BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 416 – Microprocessor and Embedded Systems Laboratory
January 2023

Experiment 01

Familiarization with Microprocessor and Embedded Systems Tools and Software

---

Evaluation Form:

IMPORTANT! You must complete this experiment during your scheduled lab period. All work for this experiment must be demonstrated to and verified by your lab instructor *before the end* of your scheduled lab period.

STEP	DESCRIPTION	MAX	SCORE
1	Install Keil MDK v5	10	
2	Create a Project in Keil MDK v5	10	
3	Install Keil MDK v4	15	
4	Create a Project in Keil MDK v4	20	
5	Complete Lab Exercise 6.2 (Section 6)	30	
6	Answer all questions	15	
	TOTAL	100	

Signature of Evaluator: _____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. *You will not receive credit for this lab experiment unless this statement is signed in the presence of your lab instructor.*

"In signing this statement, I hereby certify that the work on this experiment is my own and that I have not copied the work of any other student (past or present) while completing this experiment. I understand that if I fail to honor this agreement, I will receive a score of ZERO for this experiment and be subject to possible disciplinary action."

Name: _____ **Section:**_____ **Lab Group:** ____ **Date:** _____

E-mail: _____@eee.buet.ac.bd **Signature:** _____

Lab Group Member Student ID: _____

Contents

1 Introduction

Lab Policies

- Please refer to the front matter of your lab manual for lab policies.
  Will be discussed in lab
- Please form a group of 4 members for performing lab. This group will also be same for your project groups.

2 Hardware and Software Requirements

2.1 Software requirements

The first part of the lab is for Computer Architecture. The following table is a list of required software tools that you need for the labs.

Note:

- The software versions listed in the table are versions that have been verified working with the labs. **Therefore, we recommend that you use the versions listed below.** You can use the latest available (and most stable) versions of the software if backward and forward compatibility are supported by the versions.
- The instructions in the lab manuals are based on a Windows OS. However, most of the software tools in this lab have a Linux version too. Therefore, you may have to modify the lab instructions and/or environment settings on your own if you are running this lab on a Linux OS.

2.2 Hardware requirements

We will use an STM32 Nucleo board for the experiments. For the lab we use a STM32 Nucleo Board, with STM32F446RE processor. The STM32 Nucleo development boards allow developers to try out new ideas and quickly create prototypes with any STM32 MCU. Thanks to their connectors, STM32 Nucleo boards can extend their capabilities using multiple application-related hardware add-ons. The board used in the lab, Nucleo-64 boards feature Arduino Uno rev3 and ST morpho connectors. STM32 Nucleo boards integrate an ST-LINK debugger/programmer, eliminating the need for a separate probe. The development boards come with a comprehensive STM32 software HAL library and many software examples, and seamlessly work with a wide range of development environments, including IAR EWARM, Keil MDK-ARM and GCC/LLVM-based IDEs.

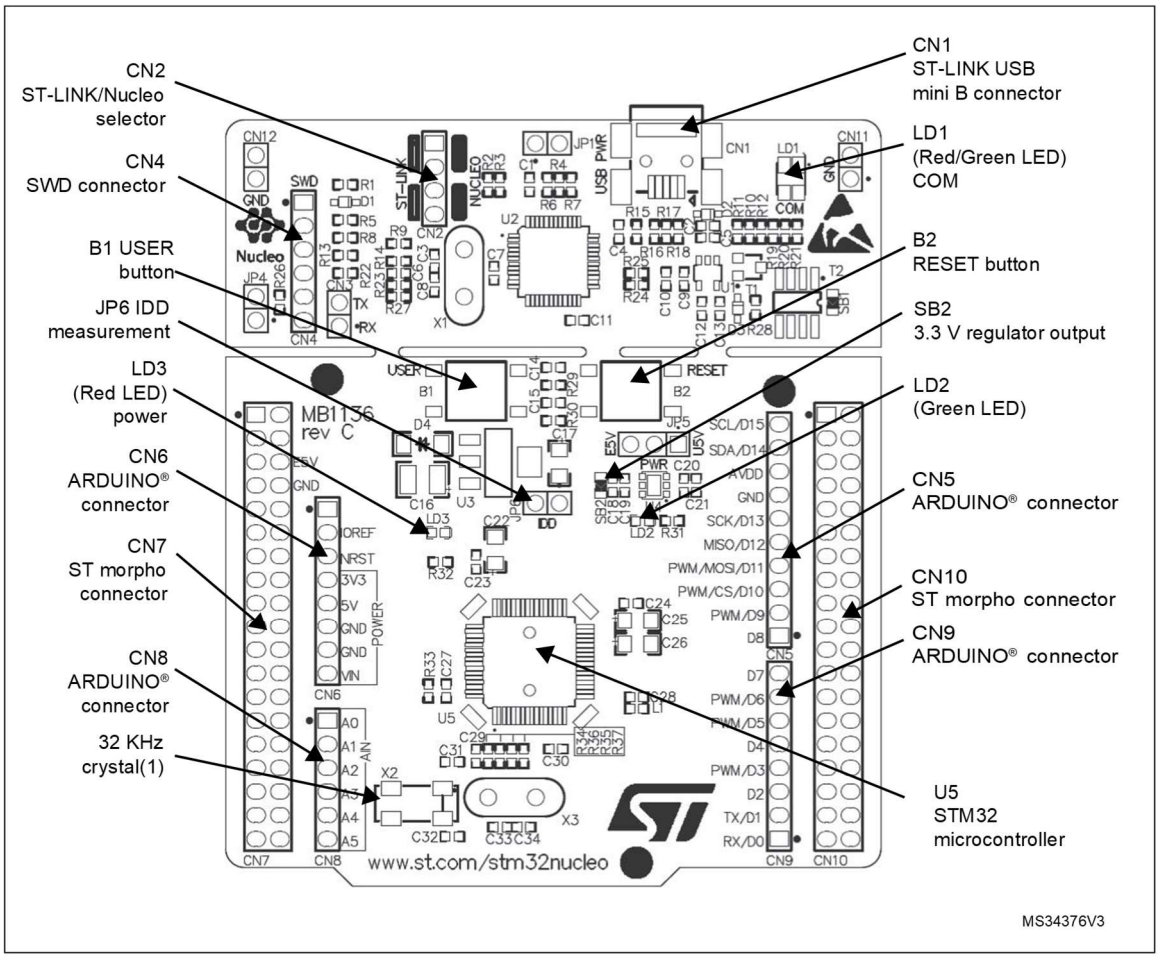


Figure 1: Component Layout of STM32-Nucleo Board Used in lab

Software	**Usage**	**Website**	**Version**
Operating System	All labs	–	Windows 11
Keil uVision 5	Exp 1-4	https://www.keil.arm.com/mdk-community/	5
Keil uVision 4	Exp 5-6		4
Model Sim – Intel FPGA Starter Edition	Exp 5-6		**10.5b**

Table 1: Software tools and versions

3 Software: Keil MDK v5

3.1 Install Keil MDK v5

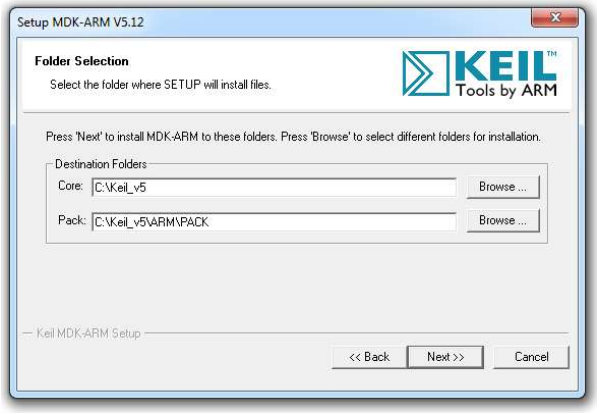
Warning: Do not connect the Discovery Kit into your PC or laptop before the software installation completes. If you connect your kit to PC before installing the USB driver, Windows OS often mistakenly associates a wrong USB driver to the kit. Thus, you will not be able to program the kit. The solution is to go to the control panel and change the USB driver to ST-Link USB driver.

Step 1: Install Keil MDK-ARM

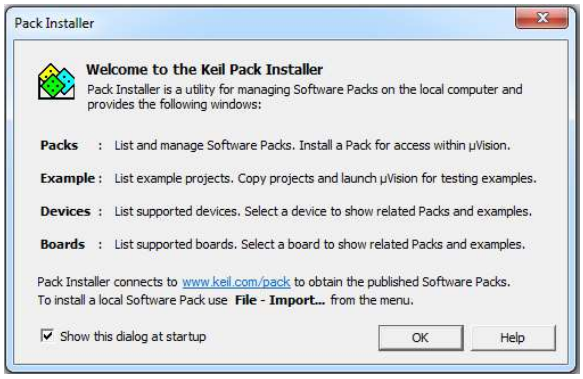
1. Download the latest free evaluation version Keil MDK-ARM from the following link:

   https://www.keil.com/demo/eval/arm.htm

   - Keil MDK-ARM contains µVision 5 IDE (Integrated Development Environment) with debugger, flash programmer and the ARM compiler toolchain.
   - The major limitation of the free version is that programs that generate more than 32 Kbytes of code and data will not compile, assemble, or link.
2. Run the downloaded ***MDK5xx.exe*** and install to the default path. The software takes 2GB disk storage space. You can install it to a different driver, instead of the default C drive, if there is limited space in C drive.

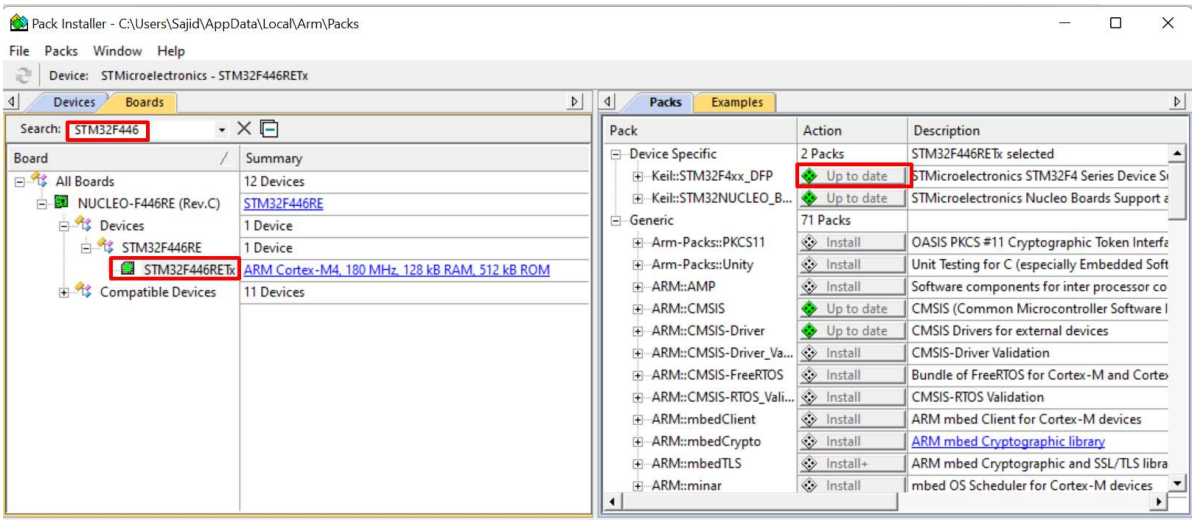


3. After the core software is installed, a dialog will show up to install Keil Pack. It automatically downloads selected components (called packs) from http://www.keil.com/dd2/pack/
4. Click OK and then the following window shows up.



5. If you use the Discovery kit with STM32L4 MCU, please select the device **STM32F4 Series** on the right and all its available components will be shown on the left. Then, install or update the following software components:

- **ARM::CMSIS**
- **Keil::MDK-Middleware**
- **Keil::STM32F4xx_DFP**



Step 2: Install ST-Link USB Driver

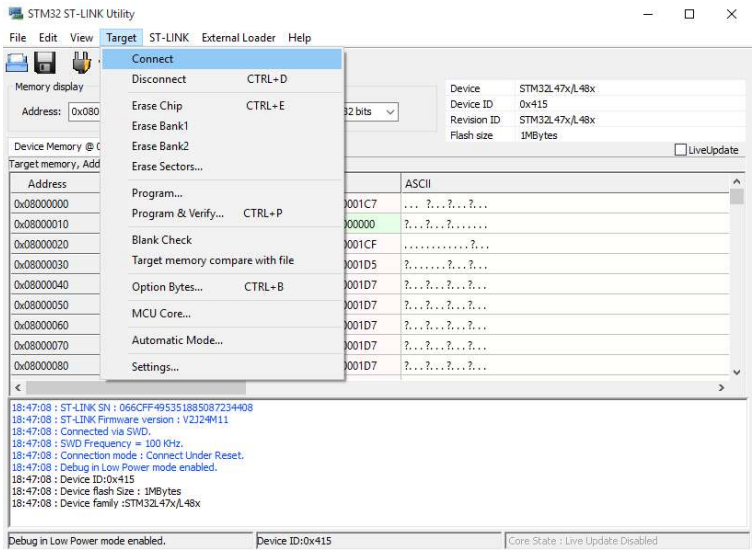
- Do not connect the discovery kit before you install the USB driver for ST-Link.
- Go to the directory **C:\Keil_v5\ARM\STLink\USBDriver** and run **stlink_winusb_install.bat** in administrator mode (Right click and select run as a administor).
- Now you can connect the discovery kit to computer via a "Type A to mini-B" USB cable. The discovery kit should be correctly recognized as "STMicroelectronics STLink dongle."

Step 3: Install STM32 ST-Link Utility

You can download the installation software from the following link:

http://www.st.com/web/en/catalog/tools/PF258168

Typically, we use Keil to program the discovery kit. However, the ST-Link utility is helpful to erase the flash memory if you mistakenly program the debug/program pins of the STM32L processor. See the YouTube tutorial for more details: https://youtu.be/OiwwB0AvIBI



3.2 Creating a Base Project in Keil MDK v5

This tutorial takes the follwing the kits as an example of creating a project in Keil IDE for assembly programs.

- Discovery kit with STM32F446RE MCU (Cortex-M4)
- Discovery kit with STM32L476VG MCU (Cortex-M4 with FPU and DSP)
- Nucleo-L476RG kit with STM32L476RG MCU (Cortex-M4 with FPU and DSP)

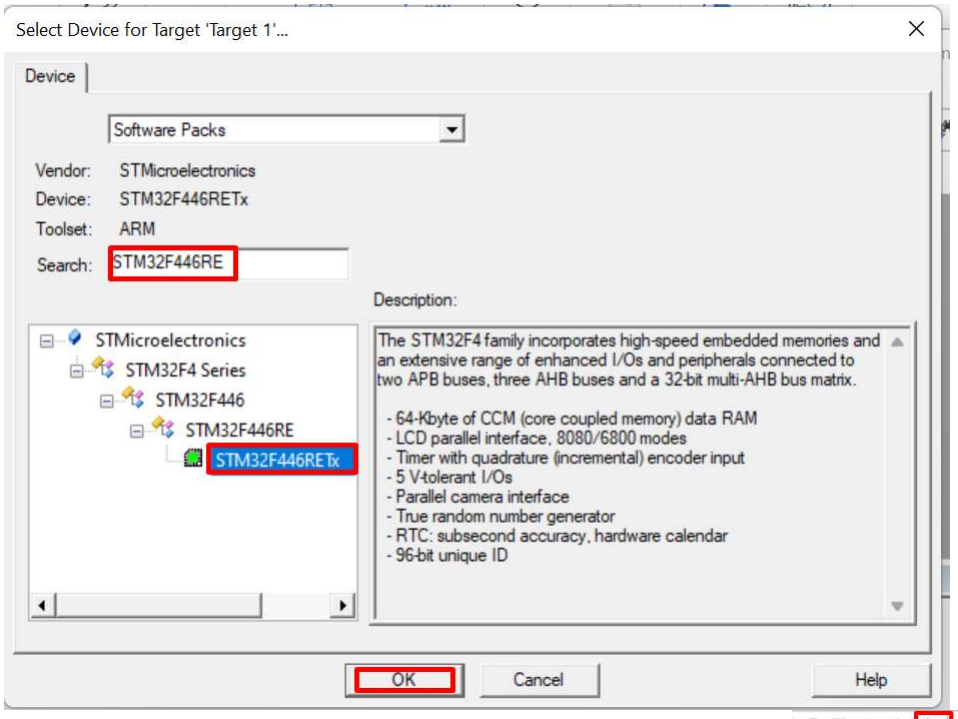
Note that the project does not use the default startup files provided by Keil. You need to download a modified version of ***startup_stm32l1xx_md.s*** or ***startup_stm32l476xx.s*** from the book website: http://web.eece.maine.edu/~zhu/book/lab.php.

Identifying Target Processor

Kit	Processor	Core	Flash	RAM
Nucleo-F4RE	STM32F445RE	Cortex-M4 (DSP + FPU)	512 KB	128 KB
STM32L4 Discovery Kit	STM32L476VG	Cortex-M4 (DSP + FPU)	1 MB	128 KB
Nucleo-L476RG	STM32L476RG	Cortex-M4 (DSP + FPU)	1 MB	128 KB

Steps to create a new project in Keil

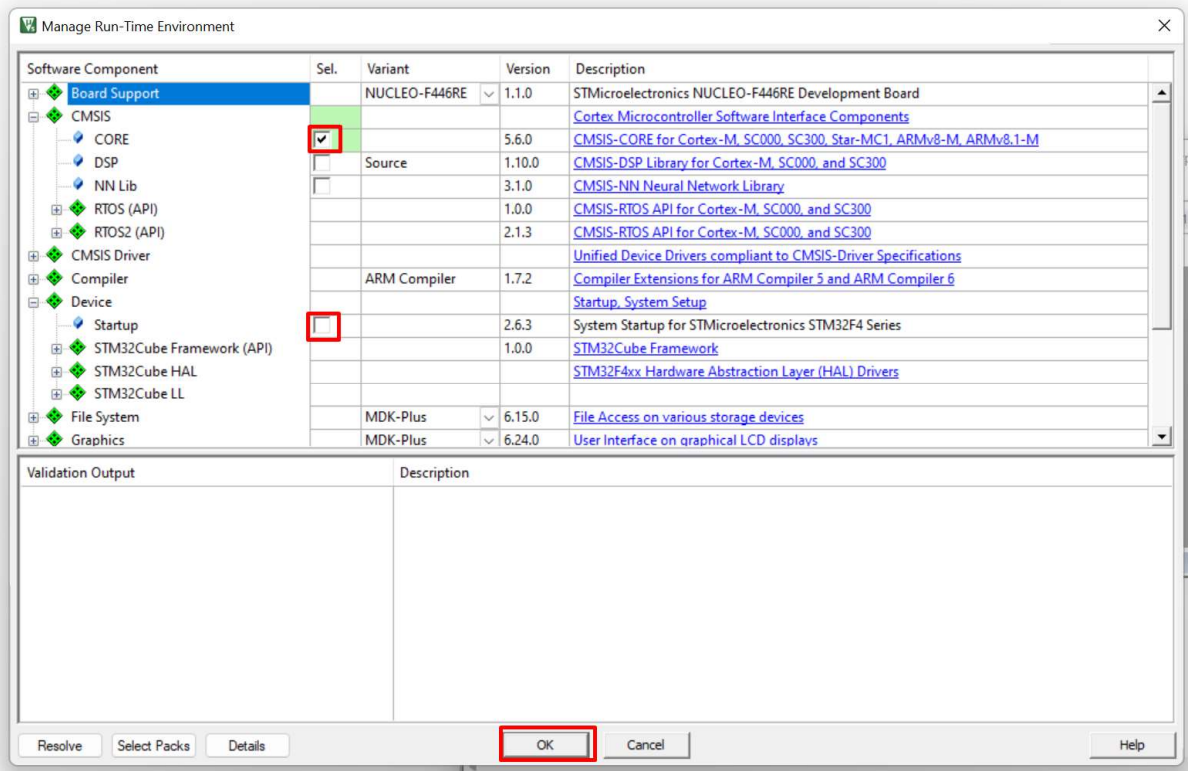
1. From the menu **Project → New µVision Project**
2. Give the project a name and select its storage directory. In this tutorial, the project is named as "lab".
3. Select the target device
   - If you use Nucleo-F446RE, select **STM32F4 Series**, and then select **STM32F446RE** or STM32L152RB as the CPU.
   - If you use STM32L4 Discovery Kit, select **STM32L4 Series**, and then select **STM32L476VGTx**.
   - If you use the Nucleo-L476RG, select the device **STM32L4 Series**, and then select **STM32L476RGTx**.



If did not see the targeted processor in the list, click the "**Pack Installer**" () button in the tool bar and install the component **Keil::STM32F4xx_DFP** or **Keil::STM32L4xx_DFP**.

(Please refer to page 3 of **Error! Reference source not found.** for details on Package Installer)

4. Select **CMSIS Core** only. Do NOT select "Device Startup". Instead, you should use the one provided by the course website.



5. ~~If you are **Nucleo-STM32F446RE**, add the following source code files into the project. Right click the "**Source Group**" and select "Add Existing Files to Group." You can download the following source codes from the textbook website and adds into the **project if you are creating an assembly-based project**.~~

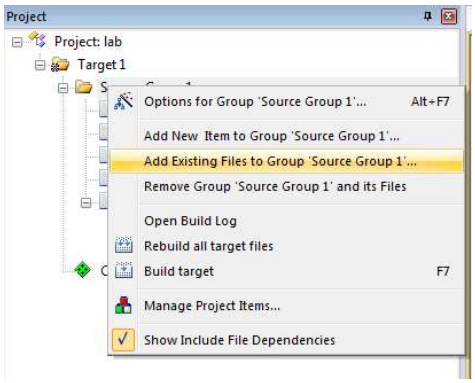
   - ~~**startup_stm32f4xx_md**.s~~
   - ~~**core_cm4_constant**.s~~
   - ~~**stm32f4xx_constants**.s~~
   - ~~**stm32f4xx_tim_constants**.s~~
   - ~~**main**.s~~

   ~~If you use **STM32L4 discovery kit or Nucleo-L476RG**, add the following source code files into the **assembly-based project**.~~

   - ~~**core_cm4_constants.s**~~
   - ~~**stm32l476xx_constants.s**~~
   - ~~**startup_stm32l476xx.s**~~
   - ~~**main.s**~~

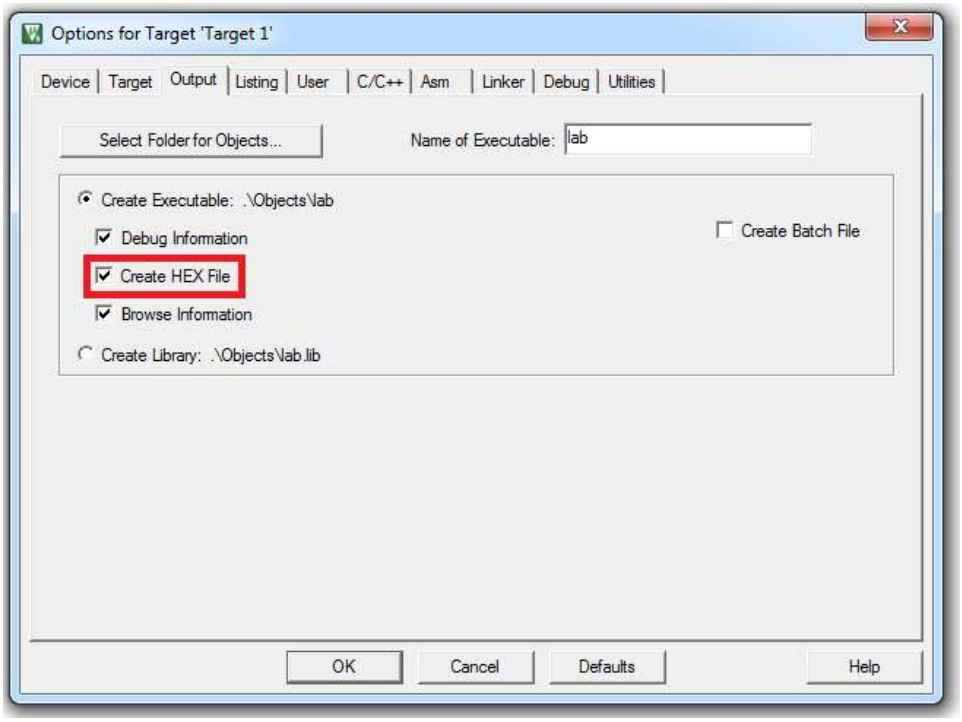
If you are creating a C project, then you should include the following:

- **startup_stm32l467xx**.s or **startup_stm32f446xx.s**
- **main.c**.

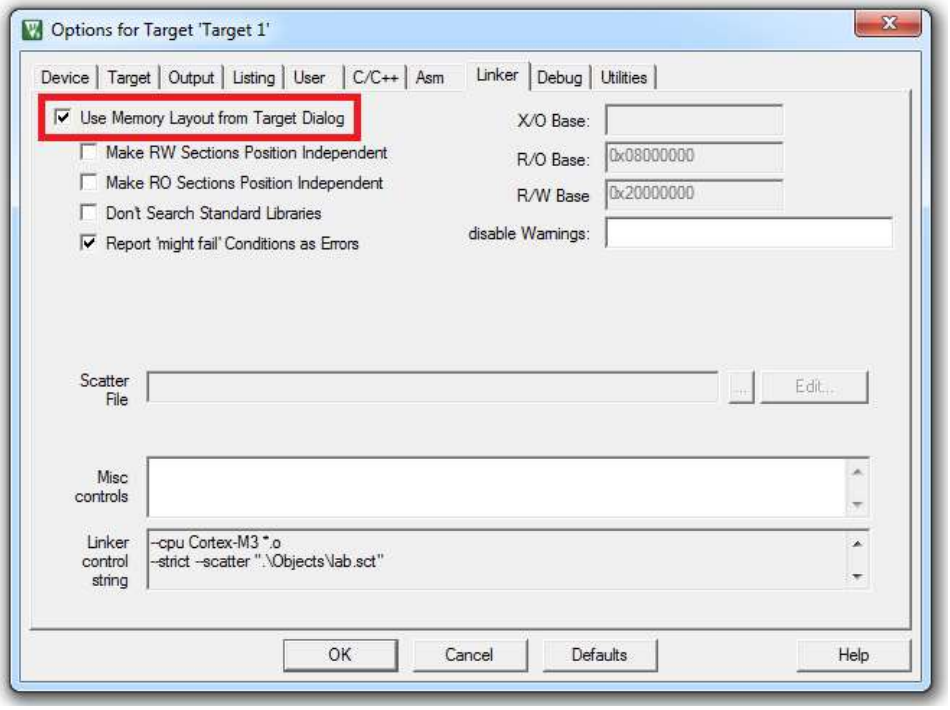


6. Set Project Properties

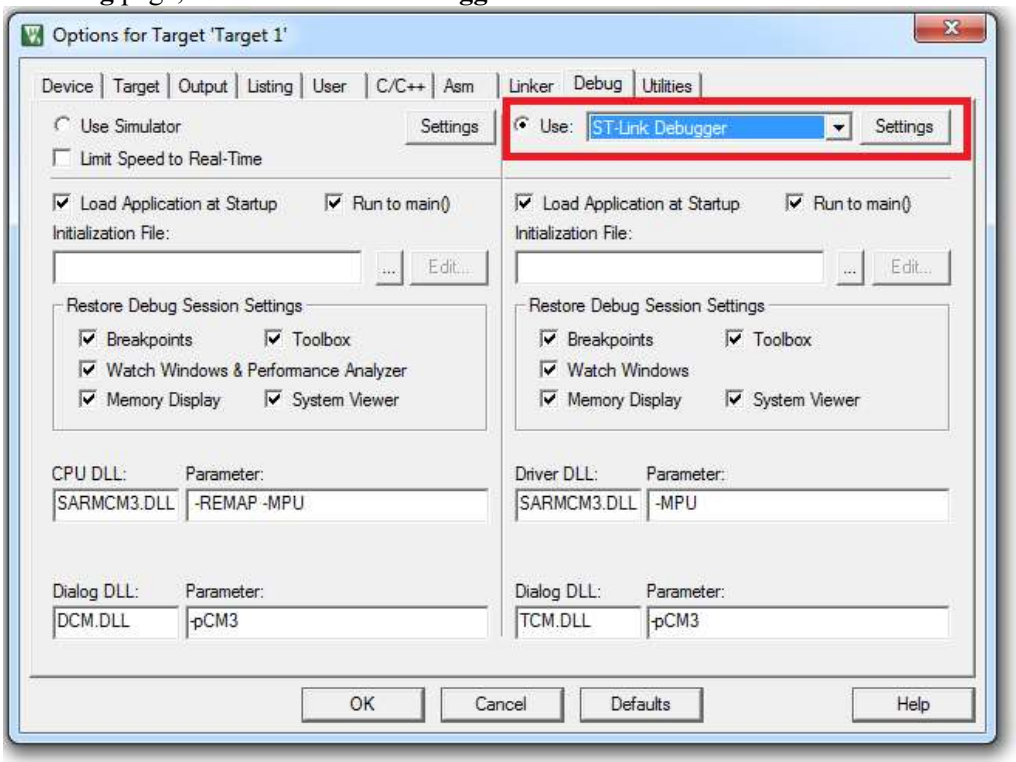
From the menu, click **Project → Option for Target**, Go to the **Output** page, select "**Create HEX file**"



Go to the **Linker** page, select "**Use Memory Layout from Target Dialog**"

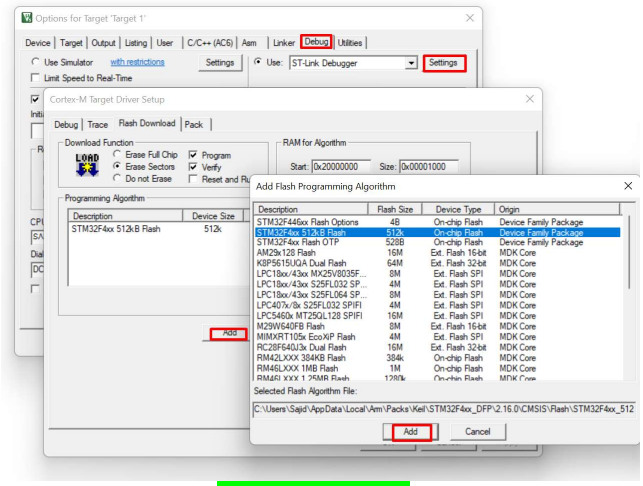


Go to the **Debug** page, select "**ST-Link Debugger"**

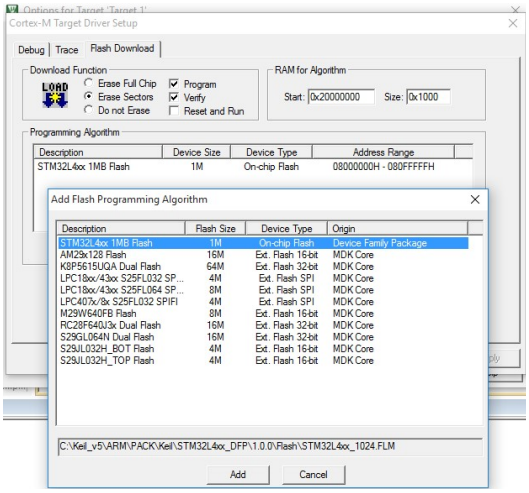


Click "**Settings**" and select "**SW**" (Serial Wire) as the port.

Go to the Flash Download page, and verity that STM32F4xx On-chip Flash is selected in the Programming Algorithm. If not, click "Add" and select STM32F4xx On-chip flash in the popped dialog.



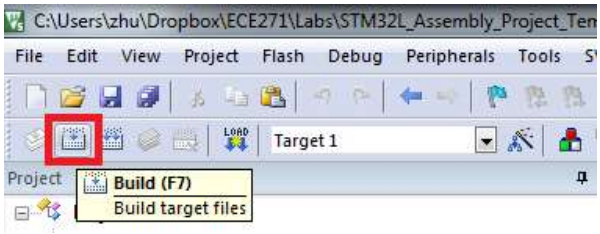
Nucleo-F446RE



STM32L4 Discovery Kit or NUCLEO-L476RG

7. Compile and run your project

Build the program:



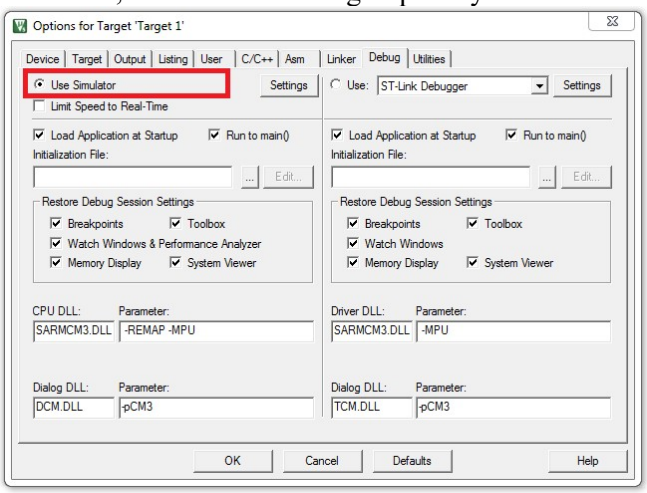
You can ignore the following warning when the linking stage:
.\Objects\lab.sct(8): warning: **L6314W: No section matches pattern *(InRoot$$Sections)**.

Connect your discovery kit to the computer and download the program to the STM32L processor.

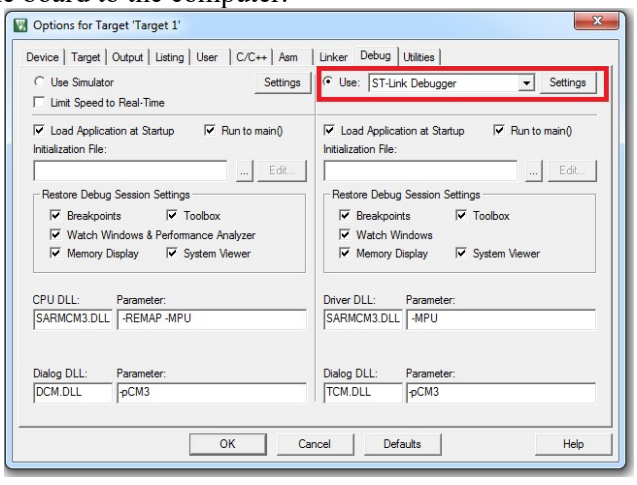
3.3 Keil Debugger Tutorial

Software *vs* Hardware Debug

There are two methods to debug your program: software debug and hardware debug. By using the software debug, you do not have to have the hardware board to debug a software program. However, the hardware debug requires you to connect the board to the computer.



Selecting **software** debug



Selecting **hardware** debug

Debug Control

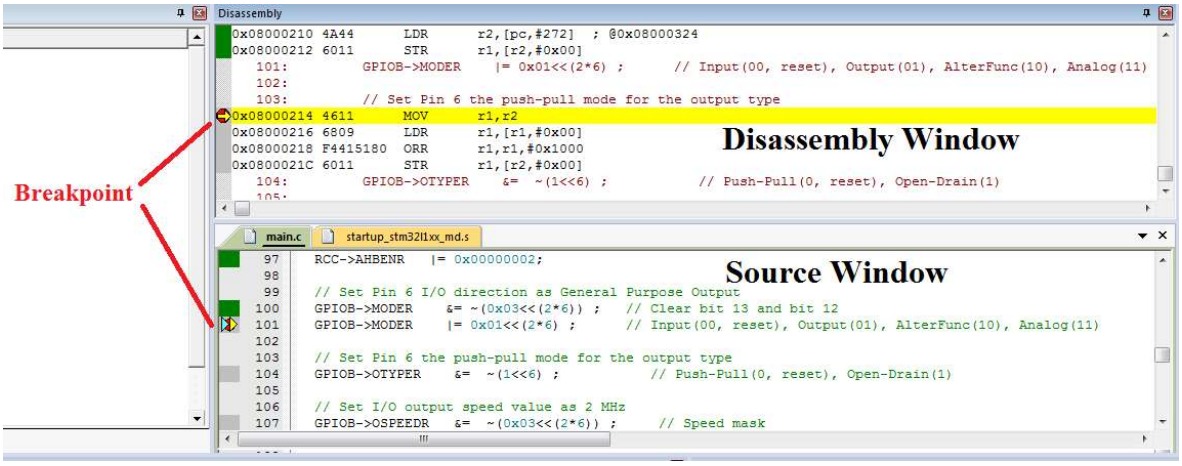
- You can program the STM32 flash by clicking the LOAD button  .
- Click the debug button  to start the debug and click it again to exit the debug. You can use the breakpoint button  to set a break point in either disassembly or source windows.

- STM32 allows up to six breakpoints during hardware debugging. When a program stops at a breakpoint, the corresponding instruction has not been executed yet.
- If the disassembly window is in focus, the debugger executes assembly instructions step by step. If the source window is focused, the debugger then steps through the source lines instead.

The following table summarizes commonly used debug control buttons.

Start Debug	Set a Breakpoint	Run	Stop Debug	Step In	Step Over	Step Out

- **Run**: Continues the execution from the current position until you click ***Stop*** or the program is paused by a breakpoint.
- **Step In**: Execute one step and enter the function if the current step calls a function.
- **Step Over**: Execute one step and run the function all at once if the current step calls a function.
- **Step Out**: Execute until the current function returns.

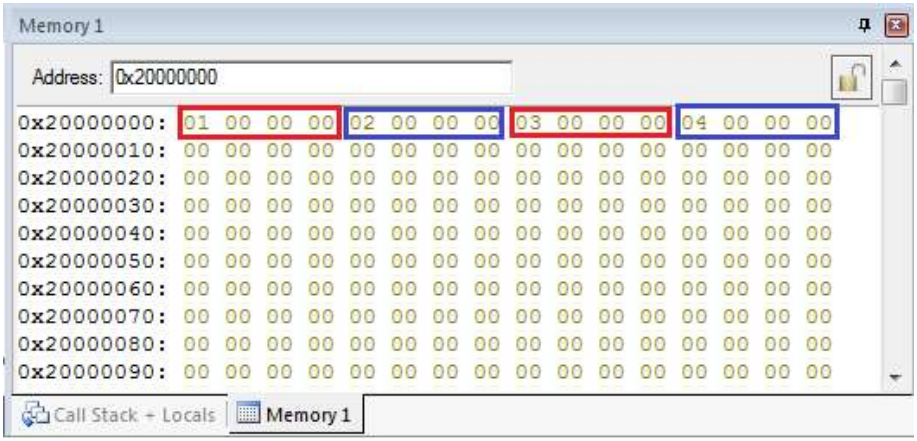


Memory Window

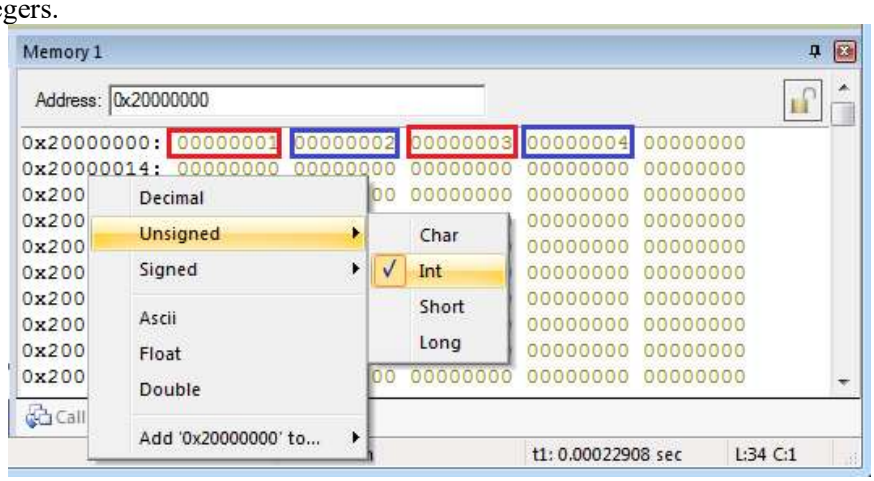
The memory window is used to view the memory content in real time. By default, the address of data memory (RAM) starts at `0x2000_0000`. This is specified in the scatter-loading file (*.sct). The following assembly program defines and allocates an array of four words. Each word consists of four bytes. When we type the memory address 0x20000000, we can see the content of this array.

```
        AREA    myData, DATA, READWRITE
        ALIGN
array DCD   1, 2, 3, 4
```

The memory content is displayed in bytes by default. Little-endian is used to store a word in the memory.



By right click, we can select different display format. For example, we can show the content as unsigned integers.



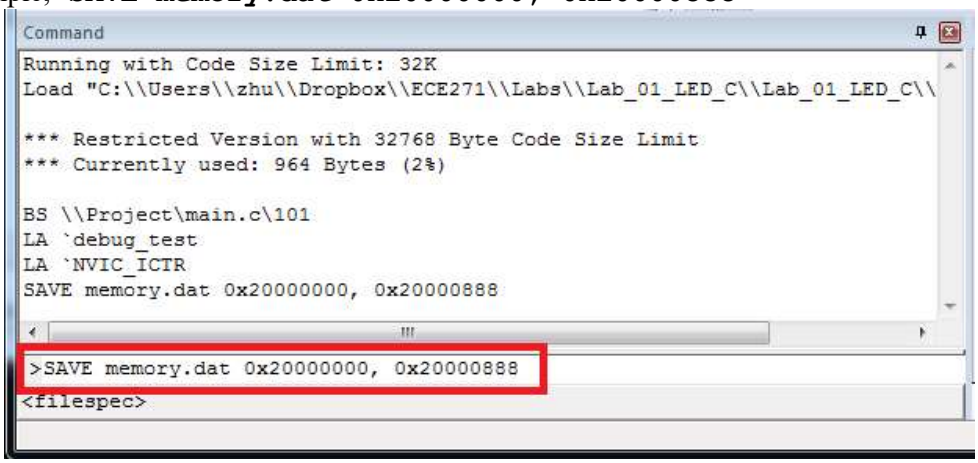
Save Memory Content to a File

In the debug environment, run the following command in the Command Window:

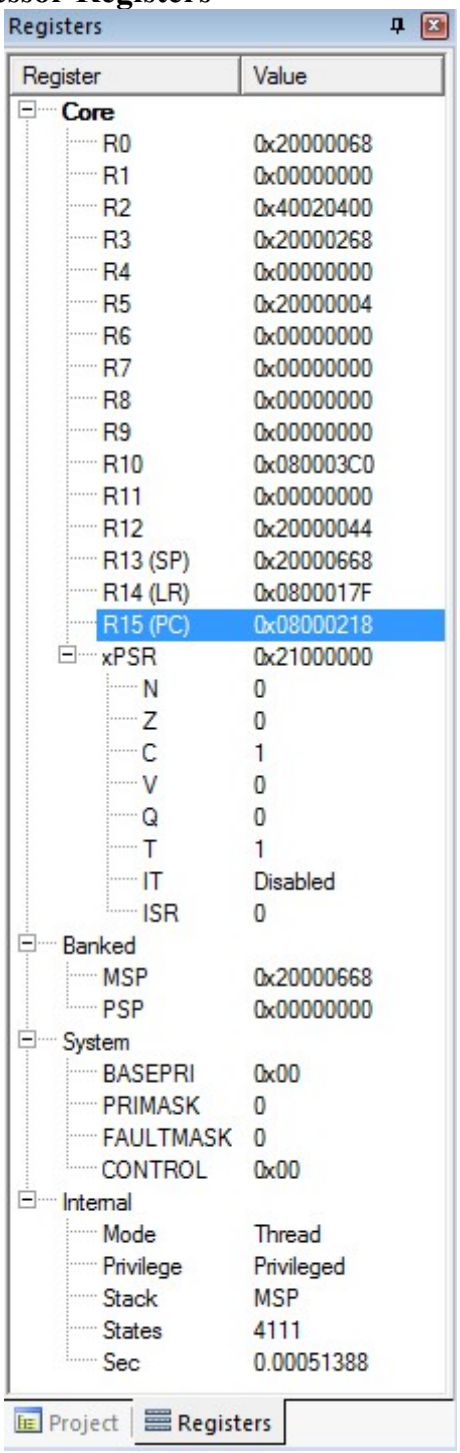
`SAVE <filename> <start address>, <end address>`

This allows you to perform data analysis in other software tools, such as Microsoft Excel and Matlab. The output is saved in Intel HEX format.

For example, **`SAVE memory.dat 0x20000000, 0x20000888`**



Processor Registers



Core Registers:

- Program counter (PC) r15 holds the memory address (location in memory) of the next instruction to be fetched from the instruction memory.
- Stack point (SP) r13 holds a memory address that points to the top of the stack. SP is a shadow of either MSP or PSP.
- xPSR (Special-purpose program status registers) is a combination of the following three processor status registers:
  - Application PSR
  - Interrupt PSR
  - Execution PSR

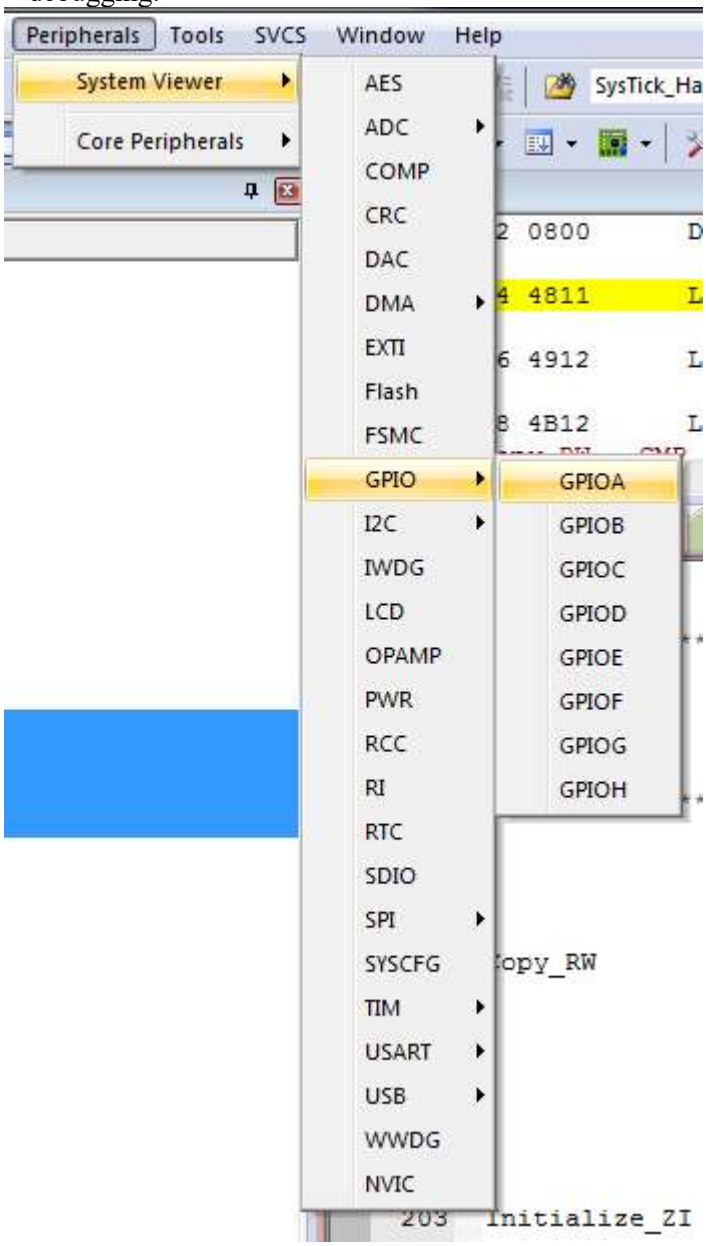
N	gative or less than flag (1 = result negative)
Z	ro flag (1 = result 0)
C	rry or borrow flag (1 = Carry true or borrow false)
V	erflow flag (1 = overflow)
Q	Sticky saturation flag
T	umb state bit
T	Then bits
SR	R Number (6 bits)

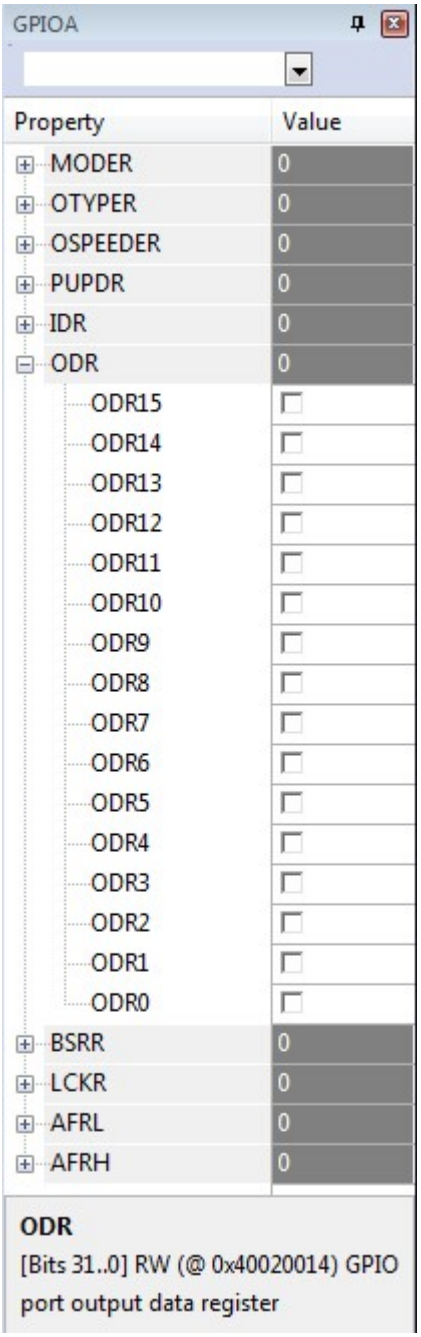
System:

- Base priority mask register (BASEPRI) defines the minimum priority for exception processing.
- Priority mask register (PRIMASK) is used to disable all interrupts excluding hard faults and non-maskable interrupts (NMI). If an interrupt is masked, this interrupt is ignored (i.e. disabled) by the processor.
- Control register (CONTROL) sets the choice of main stack or process stack, and the choice of privileged or unprivileged mode.
- Fault mask register (FAULTMASK) is used to disable all interrupts excluding non-maskable interrupts (NMI).

Peripheral Registers

From the menu: **Peripherals ⟶ System Viewer**, we can view and update the control and data registers of all available peripherals. The following figures show all registers for GPIO Port A, such as mode register (MODER), output type register (OTYPER), Output Speed Register (OSPEEDER), Input Data Register (IDR) and Output Data Register (ODR). This provides great conveniences for debugging.





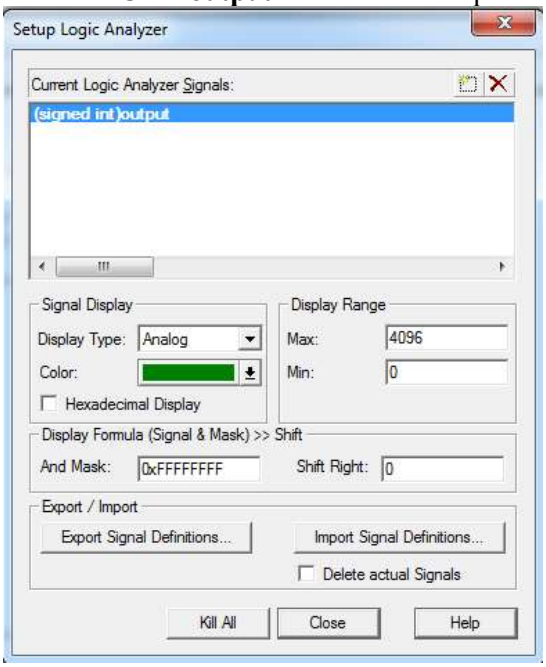
Logic Analyzer

The Logic Analyzer can display the history values (trace) of static or global variables over runtime. Local variables cannot be displayed. The value stored in a register cannot be analyzed.

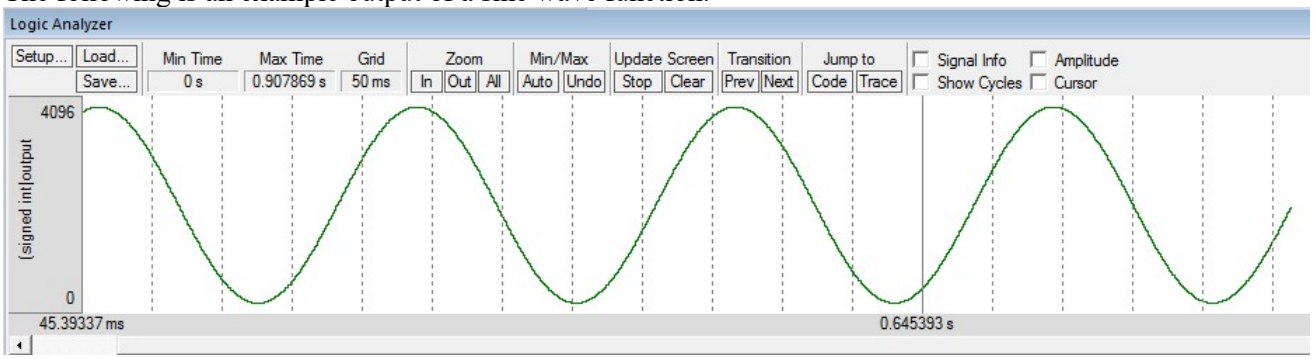
In the following program, we use logic analyzer to monitor a variable “output” defined in the data area.

```
        EXPORT output

        AREA myData, DATA
        ALIGN
output  DCD     0x000
```

In the Logic Analyzer, you can click “Setup” and add a variable “(signed int)output” to observe. Make sure to adjust the data display range to show the curve. Logic analyzer can only monitor global variable. Thus you need to add "**EXPORT output**" to make the output as a global variable.



The following is an example output of a sine wave function.



After this page, please attach printout of the main codes for each section. Use monospace fonts (Such as Noto Mono) for your codes. Use font size 8. Paste them in Microsoft Word. Give appropriate heading for each problem (Such as LED, Pushbutton, Stepper Motor, Lab Exercise of Section 6, etc).

4 Software: Keil MDK v4

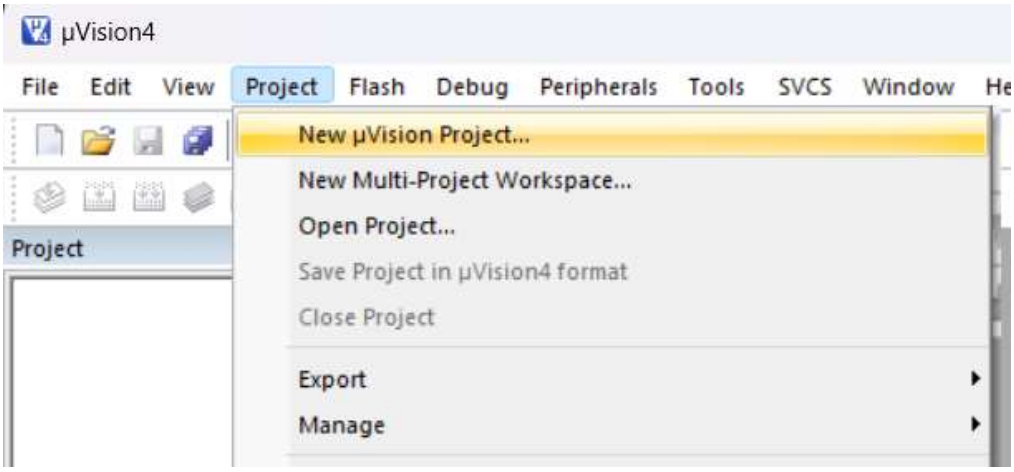
For your Microcontroller Experiments, we will use Keil MDK v5, as it supports the STM32F446 Microcontroller used in the lab. For the theory course, we are following simple instruction sets of Arm v7. Free version of Keil v5 cannot execute legacy ARM instructions, therefore we need Keil MDK v4 as well, for the theory class Assembly codes, and Experiment 5 and 6.

4.1 Installing Keil MDK v4

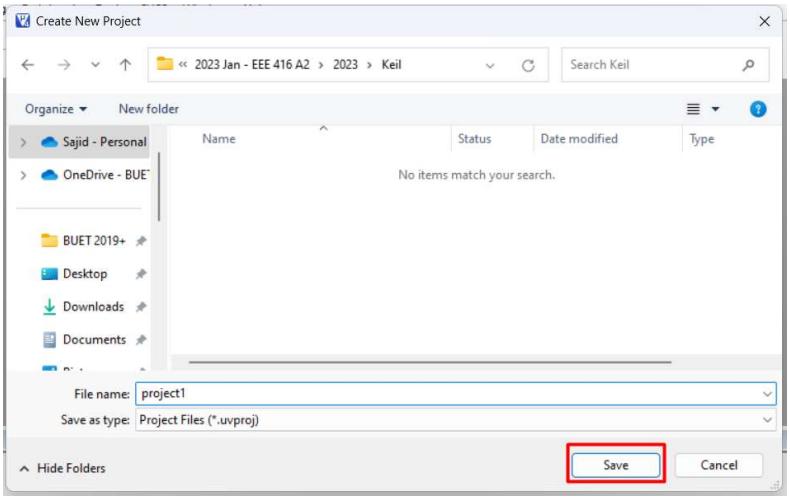
Installer of MDK v4 is given in your teams. Please use it to install Keil MDK v4 in your computer. Use a separate directory for v4 than v5.

4.2 Compiling Program in Keil MDK v4

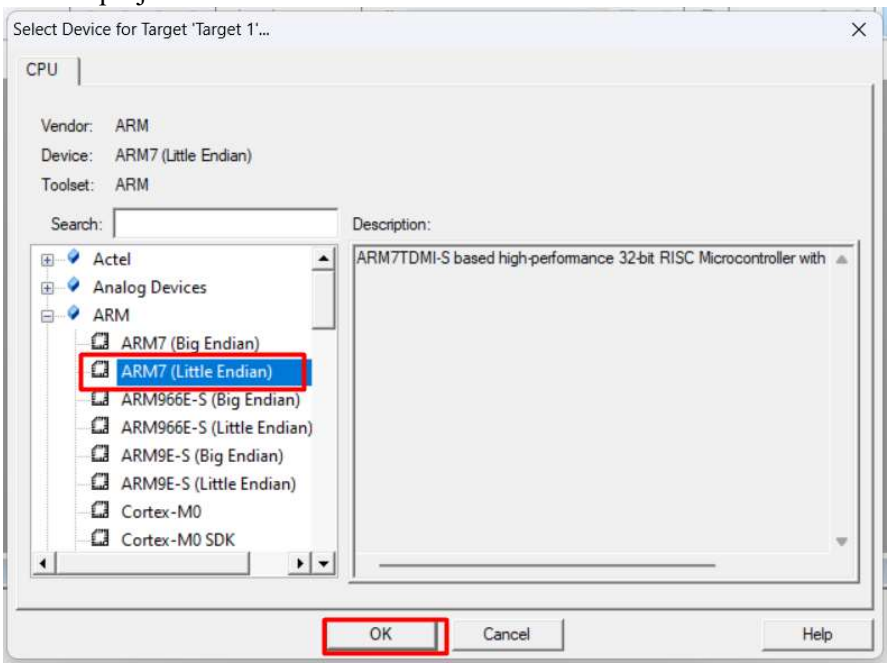
1. Create a new uVision Project in Keil MDK v4 by going to **Project>New uVision Project**



2. Give a Project name and set the project directory

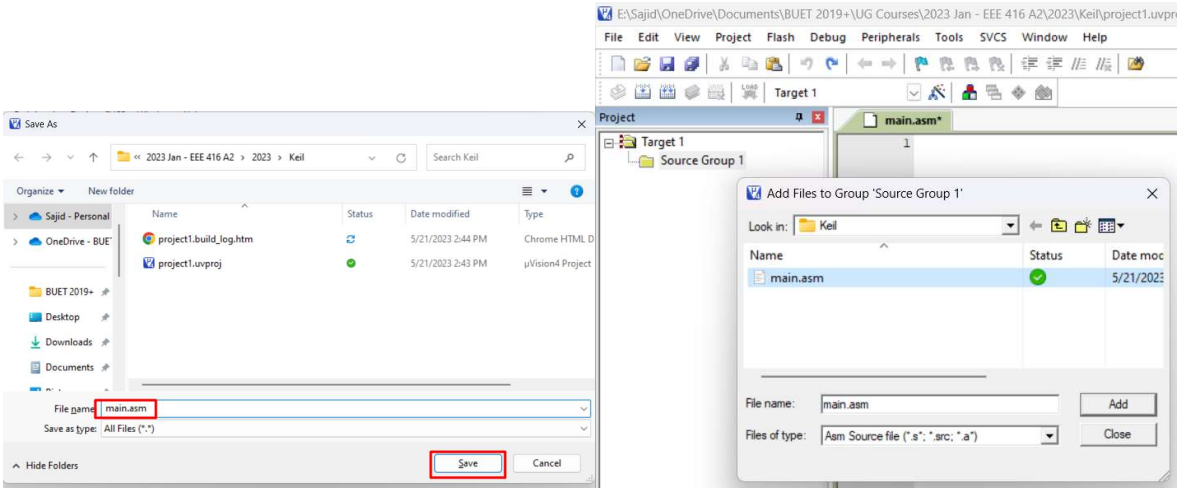


3. For Target Device, select "**ARM>ARM7 (Little Endian)**". Click okay and create the project.



4. Create a blank file. Save that file as **main.asm**

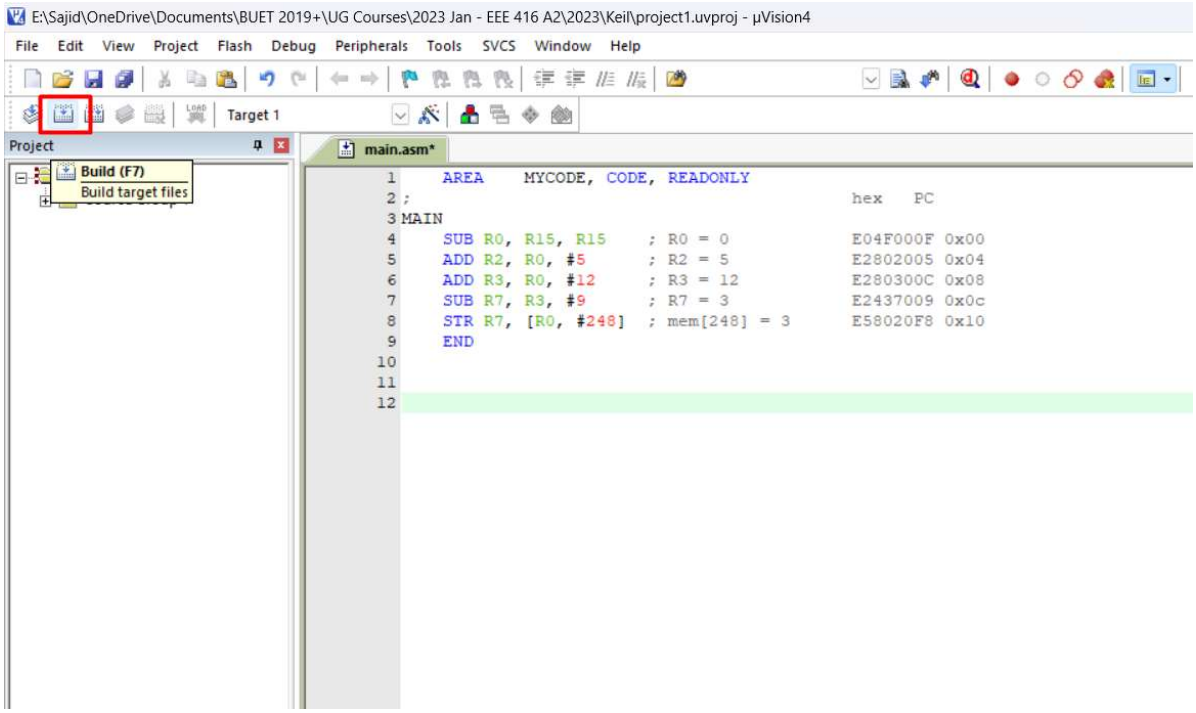
Add main.asm to Source Group 1 of your project.



Write this program in the **main.asm** file:

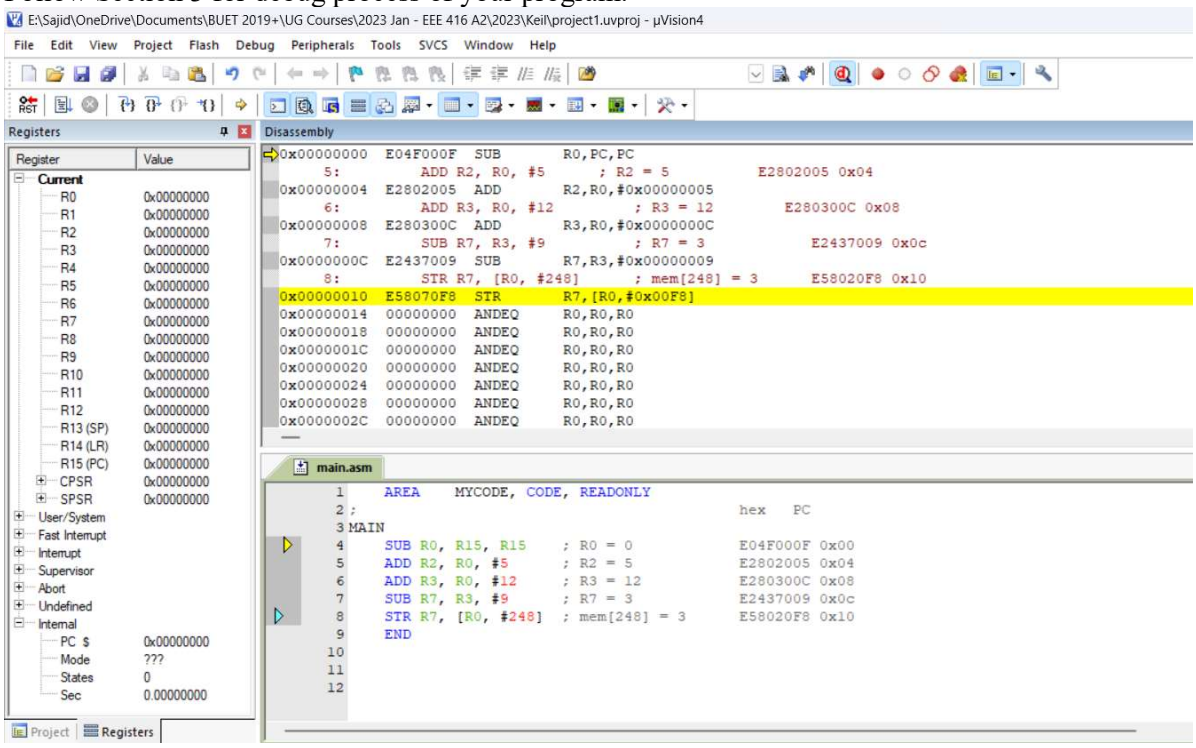
```
    AREA    MYCODE, CODE, READONLY
;                                           hex   PC
MAIN
    SUB R0, R15, R15    ; R0 = 0            E04F000F 0x00
    ADD R2, R0, #5      ; R2 = 5            E2802005 0x04
    ADD R3, R0, #12     ; R3 = 12           E280300C 0x08
    SUB R7, R3, #9      ; R7 = 3            E2437009 0x0c
    STR R7, [R0, #248]  ; mem[248] = 3      E58020F8 0x10
    END
```

5. Click on Build (F7) to build the file



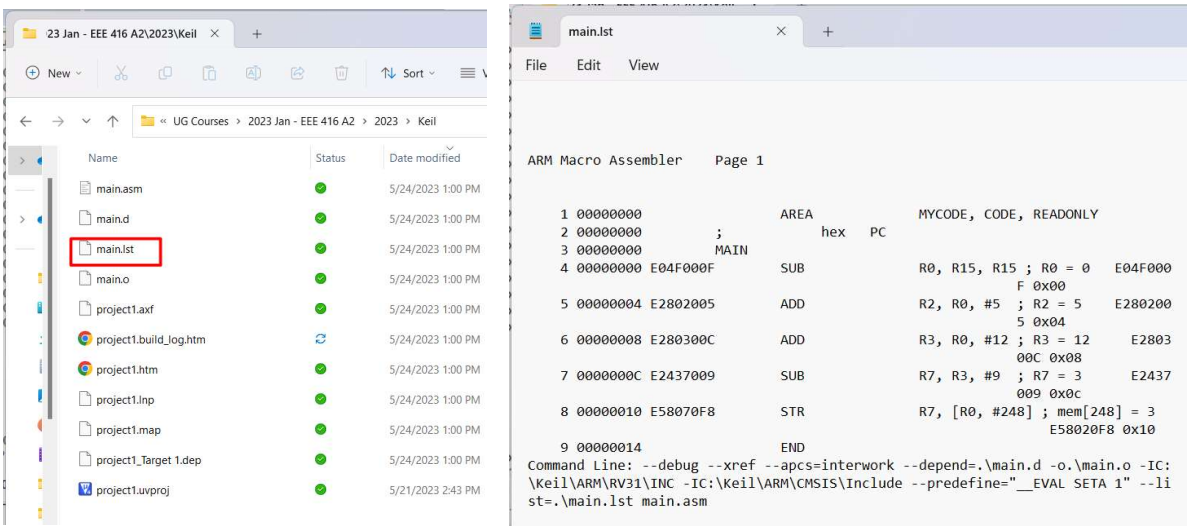
4.3 Debug Program in Keil MDK v4

Follow Section 3 for debug process of your program.



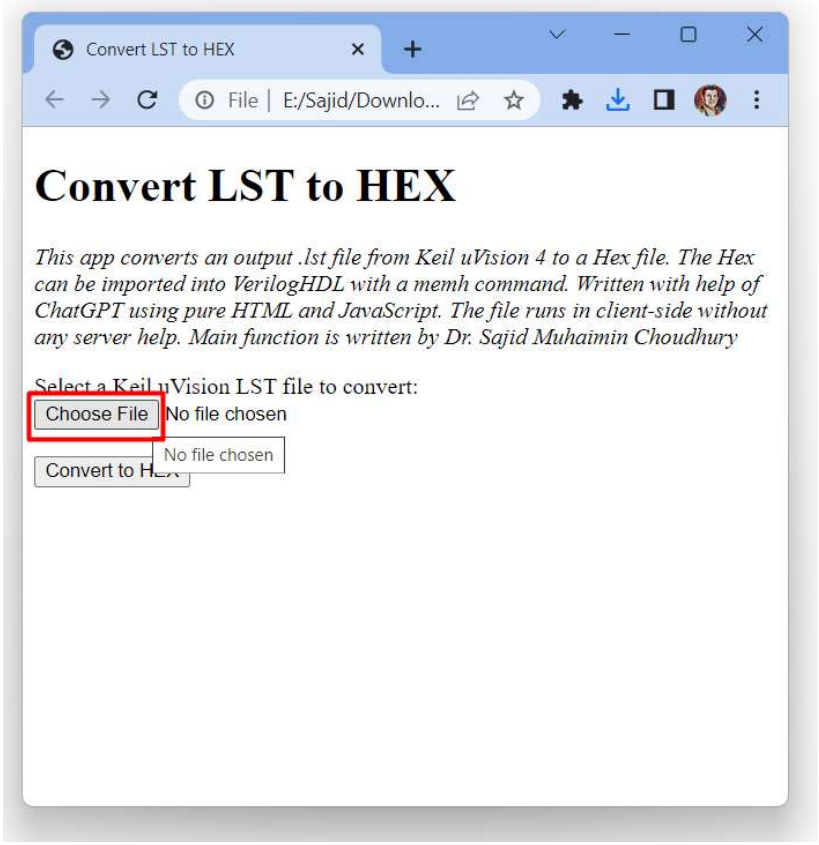
4.4 Keil MDK v4 to Hex File Generation

After successful build of your project, a lst file will be generated that contains the machine code of your project. This machine code would be needed to convert into Hex code for Verilog. Each instruction can be manually copied into a new Notepad file, but for large programs, this may be inconvenient. Therefore, this part is shown on how to quickly create a HEX file from the generated lst file.

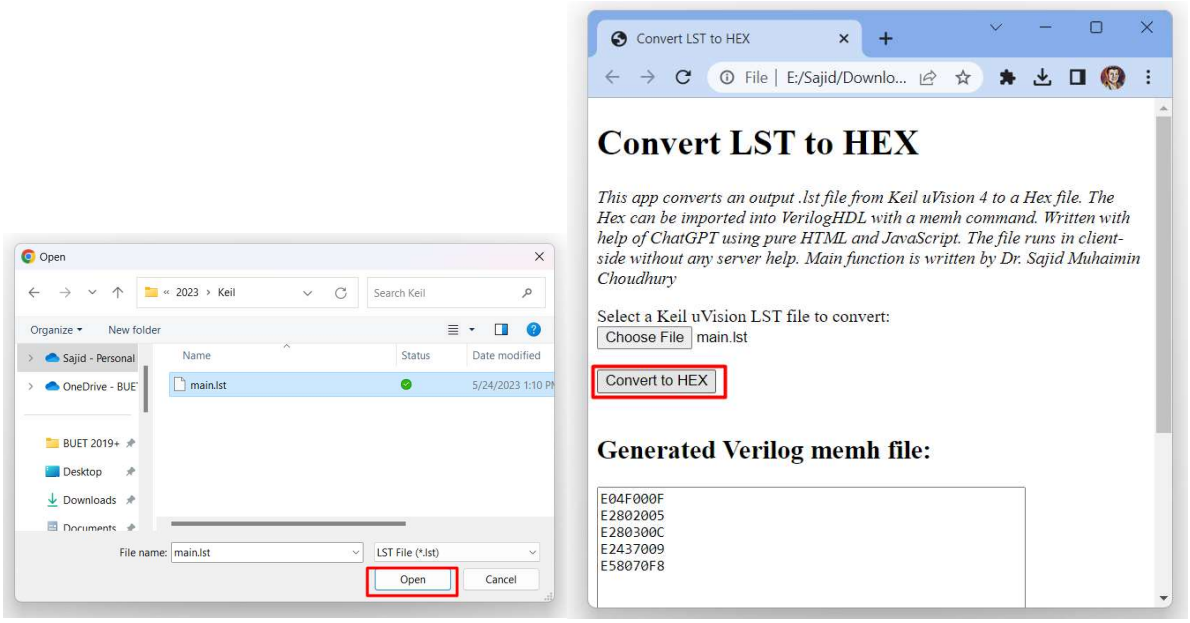


1. Download https://github.com/sajidmc/lst2hex (https://github.com/sajidmc/lst2hex/blob/main/codeconverter.html)
2. Open the codeconverter.html using your browser (Chrome or Edge). lst2hex is based on Javascript, and does not need internet to run.

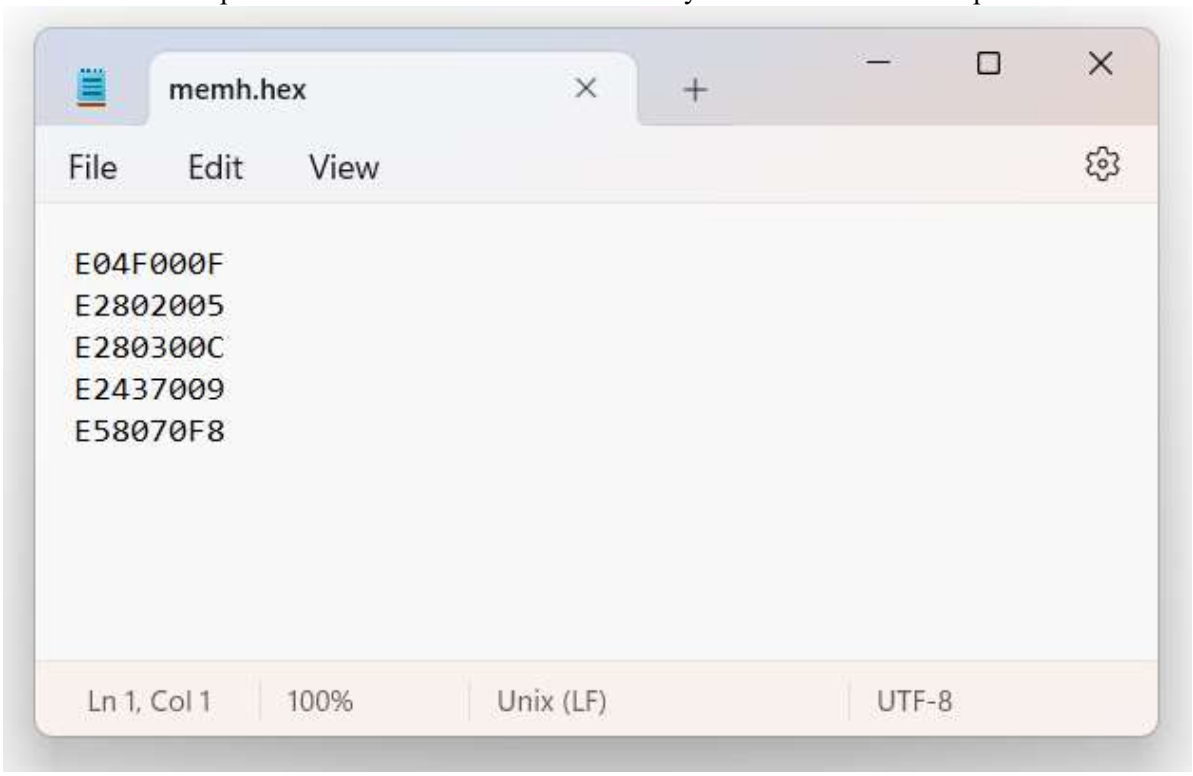
3. Click on Choose File.



4. Select the main.lst file.
5. Click on “Convert to HEX”
6. Scroll down and click on Download Hex.



7. You can open the hex file and see its content. Only the machine code is copied.



5 Software: ModelSim - Intel FPGA Starter Edition

A Verilog compiler is needed to execute a single-cycle processor. In Experiment 7, we will look into more details on the contents of a single-cycle processor. In this lab, we will just see how to install ModelSim -Intel FPGA Starter Edition.

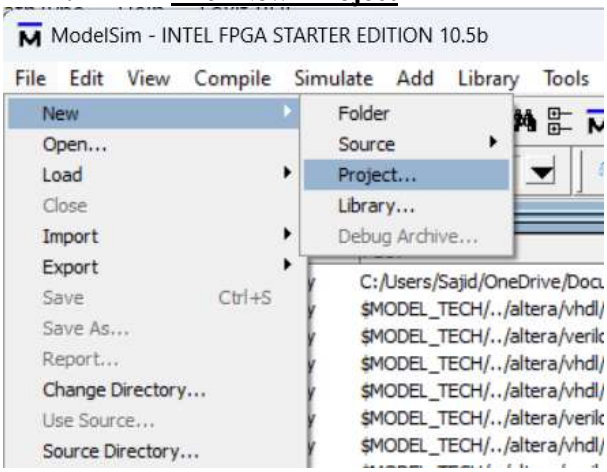
5.1 Installing Verilog Compiler

Download the installer for ModelSim (freely available).
https://www.intel.com/content/www/us/en/software-kit/750368/modelsim-intel-fpgas-standard-edition-software-version-18-1.html \

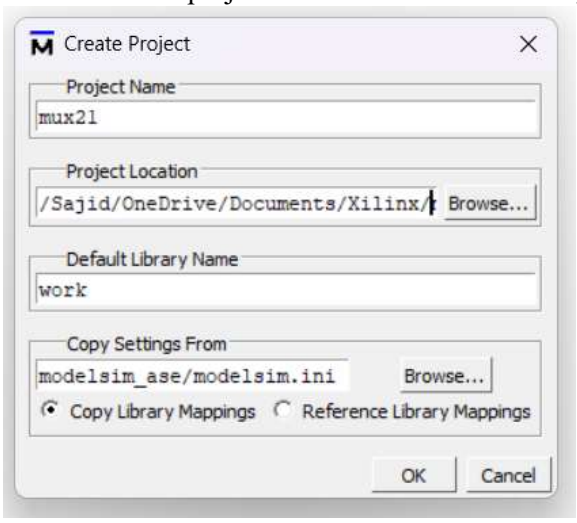
Installation is straightforward, but refer to this YouTube video if you need help 9
https://www.youtube.com/watch?v=C5J_G7qcfm4)

5.2 Creating Base Project

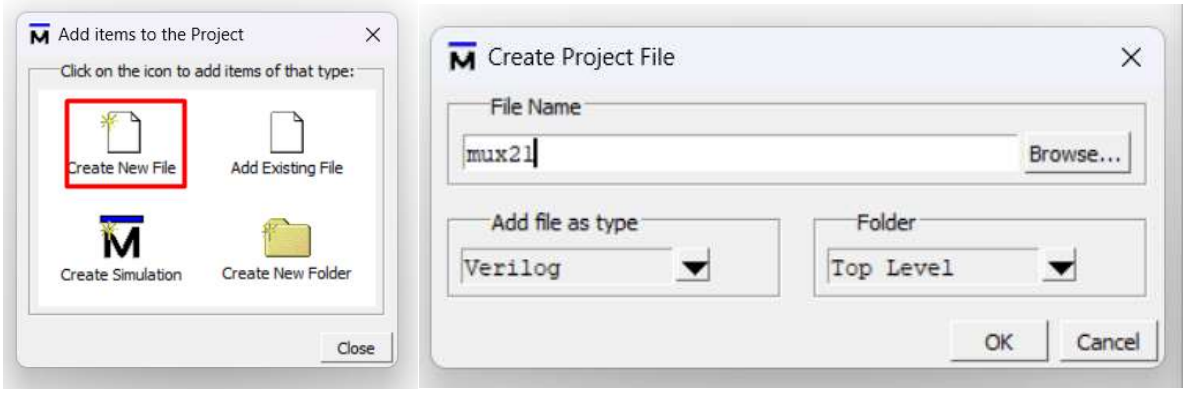
1. Click **File>New>Project**



2. Set the project location in a new directory. Give the name of the project "**mux21**"

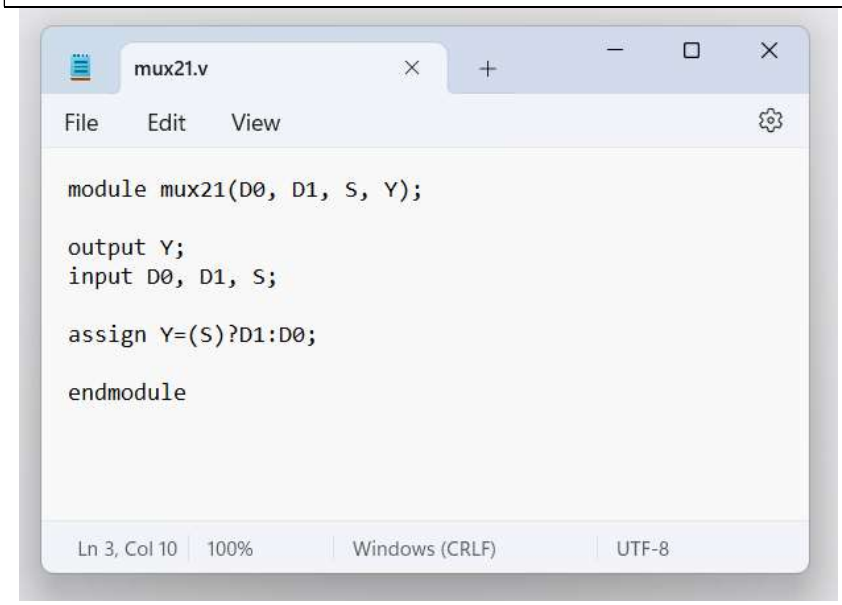


3. Click on “Create New File”. In the new Project file dialoge, select file type as “Verilog” and give name of the file.



4. We are using this code for the mux21.v. Write it in the mux21.v file

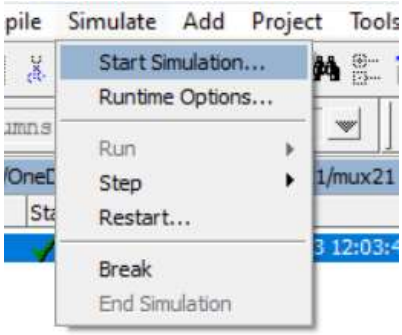
```
module mux21(D0, D1, S, Y);

output Y;
input D0, D1, S;

assign Y=(S)?D1:D0;

endmodule
```



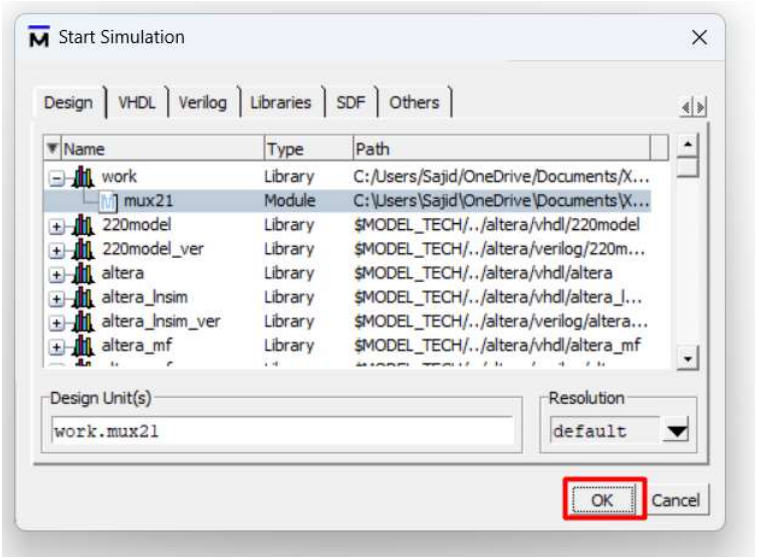
5. Click on Build All button in the toolbar.

5.3 Simulation Test

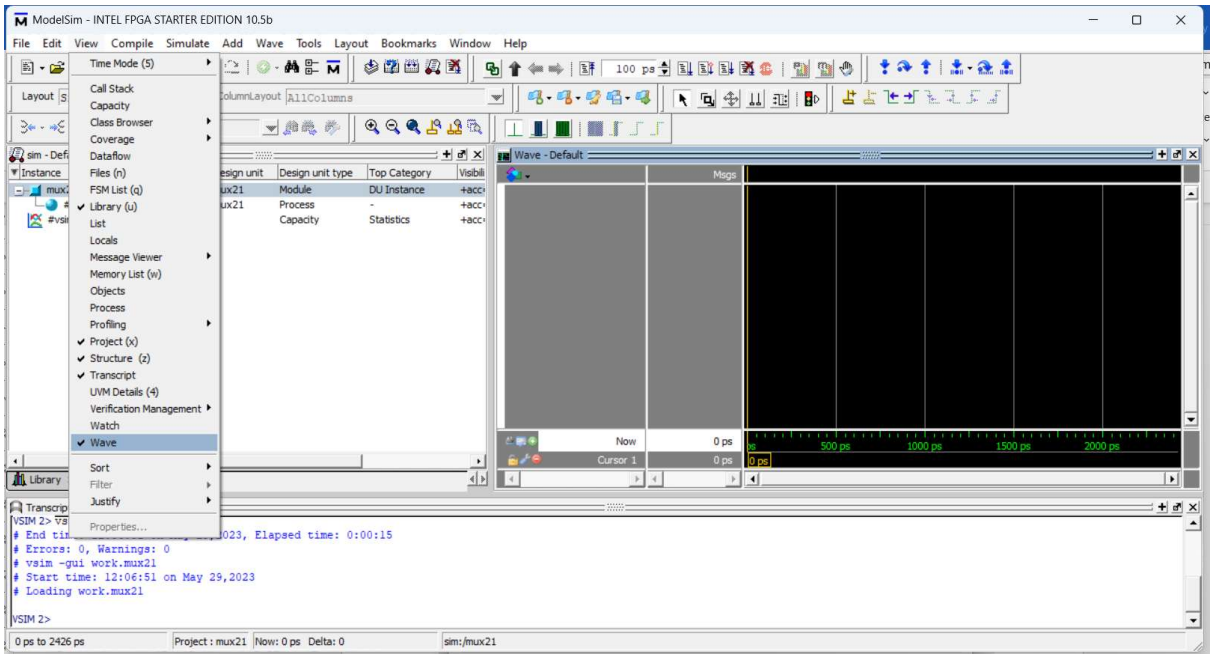
1. Click on **Simulate>Start Simulation**



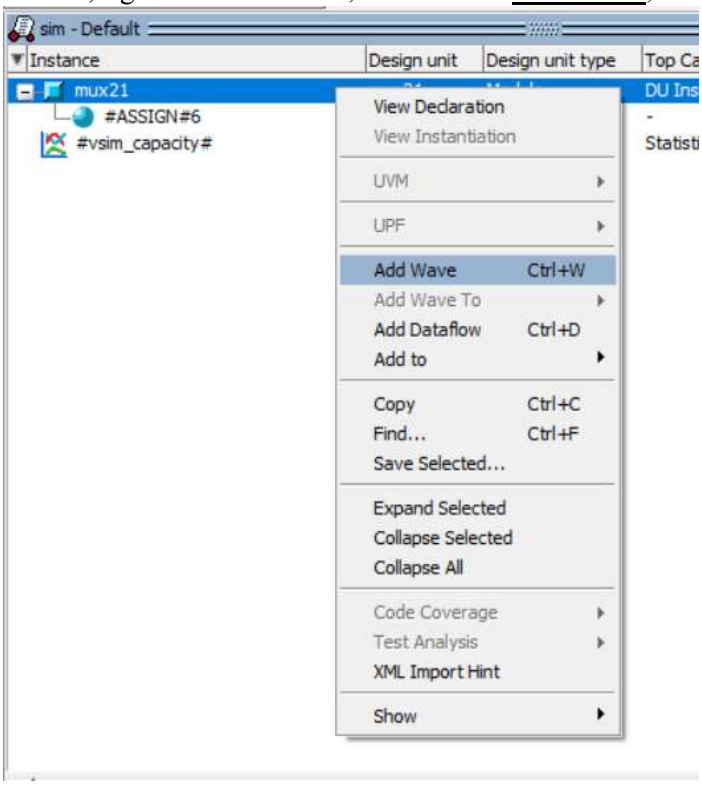
2. In the Start Simulation dialog box, choose the top level entity for simulation. In this project we did not specify a test bench, so the top level would be the Verilog module written.



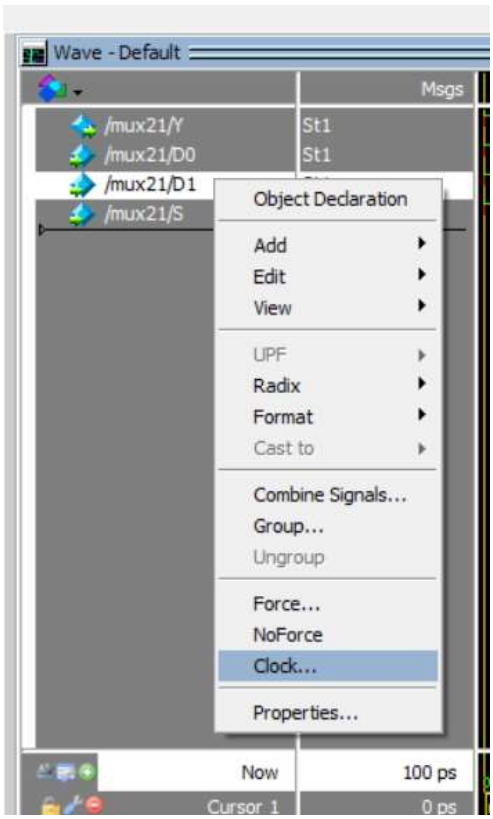
3. The simulation window will appear. Make sure that View>Wave is selected to see the waveform window.

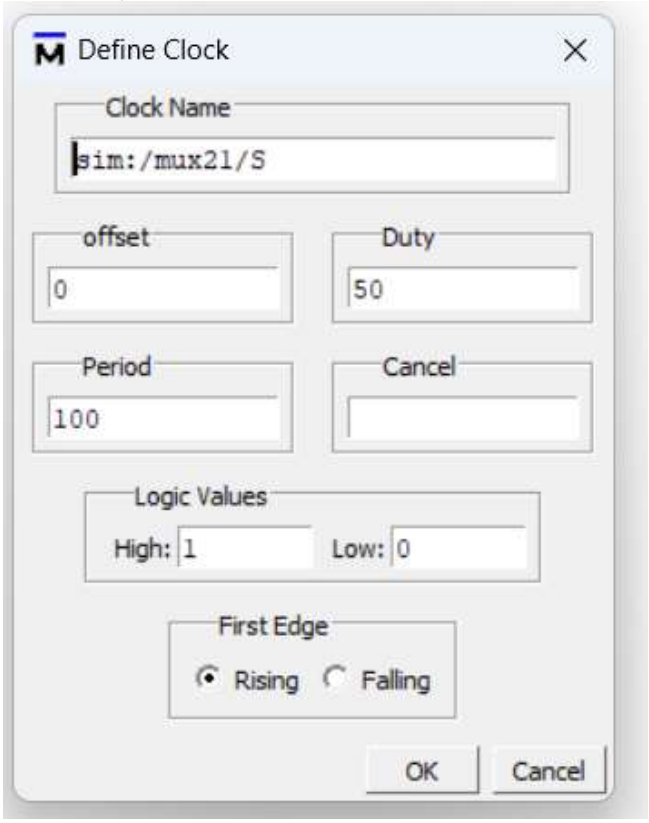


4. In the instances box, right click on mux21, and click on **Add Wave**,

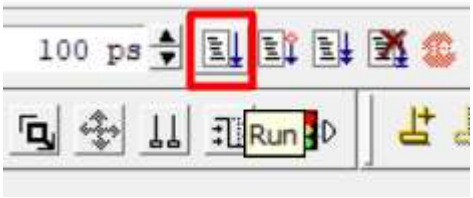


5. The instances of the input and outputs will be added to the wave dialogue. For each input, right click and select Clock. Set different clock period for each input. (S should have period of 100, D0 a period of 5, and D1 a period of 10.)

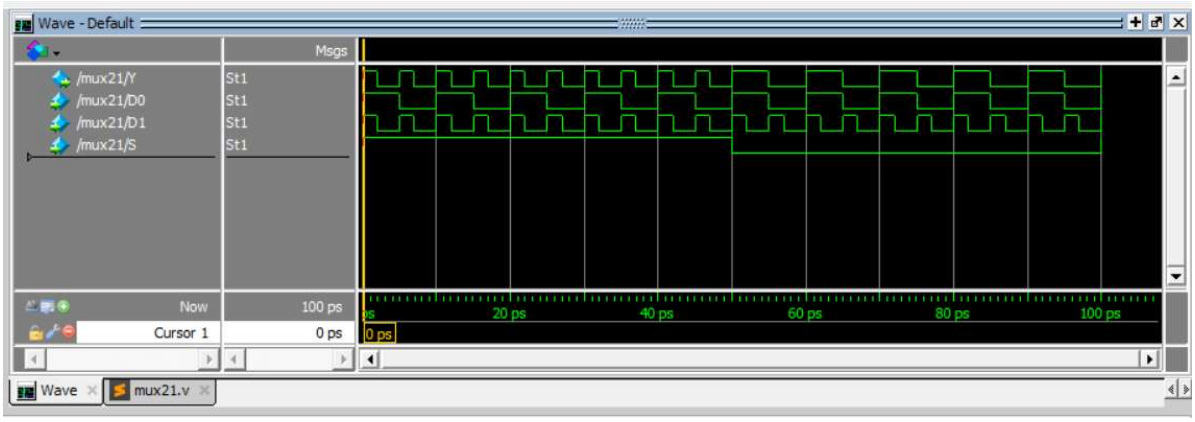




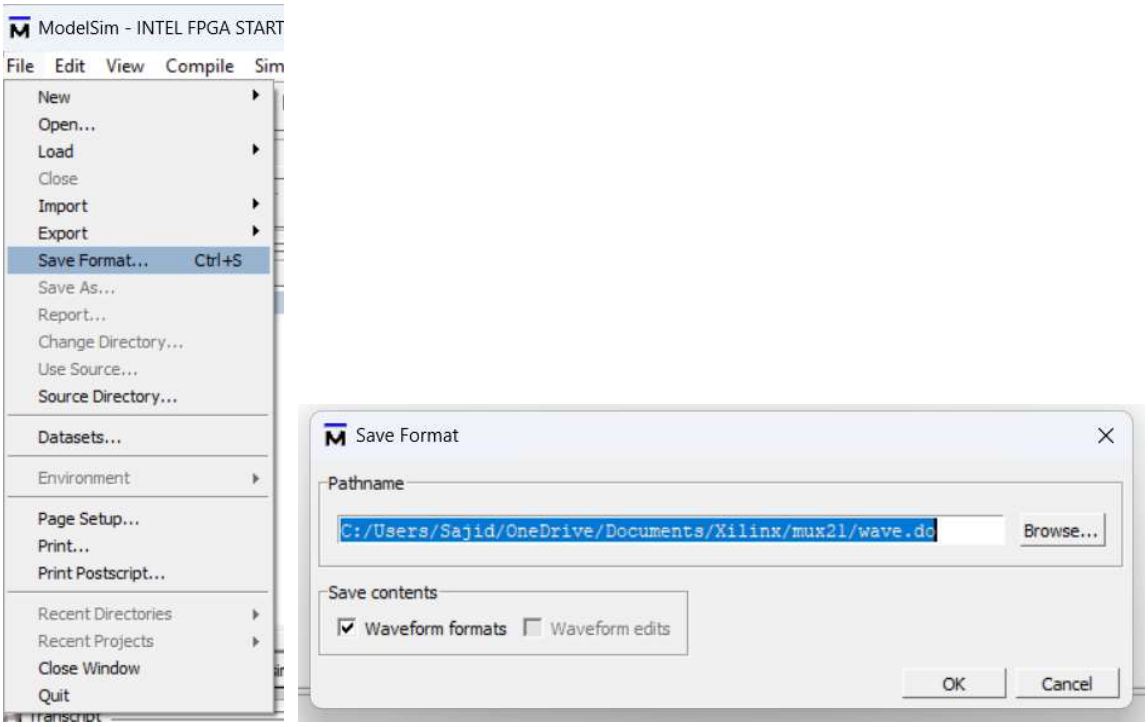
6. On the toolbar, click on Run button



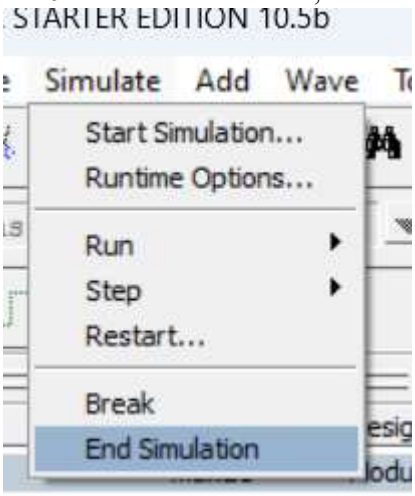
7. You would be able to see the waveforms.

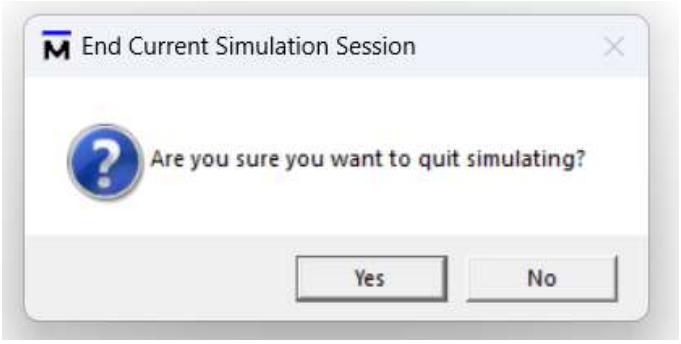


8. To save the waveforms that are added into the window, please click **File>Save Format,** and save the waveforms as a do file.

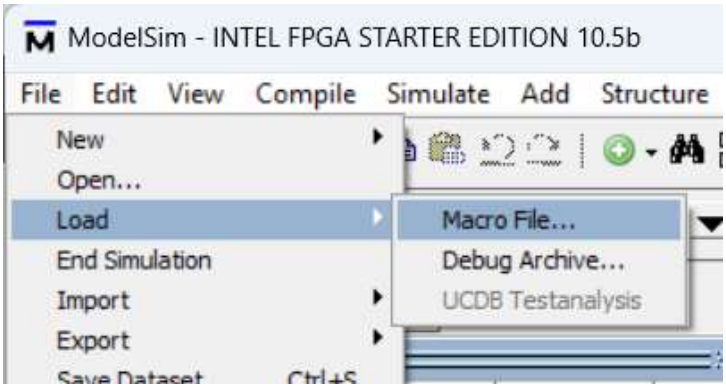


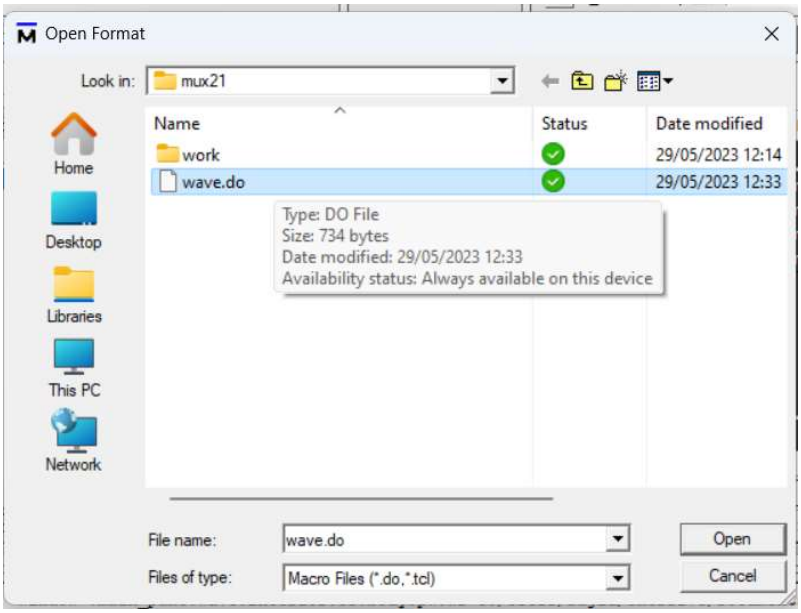
9. To end simulation, Click on Simulate>End Simulation.





10. Next time when you start the simulation, the waveform window will be blank. Click on **Load>Macro File…** and select the do file to reload the waveforms.



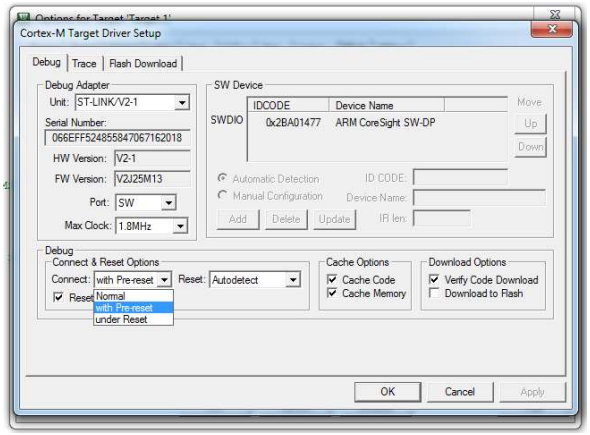


Appendix A: “Target not found” error

When you use the STM32F4 discovery board at the very first time, you might not be able to program it in Keil and receive an error of “***Target not found***” when you download the code to the board. This is because the demo program quickly puts the STM32F4 microcontroller into a very low power mode after a reset. There are several ways to solve it. Below are two simplest ones. The error will go away permanently.

Solution 1: In Keil,

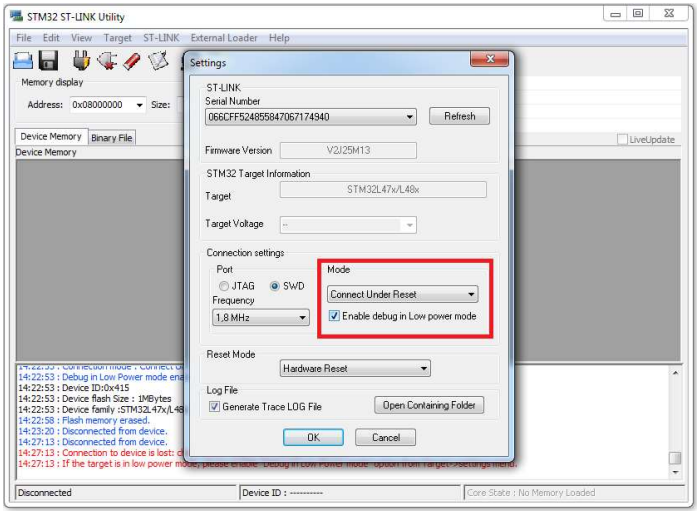
1. Click the icon "***Options for Target***"
2. Click "***Debug***" and then "***Settings***"
3. Change the connect from the default value "***normal***" to "***with pre-reset***", as shown below.



Solution 2: If the previous solution fails, you can download and install ***STM32 ST-Link Utility***
http://www.st.com/en/development-tools/stsw-link004.html

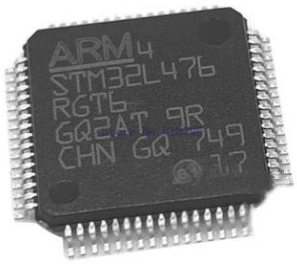
Follow the following steps:

1. Run ST-Link Utility, click menu "***Target***", click "***Settings***"
2. Select "***Connect Under Set***" as the connection mode
3. Click "***Target***" and "***Connect***", and then click "***Target***" and "***Erase Chip***"!
4. Click "***Target***" and "***Disconnect***"



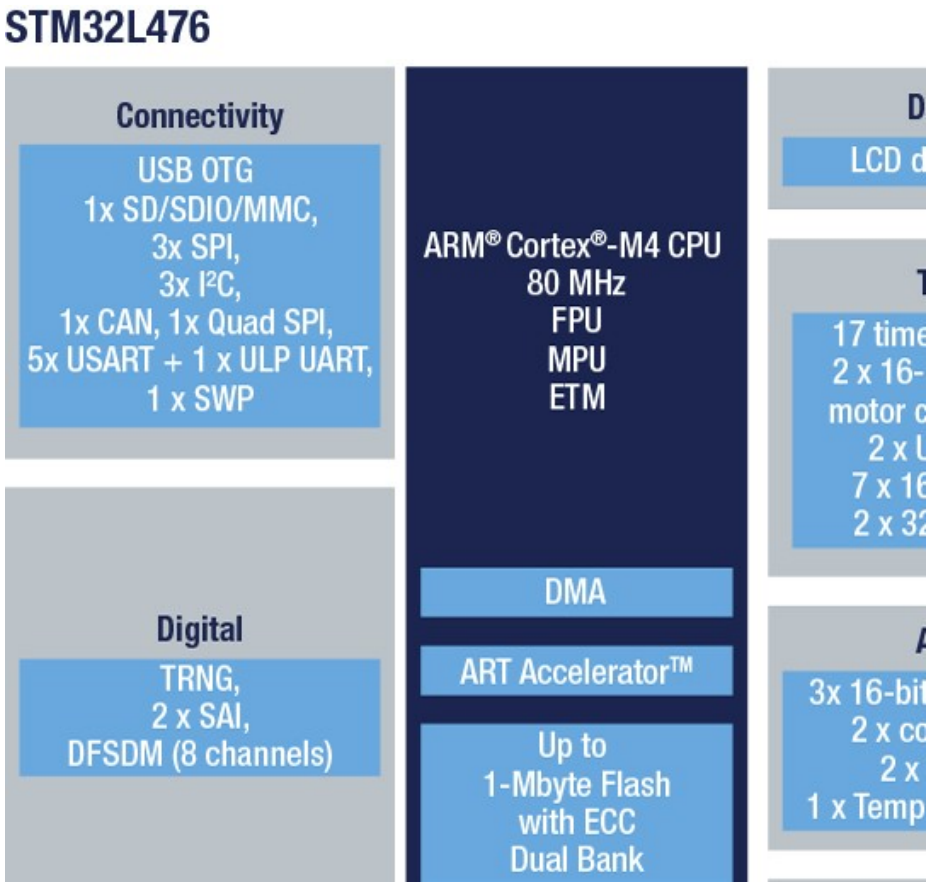
Appendix B: Microcontroller STM32L476

The NUCLEO-L476RG board has a 32-bit Arm Cortex-M4F microcontroller. This is used in our text book Yifeng Zhu "Embedded Systems with Arm Cortex-M Microcontrollers".



- Part Number: **STM32L476RGT6**
- Total number of pins: 64
- Total number of I/O pins: 51
- Maximum clock frequency: 80MHz
- Program memory size: 1 MB
- Operating supply voltage: 1.71V to 3.6V
- I/O voltage: 3.3V
- Analog supply voltage: 3.3V

The following figure summarizes the I/O connection supported and internal peripherals.



Appendix C: Microcontroller STM32F446

The NUCLEO-L446RE board has a 32-bit Arm Cortex-M4F microcontroller.
This board is used in BUET's Microprocessor and Embedded Systems lab.

- Part Number: **STM32F446RE**
- Total number of pins: 64
- Total number of I/O pins: 51
- Maximum clock frequency: **180MHz**
- Program memory size: **512 kB**
- Operating supply voltage: 1.71V to 3.6V
- I/O voltage: **1.7-3.6V**
- Analog supply voltage: **1.7-3.6V**

The following figure summarizes the I/O connection supported and internal peripherals.

Appendix D: Pin Connections on Nucleo-64 board

This diagram is common for both Nucleo-L476RG and Nucleo-L446RE

Board Component	Microcontroller Pin	Comment
Green LED	PA 5	SB42 closed and SB29 open by default
Blue user button	PC 13	Pulled up externally
Black reset button	NRST	Connect to ground to reset
ST-Link UART TX	PA 2	STLK_TX
ST-Link UART RX	PA 3	STLK_RX
ST-Link SWO/TDO	PB 3	Trace output pin/Test Data Out pin
ST-Link SWDIO/TMS	PA 13	Data I/O pin/Test Mode State pin
ST-Link SWDCLK/TCK	PA 14	Clock pin/Test Clock pin

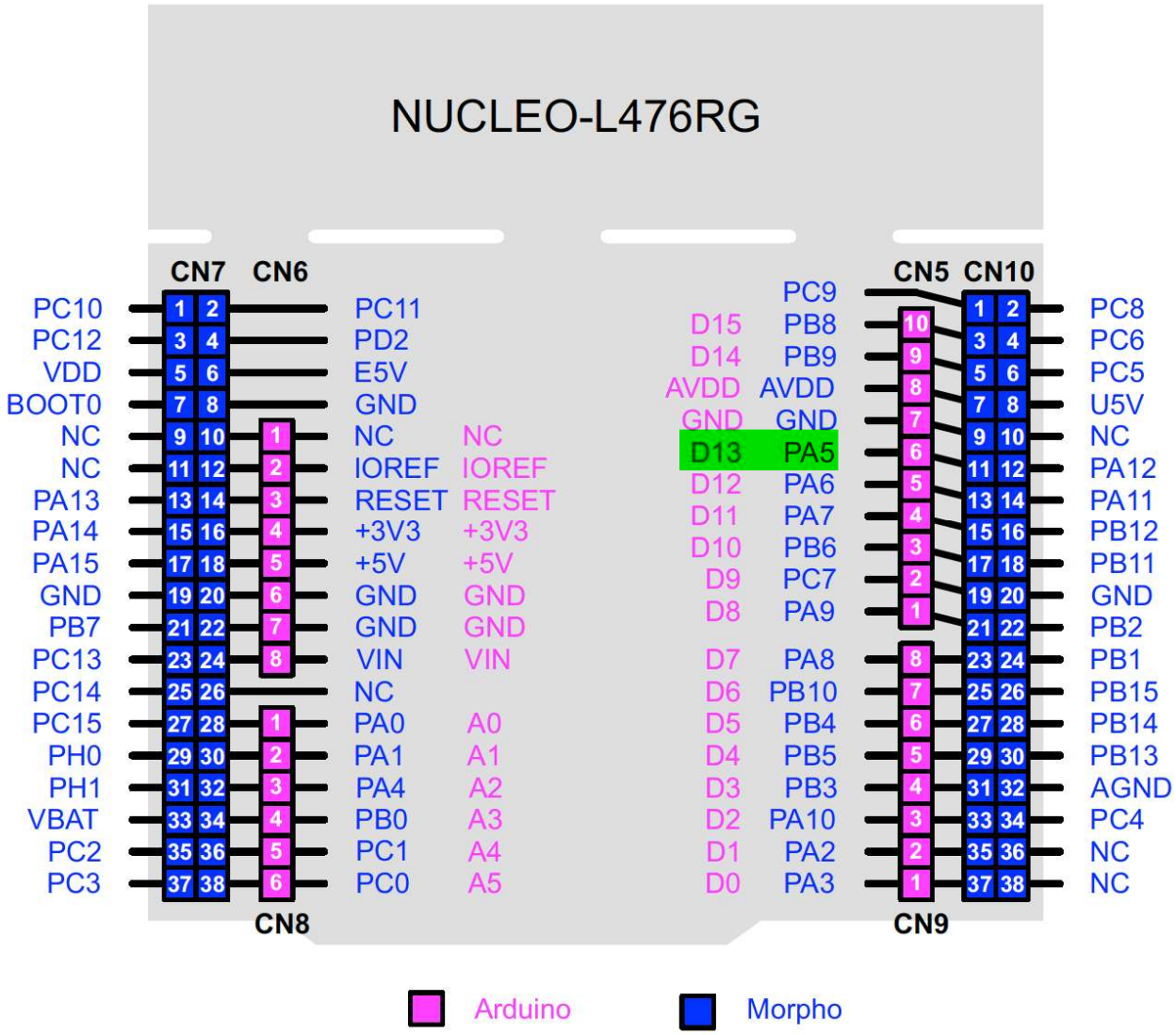


Table 12. ARDUINO® connector pinout

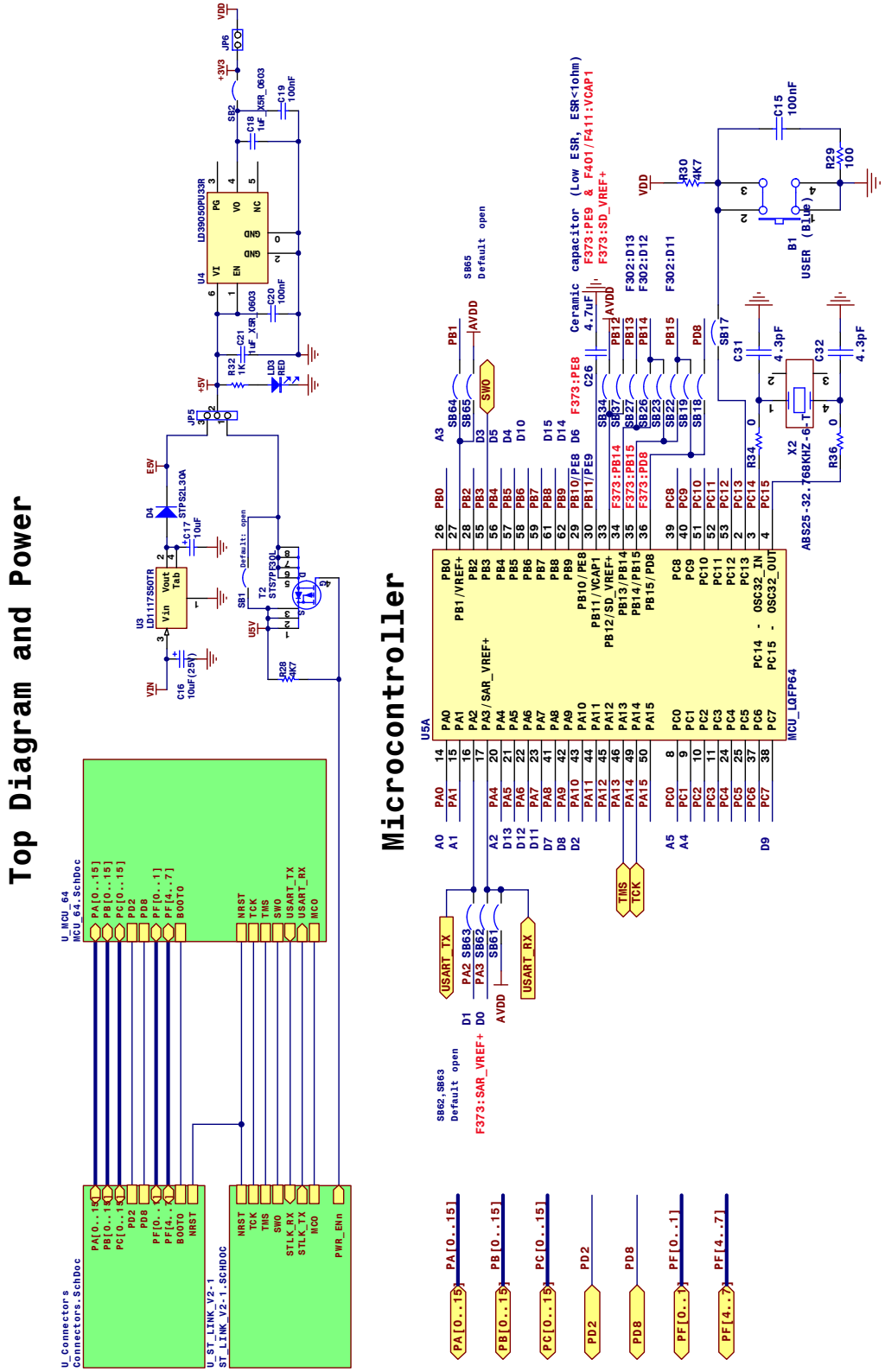
Connector	Pin number	Pin name	Signal name	STM32 pin	Function
CN6	1	NC	-	-	Reserved for test
	2	IOREF	-	-	I/O reference
	3	NRST	NRST	NRST	RESET
	4	3V3	-	-	3.3 V input/output
	5	5V	-	-	5 V output
	6	GND	-	-	GND
	7	GND	-	-	GND
	8	VIN	-	-	7 V to 12 V power input
CN8*	1	A0	ADC	PA0	ARD_A0_IN0
	2	A1	ADC	PA1	ARD_A1_IN1
	3	A2	ADC	PA4	ARD_A2_IN4
	4	A3	ADC	PB1	ARD_A3_IN9
	5	A4	ADC	PB9 or PB11	ARD_A4_IN15‖ I2C_1_SCL
	6	A5	ADC	PB8 or PB12	ARD_A5_IN16‖ I2C_1_SDA
CN5	10	SCL/D15	ARD_D15	PB8	I2C_1_SCL
	9	SDA/D14	ARD_D14	PB9	I2C_1_SDA
	8	AVDD	VREF+	-	VREF+
	7	GND	-	-	GND
	6	SCK/D13	ARD_D13	PA5	SPI_1_SCK
	5	MISO/D12	ARD_D12	PA6	SPI_1_MISO
	4	PWM/MOSI/D11	ARD_D11	PA7	SPI_1_MOSI ‖ TIM_14_CH1
	3	PWM/CS/D10	ARD_D10	PB0	SPI_1_NSS ‖ TIM_3_CH3
	2	PWM/D9	ARD_D9	PC7	TIM_3_CH2
	1	D8	ARD_D8	PA9	IO
CN9	8	D7	ARD_D7	PA8	IO
	7	PWM/D6	ARD_D6	PB14	TIM_15_CH1
	6	PWM/D5	ARD_D5	PB4	TIM_3_CH1
	5	D4	ARD_D4	PB5	IO
	4	PWM/D3	ARD_D3	PB3	TIM_1_CH2
	3	D2	ARD_D2	PA10	IO
	2	TX/D1	ARD_D1	PC4	UART_1_TX
	1	RX/D0	ARD_D0	PC5	UART_1_RX

Table: STM32 Nucleo-64 board I/O assignment

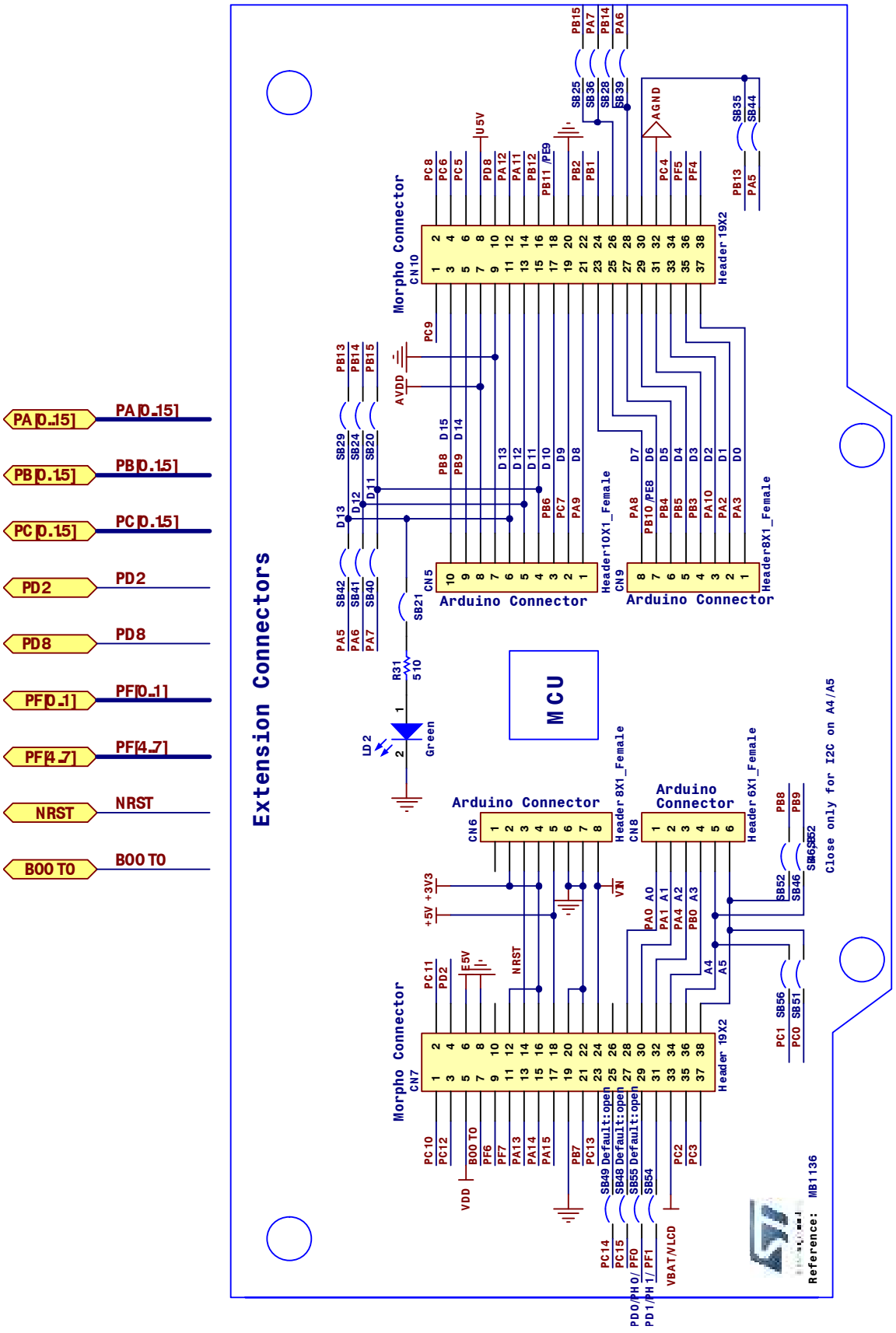
Pin No	Pin name	Signal or label	Main feature / optional feature (SB)
1	PC11	PC11	IO
2	PC12	PC12	IO
3	PC13	PC13	User Button/IO
4	PC14 - OSC32_IN	PC14 - OSC32_IN	LSE CLK/IO
5	PC15 - OSC32_OUT	PC15 - OSC32_OUT	LSE CLK/IO
6	PF3 - VBAT	VBAT	PWR VBAT
7	PF4 - VREF+	AVDD	PWR AVDD
8	VDD_1	VDD	PWR VDD
9	VSS_1	GND	PWR GND
10	PF0 - OSC_IN	PF0 - OSC_IN	HSE CLK/IO
11	PF1 - OSC_OUT	PF1 - OSC_OUT	HSE CLK/IO
12	PF2 - NRST	PF2 - NRST	RESET
13	PC0	PC0	IO
14	PC1	PC1	IO
15	PC2	PC2	IO
16	PC3	PC3	IO
17	PA0	PA0	ARD_A0_IN0
18	PA1	PA1	ARD_A1_IN1
19	PA2	UART2_TX	STLK_RX
20	PA3	UART2_RX	STLK_TX
21	PA4	PA4	ARD_A2_IN4
22	PA5	PA5	ARD_D13 || SPI_1_SCK
23	PA6	PA6	ARD_D12 || SPI_1_MISO
24	PA7	PA7	ARD_D11 || SPI_1_MOSI || TIM_14_CH1
25	PC4	PC4	ARD_D1 || UART_1_TX
26	PC5	PC5	ARD_D0 || UART_1_RX
27	PB0	PB0	ARD_D10 || SPI_1_NSS || TIM_3_CH3
28	PB1	PB1	ARD_A3_IN9
29	PB2	PB2	IO
30	PB10	PB10	IO
31	PB11	PB11	ARD_A4_IN15
32	PB12	PB12	ARD_A5_IN16

Appendix E: Nucleo-64 board Schematics

E1. Microcontroller Unit Schematic



E2. External Connectors Schematic



Appendix F: Code Comments and Documentation

Program comments are used to improve code readability, and to assist in debugging and maintenance. A general principal is "***Structure and document your program the way you wish other programmers would***" (McCann, 1997).

The book titled "*The Elements of Programming Style*" by Brian Kernighan and P. J. Plauger gives good advices for beginners.

- **Format your code well.** Make sure it's easy to read and understand. Comment where needed but don't comment obvious things it makes the code harder to read. If editing someone else's code, format consistently with the original author.
- Every program you write that you intend to keep around for more than a couple of hours ought to have documentation in it. Don't talk yourself into putting off the documentation. A program that is perfectly clear today is clear only because you just wrote it. Put it away for a few months, and it will most likely take you a while to figure out what it does and how it does it. If it takes you a while to figure it out, how long would it take someone else to figure it out?
- **Write Clearly** - don't be too clever - don't sacrifice clarity for efficiency.
- **Don't over comment.** Use comments only when necessary.
- Format a program to help the reader understand it. **Always Beautify Code.**
- Say what you mean, **simply and directly**.
- **Don't patch bad code** - rewrite it.
- **Make sure comments and code agree.**
- Don't just echo code in comments - **make every comment meaningful.**
- **Don't comment bad code** - rewrite it.
- **The single most important factor in style is consistency.** The eye is drawn to something that "doesn't fit," and these should be reserved for things that are actually different.

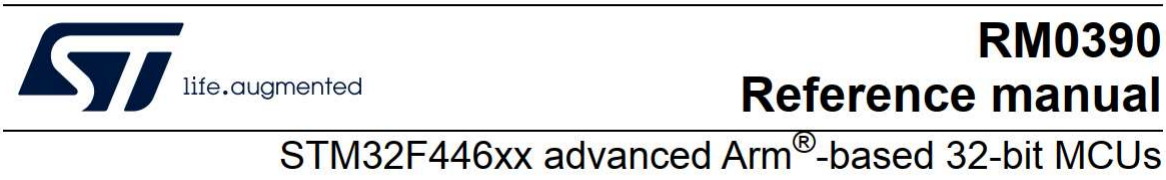
Appendix G: Acknowledgement and References

The labsheet is prepared by **Dr. Sajid Muhaimin Choudhury**, Dept of EEE, BUET, on 04/03/2023

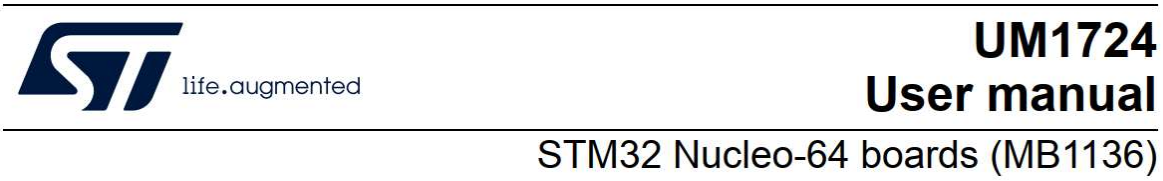
The labsheet is based on own materials and Lab Materials provided as Instuctor Suppliment of "Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, Third Edition." By Dr. Yifeng Zhu,. The Labsheets are modified from STM32L4 architecture to STM32F4 architecture. STM32F4 Documentations are taken from st.com

The following documents are strongly recommended as reference:

Digital Design and Computer Architecture: ARM Edition 1st Edition by Sarah Harris , David Harris

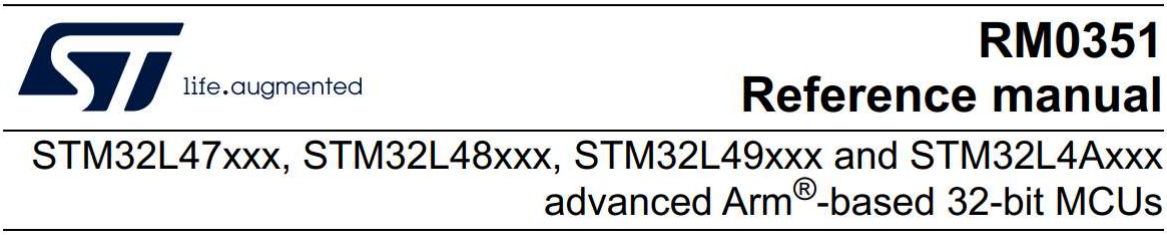


https://www.st.com/resource/en/reference_manual/rm0390-stm32f446xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf



https://www.st.com/resource/en/user_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf

The following Document is recommended for Yifeng Zhu's book literature



https://www.st.com/resource/en/reference_manual/rm0351-stm32l47xxx-stm32l48xxx-stm32l49xxx-and-stm32l4axxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf