LAB-3 REPORT

Subject: **Embedded Hardware Design**

Subject Code: **EL203**

Members: 1) Shantanu Tyagi **(201801015)**
2) Nikhil Mehta **(201801030)**
3) Sudhanshu Mishra **(201801114)**
4) Bhavana Kolli **(201701082)**

Environment Setup

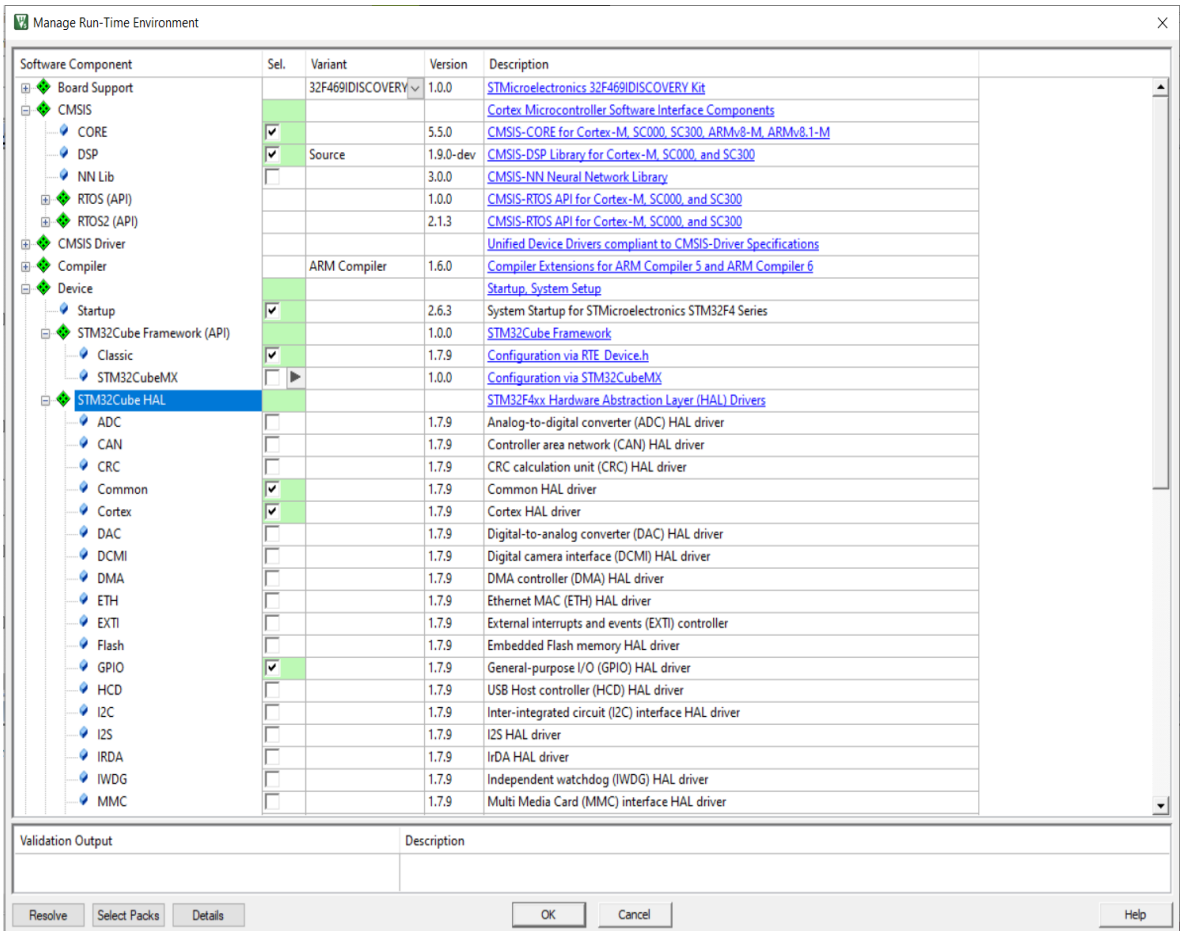
In the IDE, install the Keil::STM32F4xx DFP Package from Pack Manager.

1) Create a new Project and rename it using the appropriate nomenclature.

    1.1.1.1. Set STM32F407VGTx as the target device under the device header

    1.1.1.2. Modify the options for the target device:

2.



2.1. Create a new group under the project

2.2. Make modifications for the following additional options for the target device:

2.2.1. Set ARM compiler version 6 under the Target -> Code Generation header

2.2.2. Select the Simulator radio button and load the KEIL_STM.ini file as an initialization file under the Debug header

2.3. Add the main.c and sinewave.c file under the created group in the project with the relevant header file (#include "stm32f4xx_hal.h",#include "arm_math.h")

Running the Simulation

1 To run the code, first we have to compile the main.c file. This is done by clicking on the build button in Keil IDE.
2 Once the build is complete, we are prompted with errors and we need to fix them.
3 After fixing the bugs we move to the debug section and run the code to get the results.
4 To see the results, we can either see the result value or check out the graph for the simulation.
5 We can see the values of the variables directly by clicking on the variable and then on ‘add to watch’.
6 In order to get the graph, we select the variable and click on ‘analyze’.
7 After checking the output, we will have to close the debug session to make further changes in the code. Then follow the steps mentioned above again to see the new output.

CODE:

```
% MATLAB
clear all;

freq = 5;
Amp = 2;
ts = 0.001;
T = 5;

t=0:ts:T;
y = Amp*sin(2*pi*freq*t);
plot(t,y)
csvwrite("_5Hz_sine_wave.txt",y)
```

```
// main.c
#include "stm32f4xx_hal.h"              // Keil::Device:STM32Cube
HAL:Common
#include "arm_math.h"               // ARM::CMSIS:DSP

#define SIG_LENGTH 5001

uint32_t freq;
extern float32_t _10Hz_sine_wave[SIG_LENGTH];
//void SysTick_Handler(void);
void plot_input_sine_wave(void);
void plot_output_sin_wave(void);
```

```
void plot_both_signal(void);

float32_t inputSample, outputSample;
float32_t outputSignal_f32[SIG_LENGTH];

int main()
{
	int i;
	HAL_Init();
	freq = HAL_RCC_GetHCLKFreq();

	for(i = 0; i < 5001 ;i++)
	{
		outputSignal_f32[i]=_10Hz_sine_wave[i];
	}

	plot_both_signal();

	while(1)
	{

	}
}
void plot_input_sine_wave(void)
{
	int i,j;
	for(i = 0;i<SIG_LENGTH;i++)
		{
			inputSample = _10Hz_sine_wave[i];
			HAL_Delay(1);
				if(i==SIG_LENGTH-1)i=0;
		}
}

void plot_output_sin_wave(void)
{
	int i,j;
	for(i = 0;i<SIG_LENGTH;i++)
		{
			outputSample = outputSignal_f32[i];
			HAL_Delay(1);
				if(i==SIG_LENGTH-1)i=0;
		}
}

void plot_both_signal(void)
```

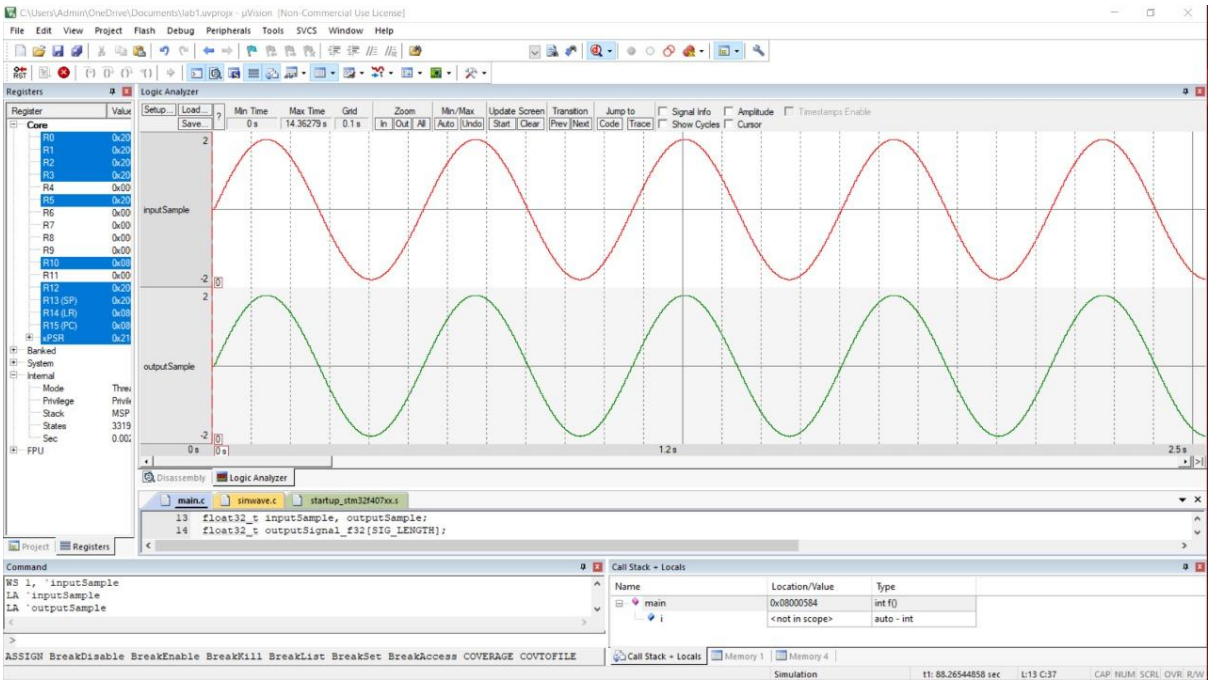
```
{
	int i,j;
	for(i = 0 ;i < SIG_LENGTH; i++)
		{
			inputSample = _10Hz_sine_wave[i];
			outputSample = outputSignal_f32[i];
				HAL_Delay(1);
				if(i==SIG_LENGTH-1)i=0;
		}
}

void SysTick_Handler(void)
{
	HAL_IncTick();
	HAL_SYSTICK_IRQHandler();
}
```

```
// sinewave.c
#include "arm_math.h"                    // ARM::CMSIS:DSP

float32_t _10Hz_sine_wave[5001] =
{
   // enter 501 values of sine wave generated from matlab.
}
```

SIG_LENGTH=5001



Steps(ModelSim)

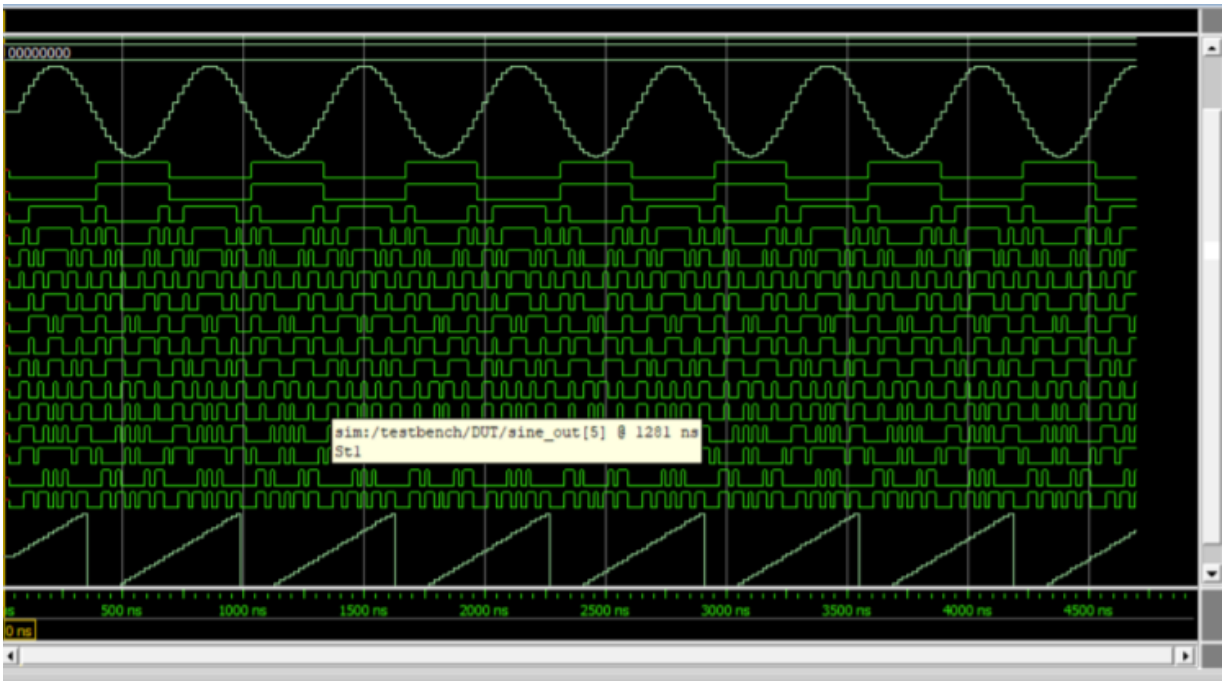
Environment Setup

1) We download the ModelSim - Intel FPGA Edition Software under Quartus Prime Pro Edition from the website: https://fpgasoftware.intel.com/?product=modelsim_ae#tabs-2
2) We run the installer after downloading and start by creating a new project.

Running and simulating code

1) DDS.v, sine tb.v, and sync room.v should be included in the project.
2) Compile and run the above files.
3) To simulate the code, choose the ‘testbench' module from the Simulate Tab, then Start Simulation.
4) Add a wave and then right click on the ‘DUT’ module.
5) Select ‘Add Wave’ and then click on the Run button .
6) Expand sine out and accumulator objects in the context of this project.

Sine wave generation in ModelSim



Conclusion:

1. We can implement continuous sine wave at the microcontroller level using Keil uVision, but MATLAB only simulates it at the software level.
2. In Keil uVision, we may transform a finite sine wave to a continuous sine wave by significantly altering the loop structure.
3. The noise in the waveform measured in Keil reduces as the size of the base array containing the sine wave simulated values increases, and vice versa.
4. This is because the base array size is raised by using smaller step counts in the MATLAB sine wave generator, which results in improved precision.
5. Individual sine wave components may also be viewed in Model Sim in the reference image above. The ultimate necessary sine wave is the sum of all such components.
6. ModelSim allows us to create a sine wave circuit at the circuit level. When compared to MATLAB, it is rather loud. This is due to the fact that ModelSim represents a precise hardware implementation.