

EE3220: System-On-Chip Design

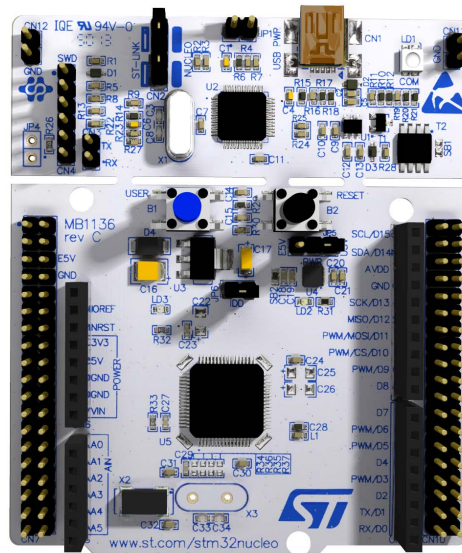
Lab 1: Embedded System Design with STM32F401RE ARM Board

Objectives

At the end of this exercise, students should be able to:

- Familiarize with Nucleo-F401RE board and STM32F401RE microcontroller
- Create new Mbed OS project in the Keil Studio Cloud
- Write a C/C++ program to control LED with a user button and a keyboard
- Communicate with the board via USART serial communication
- Program the board from the ARM Development Studio

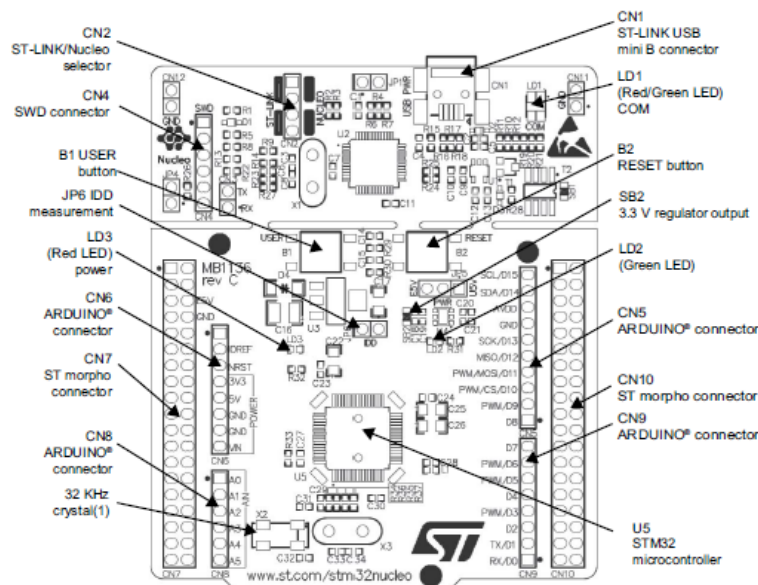
System overview:



This laboratory session familiarises students with the STM32-Nucleo-F401RE board and the Keil Studio Cloud for the development of ARM Mbed programs. Students will write a C/C++ program to blink an LED for the number of times a user button is pressed. The LED blinks 'n' times when the button is pressed 'n' times. The value of 'n' (number of presses to blink) is stored in the program. The program also prints texts on a PC terminal via the serial UART2 port using Putty. Students will also control the LED by sending 1 for ON or 0 for OFF from a keyboard using the UART serial communication between the board and the PC. The UART2 settings are: 9600 baud rate, 8 data bits, no parity, 1 stop bit.

Part 1: Overview of Nucleo-F401RE board and STM32F401RE Microcontroller

The STM Nucleo-64 boards is an STM32 Microcontroller board featuring the STM32F401RE Microcontroller, Mbed support, Arduino™ Uno V3 connectivity support, and the ST morpho headers for the easy expansion of the functionality of the STM32 Nucleo open development platform with a wide choice of specialized shields. Nucleo-F401RE board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features. For the compatible boards, the external SMPS significantly reduces power consumption in Run mode. The STM32 Nucleo-64 board does not require any separate probe as it integrates the ST-LINK debugger/programmer. The STM32 Nucleo-64 board comes with the STM32 comprehensive free software libraries and examples available with the STM32Cube MCU Package, as well as direct access to the Arm® Mbed™ online resources at <http://mbed.org/>. ##### **1. Board Features:**



1. Codification: The table below shows the codes used to identify the features of Nucleo-boards

NUCLEO-XXYYRT NUCLEO-XXYYRT-P	Description	Example: NUCLEO-L452RE
XX	MCU series in STM32 Arm Cortex MCUs	STM32L4 Series
YY	MCU product line in the series	STM32L452
R	STM32 package pin count	64 pins
T	STM32 Flash memory size: <ul style="list-style-type: none">• 8 for 64 Kbytes• B for 128 Kbytes• C for 256 Kbytes• E for 512 Kbytes• G for 1 Mbyte• Z for 192 Kbytes	512 Kbytes
-P	STM32 has external SMPS function	No SMPS

2. STM32F401RE Microcontroller features

Peripherals		STM32F401xD			STM32F401xE		
Flash memory in Kbytes		384			512		
SRAM in Kbytes	System	96					
Timers	General-purpose	7					
	Advanced-control	1					
Communication interfaces	SPI/ I ² S	3/2 (full duplex)		4/2 (full duplex)	3/2 (full duplex)		4/2 (full duplex)
	I ² C	3					
	USART	3					
	SDIO	-	1		-	1	
USB OTG FS		1					
GPIOs		36	50	81	36	50	81
12-bit ADC		1					
Number of channels		10	16		10	16	
Maximum CPU frequency		84 MHz					
Operating voltage		1.7 to 3.6 V					
Operating temperatures		Ambient temperatures: -40 to +85 °C/-40 to +105 °C					
		Junction temperature: -40 to + 125 °C					
Package		WLCSP49 UFQFPN48	LQFP64	UFBGA100 LQFP100	WLCSP49 UFQFPN48	LQFP64	UFBGA100 LQFP100

3. Development tools

The board is supported by many tools and IDEs from many vendors.

- Keil Studio Cloud (free)
- ARM Development Studio (licensed)
- Arm® Mbed™ online (free) (see <http://mbed.org>)
- STM32CubeIDE (free)
- IAR Systems - IAR Embedded Workbench® (licensed)
- Keil®: MDK-ARM(a) (licensed)
- GCC-based IDEs
- CrossWorks
- SEGGER Embedded Studio (commercial with free-trial)
- SW4STM32 (free)
- TrueSTUDIO (free)
- MATLAB® and Simulink® (licensed)

4. Getting started

Follow the sequence below to configure the STM32 Nucleo board and launch the demo software:

a) Check the jumper position on the board, JP1 off, JP5 (PWR) on U5V, JP6 on (IDD), CN2 on (NUCLEO) selected.

b) Install ST_link USB driver and connect the board

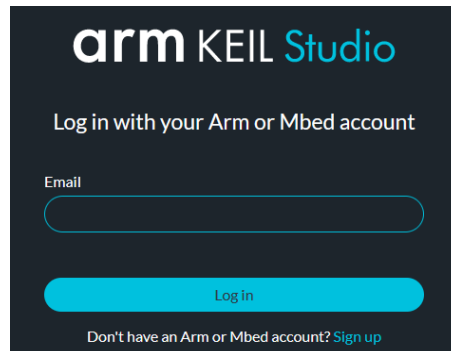
- The board is programmed via ST_link. Download and install the ST_link driver from the link provided below. Login/Registration is required to download the driver: <https://www.st.com/en/development-tools/stsw-link009.html>
- For this lab, the driver is already installed on the computers

Part 2: LED Blink Project Development

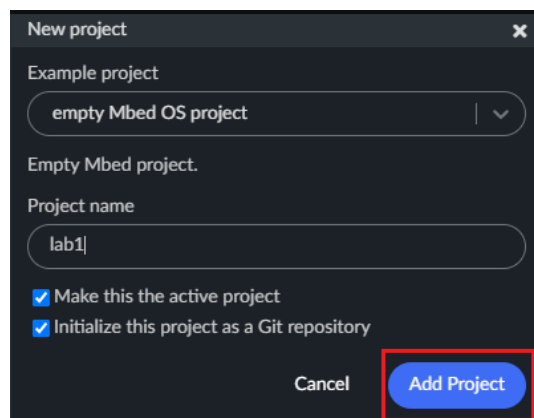
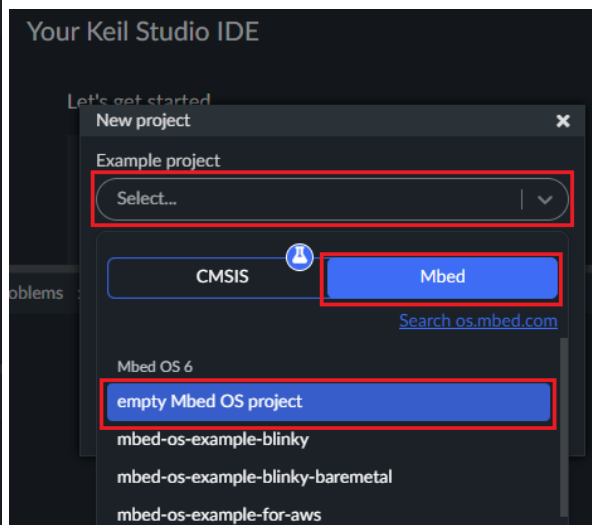
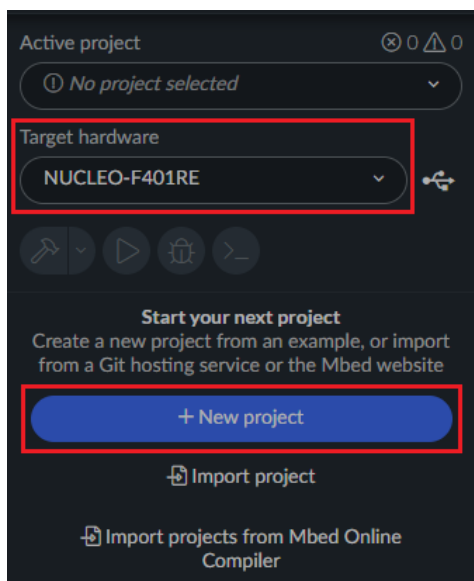
In this part, students are going to create a new project and write a program to blink an LED using the Keil Studio Cloud platform. Students are required to register an account with Mbed or the Keil for this task. Now follow the following steps to achieve the goal.

1. New Mbed project creation in Keil Studio Cloud

- Login to your registered ARM or Mbed account on the Keil Studio Cloud page <https://studio.keil.arm.com/auth/login/>

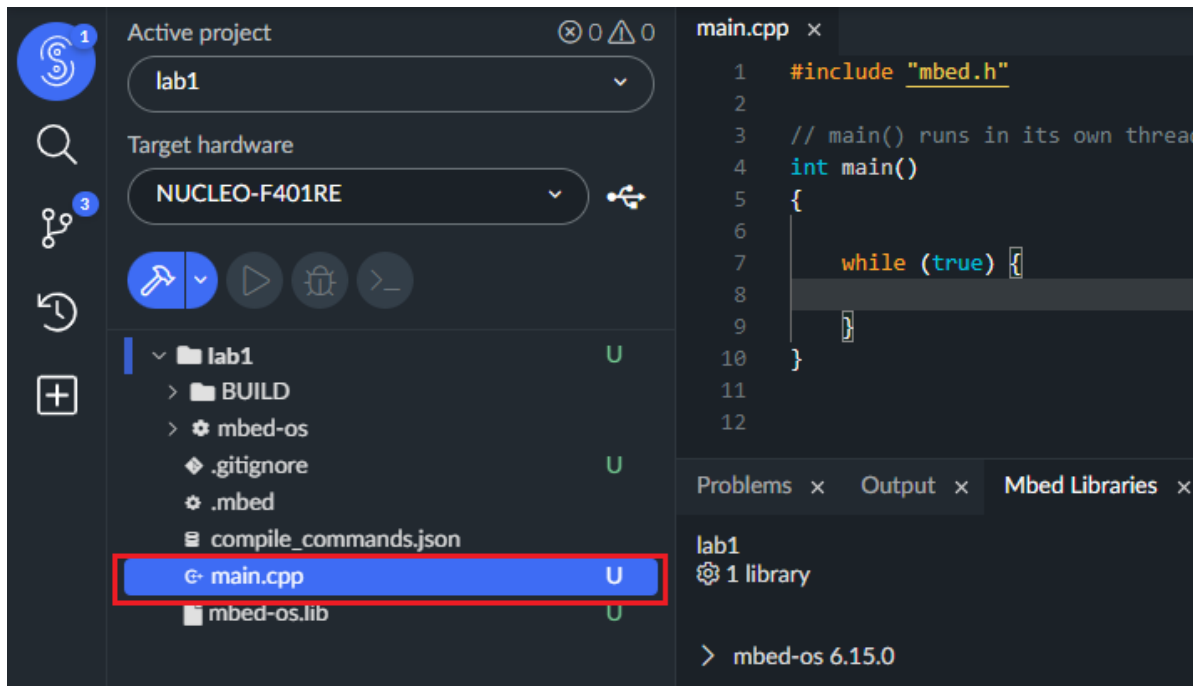


- Select **NUCLEO-F401RE** as the Target hardware
- Click **+New project** tab to create a new project
- Select Mbed and empty Mbed OS project as the example project.
- Name the new project as **lab1** and click **Add Project** button to create the project



2. Writing a C/C++ program to blink an LED

- Open the main.cpp file in the lab1 project



- Study and then copy and paste the code provided below

```
#include "mbed.h"


#define BLINKING_RATE 1000ms // Blinking rate in milliseconds

int main()
{
    DigitalOut led(LED1) ;

    while (true) {
        //Write your own code to blink the LED (led)
    }
}
```

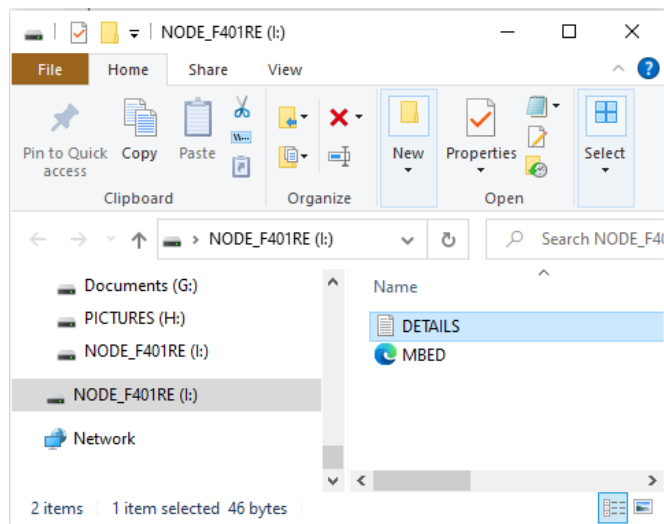
- Add your own code in the main function to blink the LED

3. Building/Compiling the project

- Click the *hammer icon*  in the Explorer Window to build the project.
- Once the build completes, save the generated **lab1.NUCLEO_F401RE.bin** file in a folder.

4. Running/Programming the program on the board

- Connect the board to the computer using the USB cable. Wait for the board to ON and finish initialization.
- You will notice a new drive NODE_F401RE created. The com port *STMicroelectronics ST Link Virtual Com Port* will also show in device manager. Notice the Port number.
- To access the Device manager go to Control Panel -> Device Manager
- To check the COM Port of the board, look for **Ports** in the device manager.



- Copy the **lab1.NUCLEO_F401RE.bin** to the board's folder.
- Now the project runs on the board Once the copy finishes
- Observe the blinking of the LED (LD2) on the board

Check Point 1:

Show your work to your lab supervisor. The supervisor will inspect your work and tick the checkpoint mark for you on the check point sheets. Proceed to the next tasks.

Part 3: Controlling LED with Push Button and Serial Communication

1. Controlling LED with Push Button

- Create a new project lab1_LED_control
- Open the main.cpp file
- Copy and paste the code provided below:

```
#include "mbed.h"

#define BLINKING_RATE    1000ms // Blinking rate in milliseconds
#define DEBOUNCE_DELAY  400ms  // Button debounce delay
#define PRESSES_TO_BLINK 3      // Button debounce delay

int main()
{
    // Initialise the digital pins
    DigitalOut led(LED1);        // LED1 set as an output
    DigitalIn  button(BUTTON1); // BUTTON1 set as an input

    int pressed;                 // variable declaration

    pressed = 0;
    led = 0;

    printf("Press the button %d times to blink the LED !!!\r\n",
PRESSES_TO_BLINK);
```

```

while (true) {

    // Detect a button press
    while(button.read());
    ThisThread::sleep_for(DEBOUNCE_DELAY);
    pressed +=1 ;

    // Check if to blink
    if(pressed >= PRESSES_TO_BLINK){

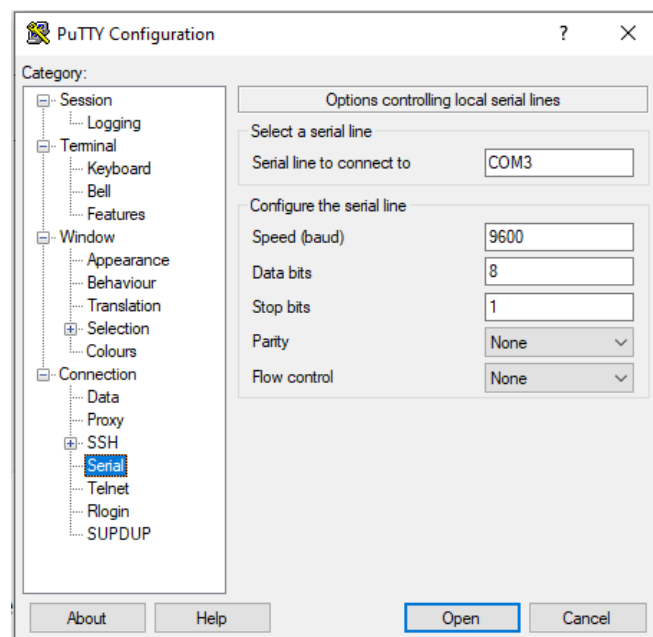
        printf("LED will blink %d times !!\r\n", PRESSES_TO_BLINK);

        // blink LED if the required presses done
        while (pressed) {
            led = 1 ;
            ThisThread::sleep_for(BLINKING_RATE);
            led = 0 ;
            ThisThread::sleep_for(BLINKING_RATE);
            pressed -=1 ;
        }

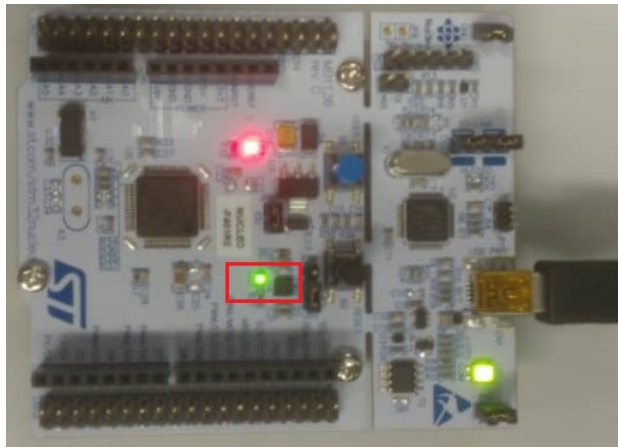
        printf("Press the button %d times to blink the LED !!!\r\n",
PRESSES_TO_BLINK);
    }
}
}

```

- Build and save the generated .bin file.
- Open the device manager to know the COM port of the STLink
- Open Putty and set the serial connection.



- Copy the *lab1_LED_control_NUCLEO_F401RE.bin* to the board drive folder.
- Observe the Putty terminal and also press the buttons as requested.



Check Point 2:

Show your work to your lab supervisor. The supervisor will inspect your work and tick the checkpoint mark for you on the check point sheets. Proceed to the next tasks.

2. Controlling LED with serial communication

- Now modify the program with the code below

```
#include "mbed.h"

static BufferedSerial pc(USBTX, USBRX);

int main(){

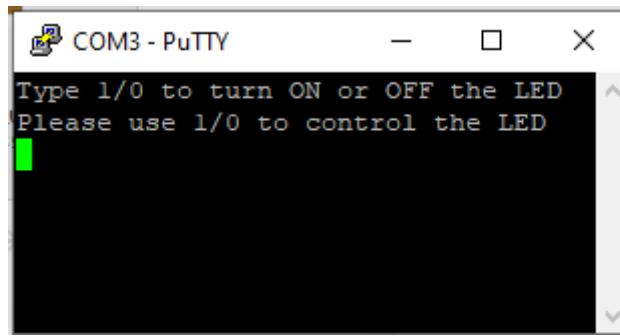
    DigitalOut led(LED1); //set LED1 as an output

    // variable declarations
    char msg1[] = "Type 1/0 to turn ON or OFF the LED \r\n";
    char msg2[] = "Please use 1/0 to control the LED \r\n";
    char command;
    pc.write(msg1, sizeof(msg1));    // print on the console

    while (true)    // infinite loop
    {
        pc.read(&command, 1); //read the keyboard

        if(command == '1'){
            led = 1; // ON LED if 1 is received
        }else if(command == '0'){
            led = 0; // OFF LED if 0 is received
        }else{
            pc.write(msg2, sizeof(msg2));
        }
    }
}
```

- Open Putty serial terminal
- Build and run the program on the board




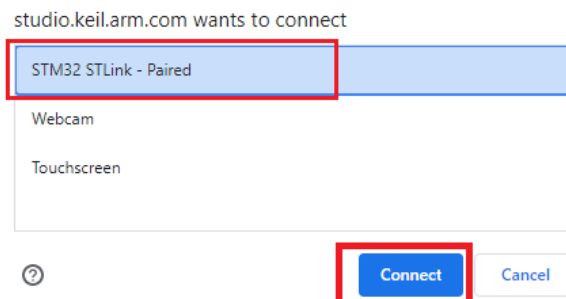
Check Point 3:

Show your work to your lab supervisor. The supervisor will inspect your work and tick the checkpoint mark for you on the check point sheets. Proceed to the next tasks.

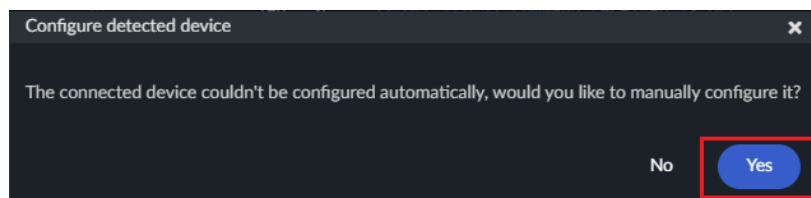
Part 4: Debugging with the Keil Studio Cloud

1. Setting up the debug

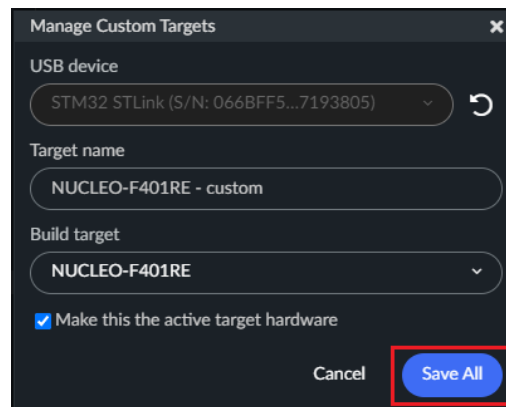
- Connect the board over USB
- Click the connect to target hardware button  to the right of the *Target hardware* drop-down list.
- Select *STM32 STLINK*, after the first successful connection, Keil Studio detects the board and suggests a matching target.




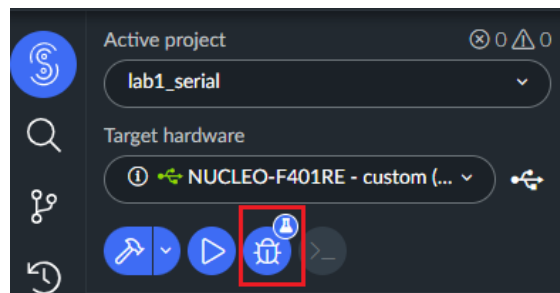
- For *NUCLEO-F401RE* board, you need to setup the debug manually,



- Select the *NUCLEO-F401RE* as the build target, the target name will automatically set. Click Save All

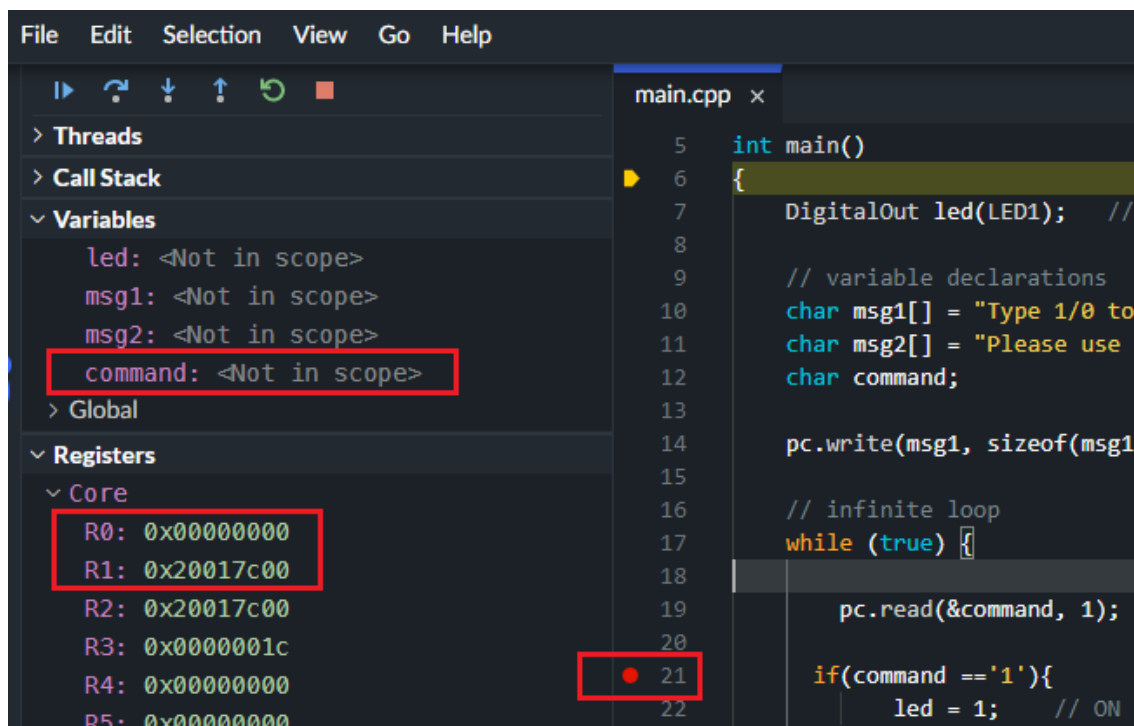


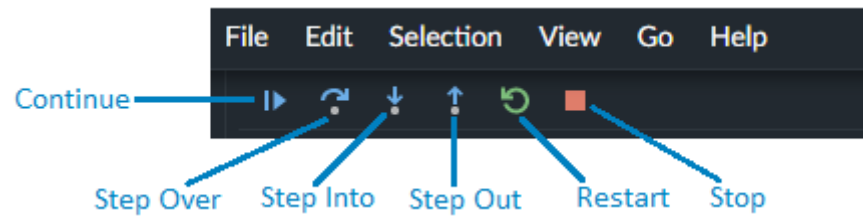
- Click the debug button, . Keil Studio automatically builds and flashes your project to the connected board. The Debug view displays and the debug session starts. The debugger stops at the function "main" of your application



1. Debugging the lab1 project

- Put a breakpoint at line **21** by clicking the margin to the left of the line of code in the editor. A red dot indicates the breakpoint. You can also right-click the margin and select *Add Breakpoint*.
- Click **Variables** and **Registers->Core** from the left side panel.
- Observe the values of the **command** variable and the **R0** and **R1** Core registers.





Continue: The debugger continues to run until a breakpoint is hit.

Step Over: Advances the debugger to the next source line that follows in the program .

Step Into: Advances the debugger into each function breaking on the first line of the function.

Step Out: Advances the debugger until the current function returns.

- Click **Continue** once, the debugger run the program up to the breakpoint
- Open the terminal and press 1, observe the value change in the variables and the registers
- Click the **Step Over** twice and observe the LED turning ON

Check Point 4:

Repeat the debugging and now OFF the LED instead. Observe any change and show your work to the lab supervisor.

~ END ~