Purdue University
School of Mechanical Engineering

ME 586: Microprocessors in Electromechanical Systems — **Fall Semester 2019** — **Dr. Peter Meckl**

Laboratory Assignment #1

OBJECTIVES

1) Acquainting yourself with the PC
2) Learning the concept of modular programming
3) Editing, building, and debugging code in the Keil µVision Integrated Development Environment (IDE)

REFERENCES

GENERAL PROCEDURES

1) To allow you to obtain hard copy printouts in ME 586 lab, you must have a valid ECN/ME account. If you do not have one, please see your lab instructor or go to room ME 2042 (Technical Services) and ask to set up an account.

2) Familiarize yourself with all the components of the HP PC. At the prompt, provide your login and password to start a Windows 10 session. Use the Tab key to go from the login field to the password field. When you log in, your ECN account will automatically be mounted as drive R: on your PC.

   Thus, any files that you have created on tools or widget (depending on where your account resides) will automatically be accessible on drive R: from your PC. In addition, drive X: will allow you to access files from the instructor's computer. This drive is read-only.

   If you cannot access drive X:, click Windows Explorer (next to the Start menu on the bottom left corner of the screen.) Click on Computer, and then Map Network Drive. In the Folder input, type \\tools.ecn.purdue.edu\me586lab, and hit enter on the keyboard. This should map the network drive.

3) In order to set aside space to store your files, you will use D: drive as your working drive. We suggest that you create directories for each lab assignment, that is, create Lab01 for the first lab, Lab02 for the second lab and so on, to store your files for that lab. To create these directories, click on Windows Explorer at the bottom left corner of the screen. Then, click drive D. Click on New Folder. The new folder appears with a temporary name. Type a name for the new folder, i.e. Lab01, and then press Enter from the keyboard. Your final directory tree should look like this:

```
D:\LAB01
  \LAB02
  \LAB03
      :
```

   You may want to create a similar directory structure on your home directory.

4) In this lab, you will acquaint yourself with the Keil µVision Integrated Development Environment (IDE). You will use previously created files that you will edit to generate working programs. The source code for your assembly program is contained in the Lab01 directory on the X: drive. The assembly file `'LEDCount.s'` contains code to count and display the output on the LEDs on the box.

   To copy these files from the X: drive into your PC directory, use Windows Explorer. Click X: drive to see the contents of the drive. You will see the Lab01 subdirectory. Double click the Lab01 subdirectory. You will see two folders. The folder labeled 'Assembly' should contain two assembly code files (.s) which are `'initports.s'` and `'LEDCount.s'`, and two project files related to the project configuration, `'Lab01.uvoptx'` and `'Lab01.uvprojx'`. The folder labeled 'C' should contain a C-code file

(‘Lab1.c’) and two project configuration related files, ‘CLab01.uvoptx’ and ‘CLab01.uvprojx’. Copy all files in X:\Lab01 into the desired directory on the D: drive. First, we will work with the assembly portion. The C files will be used later.

ARM Cortex-M3 Assembly Programming

1) This lab will introduce you to the basics of creating, compiling, linking and debugging your files using Keil µVision IDE. To start µVision, double click the Project file ‘Lab01.uvprojx’ in your working directory. A window where you can edit, compile, link and debug your files appears. On the left, you will see a project tree with AsmTemplate as the root. This is your Target. Underneath, you will see a list of files that make up the project. The folder *Support* contains the initialization code for the microcontroller, which you will not need to modify. The folder *User* contains the main code of your project that you will modify or create.

2) First, you need to check that the project contains the correct paths to the assembly files. Find the individual assembly (.s) files in the project tree, and right click on them to bring up a context menu. Select Options for File... and make sure the files are the correct ones (in your directory, not the instructor’s.) If they are not, remove those files from the project by right clicking on them and selecting Remove. Then go to the User folder, right click and select Add Existing Files... and select the correct ones.

3) Double clicking on each file name in the project tree will open that file for editing. The files ‘LedCount.s’ and ‘initports.s’ are incomplete. Please edit these files so that they look exactly like what is shown below.

LEDCount.s:

```
GPIOB_ODR       EQU 0x40010C0C
LEDDELAY        EQU 0x2FFFFF
MAXVAL          EQU     8192

                AREA    ARMex, CODE, READONLY
                ENTRY
__main PROC
                EXPORT  __main
                IMPORT initports

        bl initports

top     mov r2, #31             ; Load R2 with count starting value
        ldr r6, =GPIOB_ODR      ; load r6 with Port B output data register address
loop
        str r2, [r6]            ; send current value of counter out Port B
        ldr r1, =LEDDELAY       ; delay so we can see the count
delay1
        subs r1, #1             ; subtract 1 from delay value
         nop                    ; do nothing
        bne delay1              ; loop until you reach zero

         add r2,#32             ; increment count

         cmp r2,#MAXVAL         ; compare counter to maximum count
         bgt top                ; if it's greater than MAXVAL, you're done, start over
         b loop                 ; else display next value

         ENDP
         END
```

initports.s:

```
GPIOC_CRH       EQU   0x40011004
GPIOC_ODR       EQU   0x4001100C

GPIOB_CRL       EQU   0x40010C00
GPIOB_CRH       EQU   0x40010C04
GPIOB_ODR       EQU   0x40010C0C

RCC_APB2ENR     EQU   0x40021018

                AREA    ARMex, CODE, READONLY
initports    PROC
                EXPORT  initports

        push {lr}           ;store link register on stack
        ;; Enable the Port B peripheral clock
        ldr r6, =RCC_APB2ENR
        mov r0, #0x08
        str r0, [r6]

        ;; Set the config and mode bits for Port B bits 5-12 so they will
        ;; be a push-pull output (up to 50 MHz) by setting bits 19-16
        ;; to '0011'.

        ldr r6, =GPIOB_CRL
        ldr r0, =0x33300000  ;configure PB5-PB7 as outputs
        str r0, [r6]

        ldr r6, =GPIOB_CRH
        ldr r0, =0x00033333  ;configure PB8-PB12 as outputs
        str r0, [r6]

        pop {lr}             ;get link value
        bx lr                ;return to calling program

                ENDP
                END
```

5) Save both of your files when you are finished.

Compiling and Linking

6) Each project has a specific set of compiler and linker options associated with it. Right click on the Target folder in the Project window (near top of the tree, under the Project name) and select Options…. Click through each of the tabs and familiarize yourself with the settings (These should be preconfigured and you won't need to change anything).

   Click OK button when finished.

7) To compile and build your project, select Project…Rebuild. The results of building your project will be shown in the Build Output window.
   a) If compiling was successful (no errors), proceed with downloading the program. Warnings are OK.
   b) If compiling was not successful, you should fix the errors. Double click on the error statement in the build window. This will take you to the line in your program where the compiler registered an error.

8) Plug in the power cord for the STM32 box, and connect the two USB connectors to two USB ports on the computer. One USB port is for programming and debugging the microcontroller, and the other is for serial communication with the microcontroller (not used in this exercise). Make sure to turn on the power switch on the back of the STM32 microcontroller box before plugging the USB cables into the computer. This will allow the computer to recognize a new device has been connected.

9) After successfully building your program, download it to the STM32 by clicking Flash…Download. You will see a progress bar at the bottom left corner of the window indicating that the STM32 has successfully been erased, programmed and verified.

   To run the program, press the reset button on the STM32. Observe what happens to the LEDs.

10) Experiment with changing the values of INCREMENT, MAXVAL, and LEDDELAY. Rebuild and download your project and observe any changes in the code.

11) Once you have working code, you can make a hard copy of your file by selecting File…Print. For this lab you will not need to turn in any hard copies.

12) When finished, select Project…Close Project. This will close the project, but leave µVision IDE running for the C programming exercise below.

13) You should also make a copy of your files in your own ECN directory. To perform this task, use Windows Explorer. Then "drag" your files to the appropriate directory on drive R: (your home account).

C Programming

1) This portion of the lab will introduce you to the basics of creating, compiling, linking and debugging C programs using Keil µVision IDE. Open the `CLab01.uvprojx` file you copied from drive X:. Double click on `Lab1.c` in the project tree to open this file. An editing window opens, allowing you to make any required changes to the code. Note that a portion of the C code is missing. Please insert code in the main() function that will count from 0 to 255 and display the results on the LEDs. The following functions will be useful:

initports(0x0);	Initialize all ports as Outputs
digout(i);	Output value i
__nop();	No Operation, an assembly instruction that does nothing but use up processor time for a small delay. Note that you will need to repeat this instruction multiple times to achieve a significant delay between counts in your program to be able to see the LED values change.

2) When you have a final version of the code, it's time to check the options as before. Again, familiarize yourself with the available options, but you do not need to change any of them at this time.

3) Now it's time to build. Select the Rebuild option in the Project window to compile your C code. If you get a successful compilation, you can go on to the next step. Otherwise, your error messages will appear in the Build Output window. Double clicking on each error will place the cursor at the appropriate offending line in the C code. Use this information to fix any syntax errors.

4) Once you have successfully built your code, download it to the STM32 and run it like you did with the assembly version.

5) You can print out a copy of the functioning C code using the Print command under the File menu. Then exit the program. Also make a copy of the `*.c` file in drive R:. . For this lab you will not need to turn in any hard copies.

6) Please delete all the files you created on the D: drive.