**MECHTRON 2TA4: Laboratory 1**

**Introduction**

This assignment will introduce you to the software and hardware development tools you will use during the semester to work with the STM32F429IDiscovery development board and the Cortex-M4 ARM Processor so that you may run programs using the Keil uVision SDK. Different parts of this lab will show you:

- How to generate configuration and initialization code for a project using STM32CubeMX tool

- How to develop an application using Keil uVision IDE

- How to use timer interrupts to interface with the LEDs in the STM32F429-Discovery board

- How to get input from the "user push-button" on the board.

**Recommended Reading**

* 1. Familiarize yourself with the embedded C language. Specifically, you need to concentrate on the following material:
     + Reading Chapters 1, 2, 14 of "Embedded Systems with ARM Cortex-M Microcontrollers in Assembly and C" by Yifeng Zhu.
  2. Familiarize yourself with the architecture of the Cortex-M4 processor by
     + reading the Cortex-M4 reference manual [online](http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0439b/CHDDIGAC.html).
  3. Familiarize yourself with the structure of the STM32F429-Discovery board.
     + The STM32F429-Discovery Board User Guide found  [here](https://avenue.cllmcmaster.ca/content/enforced/515521-MECHTRON_2TA4_bokhari_2231/manuals/um1670-discovery.pdf?_&d2lSessionVal=y5SXbJBpqWofTRKQJwAv7dBjL&ou=515521)
  4. Familiarize yourself with the timers of the Cortex-M4 in the STM32F429-Discovery Board.
     + Read corresponding chapters of the reference manual here: [RM0090.](https://avenue.cllmcmaster.ca/content/enforced/515521-MECHTRON_2TA4_bokhari_2231/manuals/RM0090.pdf?_&d2lSessionVal=y5SXbJBpqWofTRKQJwAv7dBjL&ou=515521)
  5. Familiarize yourself with documents posted in module named "Reference Material".

**Hardware**

**STM32F429-Discovery**

The STM32F4-Discovery board used in the lab provides:

* + - A target STM32F429 microcontroller with onboard flash memory.
    - An embedded ST-LINK/V2 interface to program the flash memory through a mini-USB connection.
    - A separate micro-USB connection for applications.
    - One user push-button, one reset button and two user programable LEDs.
    - More information is found [here.](https://avenue.cllmcmaster.ca/content/enforced/515521-MECHTRON_2TA4_bokhari_2231/manuals/um1670-discovery.pdf?_&d2lSessionVal=y5SXbJBpqWofTRKQJwAv7dBjL&ou=515521)

For this lab you do not need to use additional connectors except for the mini-USB.

The STM32F429-Discovery is powered via the mini-USB connection and no external power supply will be needed although this option is available. When using the mini-USB connection to power the board you must be very careful that any other peripherals that are using either the 3V or 5V output do not exceed 100mA otherwise you will burn the mini-USB interface on the board.

**Software**

The software you will use consists of:

* + - STM32CubeMX graphical tool (installed by you in lab 0).
    - Keil uVision 5 IDE (installed by you in lab 0). where you will develop and compile your software.
    - Stm32F429-Discovery firmware package (installed by you in lab 0).
    - ST-LINK/V2 link utility and USB drivers installed by you in lab 0)

**General Procedure**

* + 1. Connect your stm32f4-dicovery board to a USB port of your choice on the host computer. Let windows try to find the drivers. If the drivers are not found then the st-link usb drivers are missing and you will need assistance from the TAs.
    2. This assignment consist of 3 parts. Create a separate project for each part. Name them, “lab1-1”, “lab1-2” and “lab1-3” respectively.
    3. Each project uses both the STM32CubeMX tool and uVision 5 IDE, previously installed (Ref lab0). For each project, follow the steps detailed in the document named, **“Getting Started 2023.docx”** used by you in lab0. The steps are briefly stated below:
    4. Open the STM32cubeMX software and click New Project.
    5. Select the board in the “Board Selector” tab.

A small dialog box prompts you for accepting to initialize all peripherals to their default mode.

Selecting a board and accepting to initialize all peripherals to their default mode automatically sets both the pinout and the default modes for the peripherals available on the board. This means that STM32CubeMX generates the C initialization code for all the peripherals available on the board and not only for those relevant to the user application.

* + 1. For all projects of this lab say "yes" to the prompt in item 5 above. This may not be the case for rest of the labs.
    2. The project page opens, showing the following set of views:
       - Pinout & Configuration
       - Clock Configuration
       - Project Manager
       - Tools
    3. Users can move across the different views without impacting their project configuration.
    4. Complete configuration of your project as follows:
       - In the left panel, under Categories of Middleware, select FREERTOS, in the middle panel of “FREERTOS Mode and Configuration”, under the “Mode” section, set “Interface” to Disable.
       - Also for the Middleware of “USB\_HOST”, set its “Class for “HS IP” to Disable.
       - Configure RCC
       - Configure GPIO pins
       - Configure clocks
       - Configure peripherals to be used in the application according to requirements.
       - Use Project Manager to give name to the project and a suitable location.
    5. Click “GENERATE CODE” menu item near the top right corner of the page. The project will be created with initialization and configuration C code.
    6. **Close STM32CubeMX tool.**
    7. Find the shortcut to the Keil uVision IDE on the host computer. Start uVision.
    8. Select "Project->Open Project" and select the ".uvproj" file in the folder that was specified for the location when the project was created in STM32CubeMX. This will load the project in your Keil uVision IDE.
    9. Make sure the project configuration is correct (Ref. **“Getting Started 2023.docx”**)
    10. Edit main.c and stm32f4xx\_it.c files to add application specific code.
    11. Compile your lab by selecting "Project->Build" or by pressing F7 and make sure there are no errors in compilation. If errors are present make sure to carefully investigate what is missing (ie. header or source files at compile-time, missing dependencies or implicit declarations at linker-time and so forth) before asking assistance from a TA.
    12. Load the HEX file produced by the toolchain onto your board by selecting "Flash->Download"
    13. Press the "Reset" button on the STM32F4-Discovery board.
    14. You should see the relevant LED blink in desired order.
    15. If you would like to Debug your code step by step select "Debug->Start/Stop Debug Session" or press Ctrl+F5.

**Important:** For more thorough documentation on using Keil uVision go to your Keil uVision IDE and click the "Books" tabs on the bottom left. This tabs prvides you with all the resources and information needed to use Keil uVision as well as information on programming with Cortex processors.

Timing of all functions in this lab, ***and most exercises in this course*** will be handled by interrupt-driven counters, not by software wait-loops.

**Lab1-1**

In lab0 LED3 was made to blink, however it had two issues: 1) It used "busy wait" that wastes CPU time. 2) The time delay implemented by the function "delay\_ms" does not create an exact time delay.

An exact delay can be implemented by using a timer that generates an interrupt after a specified interval. The interrupt represents an exact time interval. The system timer, sometimes called Systick is the easiest source for this purpose. The HAL\_SysTick\_Config() function, calls the SysTick\_Config() functionn which is a CMSIS function that:

* + - Configures the SysTick Reload register with value passed as function parameter.
    - Configures the SysTick IRQ priority to the lowest value 0x0F.
    - Resets the SysTick Counter register.
    - Configures the SysTick Counter clock source to be Core Clock Source (HCLK).
    - Enables the SysTick Interrupt.
    - Starts the SysTick Counter.

Launch the STM32CubeMX tool and create a new project lab1-1. Configure the project as explained in the general procedure above and close the tool.

Open the project in uVision 5.

Find a commented line in main.c with phrase, "USER CODE BEGIN Sysinit" and add the following line of code:

HAL\_SYSTICK\_Config(HAL\_RCC\_GetHCLKFreq()/1000);

Although the current HAL library provides a function named HAL\_Delay(), you are required to create your own function named Delay() that creates a delay in milliseconds using the SysTick interrupt handler, (for a better understanding). Proceed as follows:

Declare a global variable **volatile uint32 msTicks**. Find SysTick\_Handler in stm32ff4xx\_it.c and add just one statement: msTicks++;

Note that SysTick will generate an interrupt every millisecond, resulting in incrementing of msTicks. This variable can now be used in a delay function to create an exact delay in milliseconds.

void Delay(int millis){ while(msTicks < millis); msTicks = 0; }

Finally add the following code in the infinite while loop within main() function:

while(1) { HAL\_GPIO\_TogglePin(GPIOG, GPIO\_PIN\_13); Delay(500); }

Compile and debug the project, if necessary. While compiling, you may get an error message complaining, "Default Compiler Version 5 is not available". In this case go to "Options for Target" on the tool bar, click the Target tab and set the "ARM Compiler" to default compiler version 6.

Load the HEX file into MCU Flash. Push RESET button and show the result to your TA.

**Lab1-2**

Lab1-1 is an improvement over lab0, however it still wastes CPU time. In this part of the lab TIM3 and TIM4 will be used to generate interrupts of suitable intervals and the relevant interrupt handlers will toggle pins 13 and 14 of GPIOG to blink both LED3 and LED4.

Launch the STM32CubeMX tool and create a new project lab1-2. Configure the project as explained in **“Getting Started 2023.docx”**, however in this case you **need to configure TIM3 and TIM4 separately**as detailed in this document.

Notice the GPIO pins PG13 and PG14, which are connected to LED3 and LED4 on the STM32F429I-DISC board, have already been initialized by STM32CubeMX so we do not need to do any changes here for this project.

Give a Project Name, Choose a project location and for Toolchain/IDE, choose “MDK-ARM”

GENERATE CODE

**Close STM32CubeMX tool.**

Open the project in uVision 5

* + - Open the file main.c, find the function, 'static void MX\_TIM3\_Init(void)'. At the end of this function, add a line to start the timer:

HAL\_TIM\_Base\_Start\_IT(&htim3);

* + - In main.c, find the function, 'static void MX\_TIM4\_Init(void)'. At the end of this function, add a line to start TIM4:

HAL\_TIM\_OC\_Start\_IT(&htim4, TIM\_CHANNEL\_1);

* + - Find the function, 'void HAL\_TIM\_periodElapsedCallback(TIM\_HandleTypeDef \*htim)'. At the end of this function, add the following lines:

If (htim->Instance==TIM3) {

HAL\_GPIO\_TogglePin(GPIOG, GPIO\_PIN\_13);}

* + - Add a function,

void HAL\_TIM\_OC\_DelayElapsedCallback(TIM\_HandleTypeDef \*htim) {

HAL\_GPIO\_TogglePin(GPIOG, GPIO\_PIN\_14);

\_\_HAL\_TIM\_SET\_COUNTER(htim, 0X0000);}

* + - Then the project can be build and downloaded to the board.
    - After the binary is downloaded to the board, press the RESET button, the LED3 will blink every second, and LED4 will blink every 2 seconds.
    - Show the project results to your TA.

**Lab1-3**

Launch the STM32CubeMX tool and create a new project lab1-3. Configure the project as explained in the general procedure and in the project lab1-2 above.Generate the code then close the tool.

**HINT: GPIO pin PA0 must be configured for this project.**

Open the project in uVision 5 and modify the code such that LED3 and LED4 blink as done in project lab1-2 but only as long as the user button is kept pressed.

* + - Then the project can be build and downloaded to the board.
    - After the binary is downloaded to the board, press the RESET button, the LED3 will blink every second, and LED4 will blink every 2 seconds, **only when the user button is pressed.**
    - Show the project results to your TA.

**Grading**

* + 1. Completion of prelab (lab0 project)**[20 pts]**
    2. Completion of lab1-1**[20 pts]**
    3. Completion of lab1-2**[40 pts]**
    4. Completion of lab1-3**[20 pts]**

**There is no written report for Lab 1, but attendance is mandatory and you will be required to demo your program to the TA. Be prepared to answer questions about your work and to show your code.**