MECHTRON 3TA4: 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 STM32L476Discovery development board and the Cortex-M4 ARM Processor so that you may run programs using the Keil uVision SDK. The example code provided with this assignment shows you:

- how to setup a project using Keil uVision;

- how to use timer interrupts to interface with the LEDs in the STM32L476-Discovery board using the "user push-button" on the board.

Prelab

1. Familiarize yourself with the embedded C language and the STM32L4 Standard Peripheral Library that comes with your STM32L4-Discovery software. Specifically, you will 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.
    - The STM32L4 Low Layer drivers manual found here.
2. Familiarize yourself with the architecture of the Cortex-M4 processor by
    - the Cortex-M4 provides a superset of the Cortex-M3 instruction set.
    - reading the Cortex-M4 reference manual online.
3. Familiarize yourself with the structure of the STM32L476-Discovery board.
    - The STM32L476-Discovery Board User Guide found on avenue under content/ Manuals & DataSheets/en.DM00172179.
4. Familiarize yourself with the timers of the Cortex-M4 in the STM32L476-Discovery Board.
    - Read corresponding chapters of the reference manual here: DM00083560.pdf.
5. Familiarize yourself with the use of Keil's development tools by working through the first part of the PDF into together with Zip file of code below:

    - CortexTM-M4 Training STM32L476 Discovery evaluation board using ARM® Keil MDK Toolkit: found on avenue under content/ Manuals & DataSheets/en.DM00217936
    - https://www.st.com/content/ccc/resource/technical/document/user_manual/40/8a/04/4b/cf/2a/4c/73/DM00213619.pdf/files/DM00213619.pdf/jcr:content/translations/en.DM00213619.pdf
    - https://www.st.com/content/ccc/resource/technical/document/user_manual/74/09/3d/80/f9/39/4c/c7/DM00157440.pdf/files/DM00157440.pdf/jcr:content/translations/en.DM00157440.pdf

- https://pdfs.semanticscholar.org/presentation/0a9a/2ab2b2acb501d40e8b2b096ddc22f7fe7192.pdf
- Explore some project files in the examples directory in your firmware folder.

Hardware

STM32L476-Discovery

The STM32L4-Discovery board you will be using to support the Cortex-M4 processor is a small board providing:

- A target STM32L476 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 reset button, two user LEDs (one is green and one is red), and a four-direction joy stick with selection.

For this lab you will not need to use additional connectors except for the mini-USB, however, in the rest of the labs you will need to use header pins to connect to external circuits built on breadboards.

The STM32L476-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 are:

- Keil uVision where you will develop and compile your software.
- Stm32L476-Discovery firmware package
- ST-LINK/V2 link utility and USB drivers.

Procedure

1. Login to the PC at your lab station.
2. Download the Lab1 project from avenue under content/labs/lab1 .
3. Extract the lab1 starter project into your working folder. Make sure it is in a folder called lab1.

4. Connect your stm32l4-dicovery board to a USB port of your choice. 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. For the lab computers, all the drivers are ready.
5. Find the shortcut to the Keil uVision IDE. Start uVision.
6. Select "Project | Open Project" and select the ".uvproj" file in your starter project. This will load the starter project in your Keil uVision IDE.
7. Make sure the project configuration is correct:
    - In the project explorer in KEIL uVision, right click the target (the target may be renamed as “stm32L476DISCO” or other names), and select "Options for Target …" or select "Project | Options for Target…", or select “Options for Target…” icon on the tool bar.
    - Under the "Device" tab make sure the device selected is "STM32L476VGTx" as printed on your discovery board.
    - Under the "Target" tab make sure "Use MicroLIB" has a checkmark.
    - Under the "Output" tab make sure "Create HEX File" has a checkmark.
    - Under the "C/C++" tab
        - Make sure the "Define" box has “STM32L476xx,USE_HAL_DRIVER,USE_STM32L476G_DISCO_REVC” in it (without the quotes). This defines appropriate macros for the Cube HAL libraries for STM32L476-Discovery boards used in our labs, so to enable the correct definitions in the library and appropriate library functions will be used.
        - Make sure "One ELF Section per Function" is checked. Since we are using a free version of uVision which has a code size limit, to minimize the possibility for the project code size exceeds the limit, it is better to set the Optimization level as “Level 3”.
        - Make sure your include paths are correct. Our starter kit assumes the library folder, “STM32Cube_FW_L4_V1.8.0”, is a sibling folder to your lab project (the two are on the same level), so by default your project’s include paths are:

          .\inc
          ..\STM32Cube_FW_L4_V1.8.0\Drivers\CMSIS\Device\ST\STM32L4xx\Include
          ..\STM32Cube_FW_L4_V1.8.0\Drivers\STM32L4xx_HAL_Driver\Inc
          ..\STM32Cube_FW_L4_V1.8.0\Drivers\BSP\STM32L476G-Discovery
          ..\STM32Cube_FW_L4_V1.8.0\Drivers\BSP\Components\Common
          ..\STM32Cube_FW_L4_V1.8.0\Drivers\CMSIS\Include

          however, if you have a different folder structure, you will need to update this setting to the appropriate folders.

    - Under the "Linker" tab make sure that "Use Memory Layout from Target Dialog" and "report 'might fail' conditions as errors" are both checked.

- Under the "Debug" tab, make sure there's a checkmark besides "Run to main()". Make sure to select "ST-Link Debugger" in the dropdown box beside "Use". Then click the button "Settings" beside this dropdown box to open the “Cortex-M Target Driver Setup” window. In this window, under the Debug tab, set Port as “SW” to make sure "SW" (single-wire debug) is selected as the debugging interface; set Connect as “Under Reset” instead of the default setting “Normal”, to avoid troubles when first time download code to flash memory. Under the tab “"Flash Download", check that "STM32L4xx Flash" is the only listed item.
- Under the "Utilities" tab make sure "Use Target Driver for Flash Programming" is selected.

8. If the source files appearing in your "Workspace" window are not found, make sure you placed the files in the appropriate directories.
9. Take special note of the startup_stm32l4xx.s assembly file. This file defines the startup procedure before getting to your main() function as well as defines all the available interrupts. This can come in handy when you want to quickly find the name of the function for a specific interrupt (eg. TIM3_IRQHandler, EXT2_IRQHandler, etc.).
10. In the “src” folder of the lab starter project, please pay attention to the following three C files:
    - Main.c: This is the main user application file. Among many other user tasks and functions, in main.c, users can call HAL_PPP_Init() or HAL_PPP_DeInit() to do initialization or de-initialization for peripherals or hardware such as RCC, GPIO, NVIC or DMA, can call HAL_PPP_Start_IT() to start interrupt.
    - stm32l4xx_it.c: This is the source files contains interrupt handlers. Each of these interrupt handlers contains a function HAL_PPP_IRQHandler(), which in turn call a HaL_PPP_Callback() function, which requires user to implement user functions in main.c.
    - stm32l4xx_msp.c: The peripheral initialization or de-initialization is done by calling HAL_PPP_Init() or HAL_PPP_DeInit() in main.c, but the low level initialization or de-initialization of these peripherals (PPP) are performed by calling MSP callback functions HAL_PPP_MspInit() or HAL_PPP_MspDeInit(). The stm32l4xx_msp.c is the file that contains these MSP (MCU Specific Pack) initialization and de-initialization callback functions.

11. Compile your lab project by selecting "Project->Build", or by pressing F7, or by clicking the Build icon on the tool bar, 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", or by pressing F8, or by selecting Download icon on the tool bar.
13. Press the "Reset" button on the STM32L4-Discovery board, you should see the red LED4 blinks, and “MT3TA4 Lab1 Starter” will scroll on LCD for two times.
14. Press the "Selection" of the joystick (the middle of the joystick) on the STM32L476-Discovery board and you should see LED4 blinking at a different rate.

15. Pressing each direction of the joystick will fire an interrupt to make the pressed joystick's direction displayed on the LCD.
16. 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 of the project explorer window, or go to Help | Open Books Window. This books window provides 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.

Assignment

Modify the downloaded code to accomplish the followings:

1. **[15 pts]** Make the greenLED5 blink instead of LED4 when the reset button is pressed (ie. at the start of your program).
2. **[55 pts]** Pressing the Selection of the joystick to make the two LEDs blink in a turn-taking manner: LED4 on, LED4 off, LED5 on and then LED5 off, and then repeat. Also, modify the code so that they blink at 1Hz, i.e. for each LED the on-time and off-time should add up to 1 second, and it takes 2 seconds in total to blink both LEDs once.
3. **[30 pts]** Use TIM2 instead of TIM3.

Hint: Look through the comments in the C code and also through the TIM_TimeBase example of the STM32L4-Discovery board TIM examples.

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.