s\"\r\n", write_buff);
    }

#if BSP_TZ_SECURE_BUILD
    /* Enter non-secure code */
    R_BSP_NonSecureEnter();
#endif
}

/* Callback function */
void uart_cb0(uart_callback_args_t *p_args)
{
    /* TODO: add your own code here */
    if(p_args->event == UART_EVENT_TX_COMPLETE)
    {
        txflag = 1;
    }
    if(p_args->event == UART_EVENT_RX_COMPLETE)
    {
        rxflag = 1;
    }
}
```
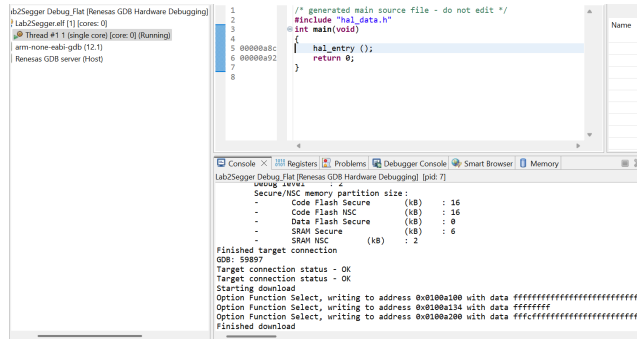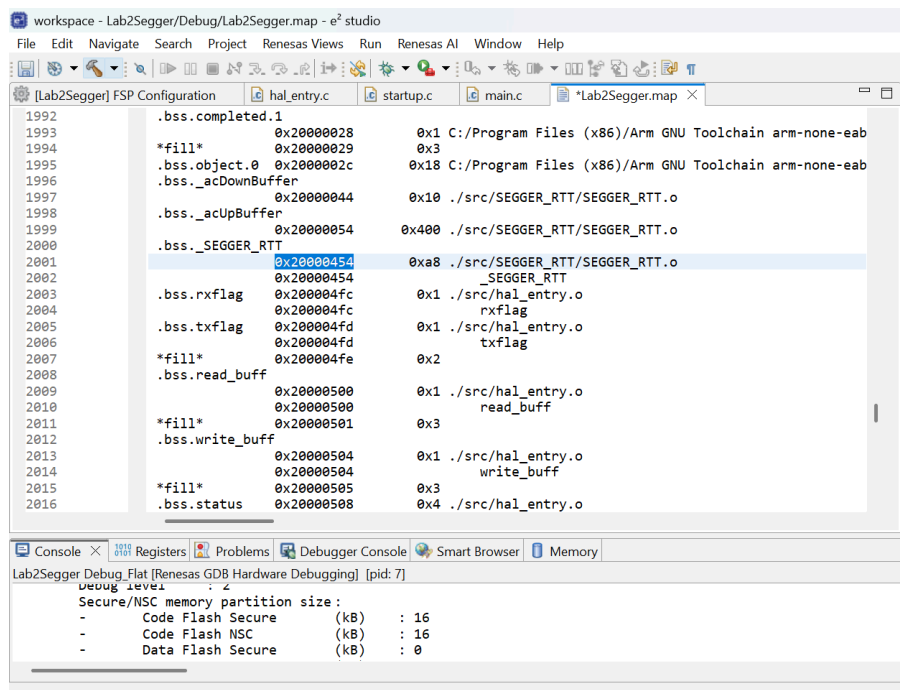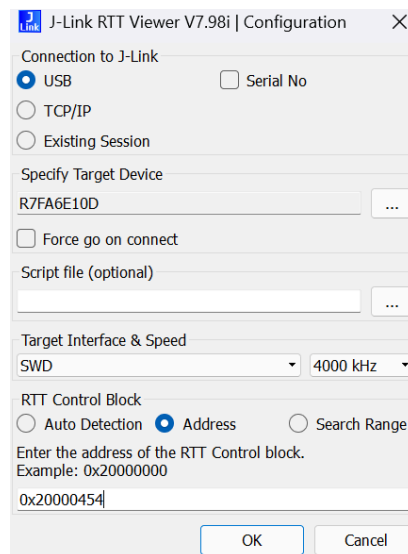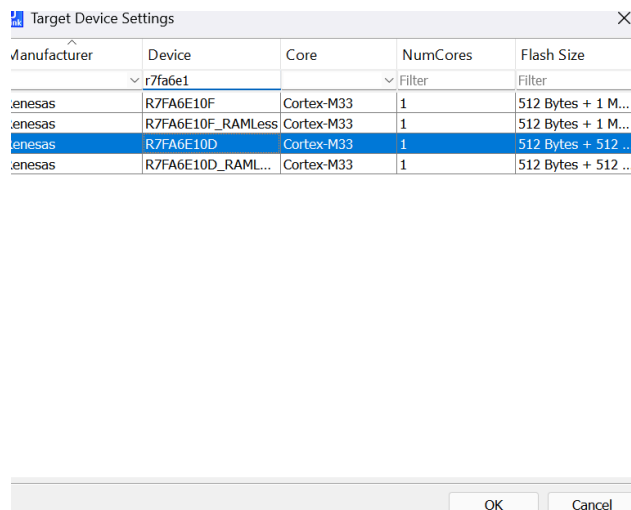
Build:

Debug:

Address for JLink:

JLink:

Terminal Output: