

# **LAB-8 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:



Software Component	Sel.	Variant	Version	Description
Board Support		32F469IDISCOVERY	1.0.0	<a href="#">STMicroelectronics 32F469IDISCOVERY Kit</a>
CMSIS				<a href="#">Cortex Microcontroller Software Interface Components</a>
CORE	<input checked="" type="checkbox"/>		5.5.0	<a href="#">CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M, ARMv8.1-M</a>
DSP	<input checked="" type="checkbox"/>	Source	1.9.0-dev	<a href="#">CMSIS-DSP Library for Cortex-M, SC000, and SC300</a>
NN Lib	<input type="checkbox"/>		3.0.0	<a href="#">CMSIS-NN Neural Network Library</a>
RTOS (API)			1.0.0	<a href="#">CMSIS-RTOS API for Cortex-M, SC000, and SC300</a>
RTOS2 (API)			2.1.3	<a href="#">CMSIS-RTOS API for Cortex-M, SC000, and SC300</a>
CMSIS Driver				<a href="#">Unified Device Drivers compliant to CMSIS-Driver Specifications</a>
Compiler		ARM Compiler	1.6.0	<a href="#">Compiler Extensions for ARM Compiler 5 and ARM Compiler 6</a>
Device				<a href="#">Startup, System Setup</a>
Startup	<input checked="" type="checkbox"/>		2.6.3	<a href="#">System Startup for STMicroelectronics STM32F4 Series</a>
STM32Cube Framework (API)			1.0.0	<a href="#">STM32Cube Framework</a>
STM32Cube HAL				<a href="#">STM32F4xx Hardware Abstraction Layer (HAL) Drivers</a>
STM32Cube LL				
Graphics Display				Display Interface including configuration for emWIN

1.2. Create a new group under the project

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

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

1.3.2. Select the Simulator radio button and load the KEIL\_STM.ini file as an initialization file under the Debug header

1.4. 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:

```
#include "stm32f4xx.h"           //device header
#include "stdio.h"
void USART2_Init(void);

int main(void)
{
    int n;
    char str[100];

    USART2_Init();
    printf("Hello from my side\r\n");
    fprintf(stdout, "test for stdout\r\n");
    fprintf(stderr, "test for stderr\r\n");

    while(1)
    {
        printf("How old are you ?");
        scanf("%d", &n);
        printf("Your age is: %d\r\n", n);
        printf("Enter your first name:");
        gets(str);
        printf("Enter your last name:");
        gets(str);
        printf("\r\n");
    }
}

//(1) Initialize USART2
```

```

void USART2_Init(void){
    RCC->AHB1ENR      |= 1;
    RCC->APB1ENR       |= 0x20000;
    GPIOA->AFR[0]      |=0x7700;
    GPIOA->MODER        |=0x00A0;
    USART2->BRR         =0x0683;
    USART2->CR1         |=0x000C;
    USART2->CR1         |=0x2000;
}

//(2) Write a character to USART2
int USART2_write(int ch){
    while(!(USART2->SR & 0x0080)){
        USART2->DR = (ch & 0xFF);
        return ch;
    }

    //(3) read a character to USART2
    int USART2_read(void){
        while(!(USART2->SR & 0x0020)){
            return USART2->DR ;
        }

        //(4) Interface C to I/O library
        struct __FILE{int handle;};
        FILE    __stdin   = {0};
        FILE    __stdout  = {1};
        FILE    __stderr  = {2};

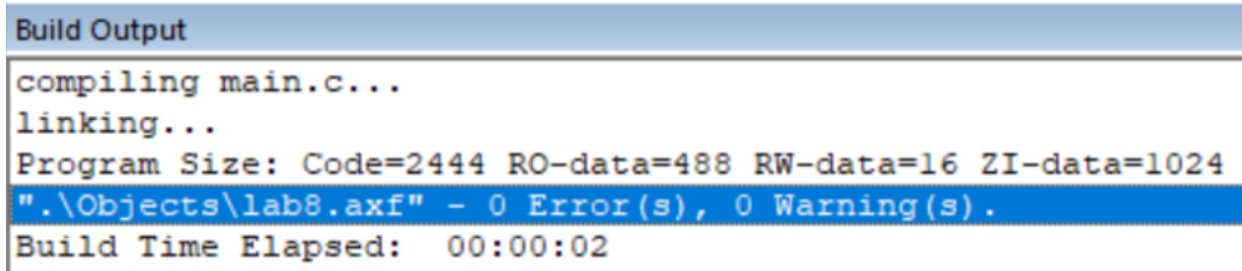
        int fgetc(FILE *f){
            int c;
            c = USART2_read();
            if (c=='\r'){
                USART2_write(c)
                c='\n';
            }
            USART2_write(c);
            return c;
        }

        int fputc(int c, FILE *f)
        {

```

```
    return USART2_write(c);  
}
```

## Screenshot:



Build Output

```
compiling main.c...  
linking...  
Program Size: Code=2444 RO-data=488 RW-data=16 ZI-data=1024  
".\Objects\lab8.axf" - 0 Error(s), 0 Warning(s).  
Build Time Elapsed: 00:00:02
```

## Conclusions/Observations:

1. A Universal Asynchronous Receiver Transmitter (UART) is an individual integrated circuit that is used for serial communications over a computer or peripheral device serial port. We have used a pin CP2102 USB to UART bridge controller for communication. We connect CP2102 with our microcontroller STM32F407VGTx, and the CP2102 with the computer in order to upload the code from Keil into the microcontroller.