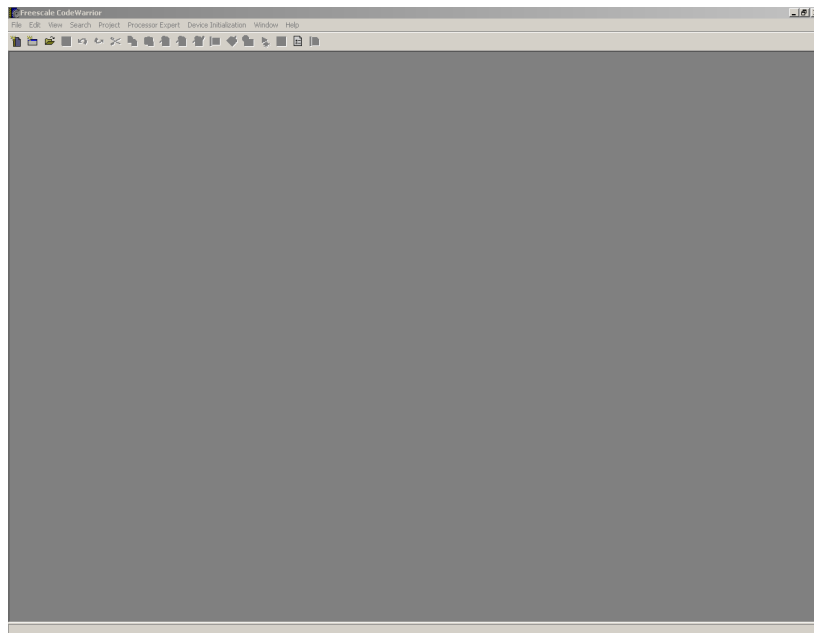


ME 305 Mechatronics
Department of Mechanical Engineering
Cal Poly
Spring Quarter - 2022

Lab 1A: Hardware and Software Familiarization

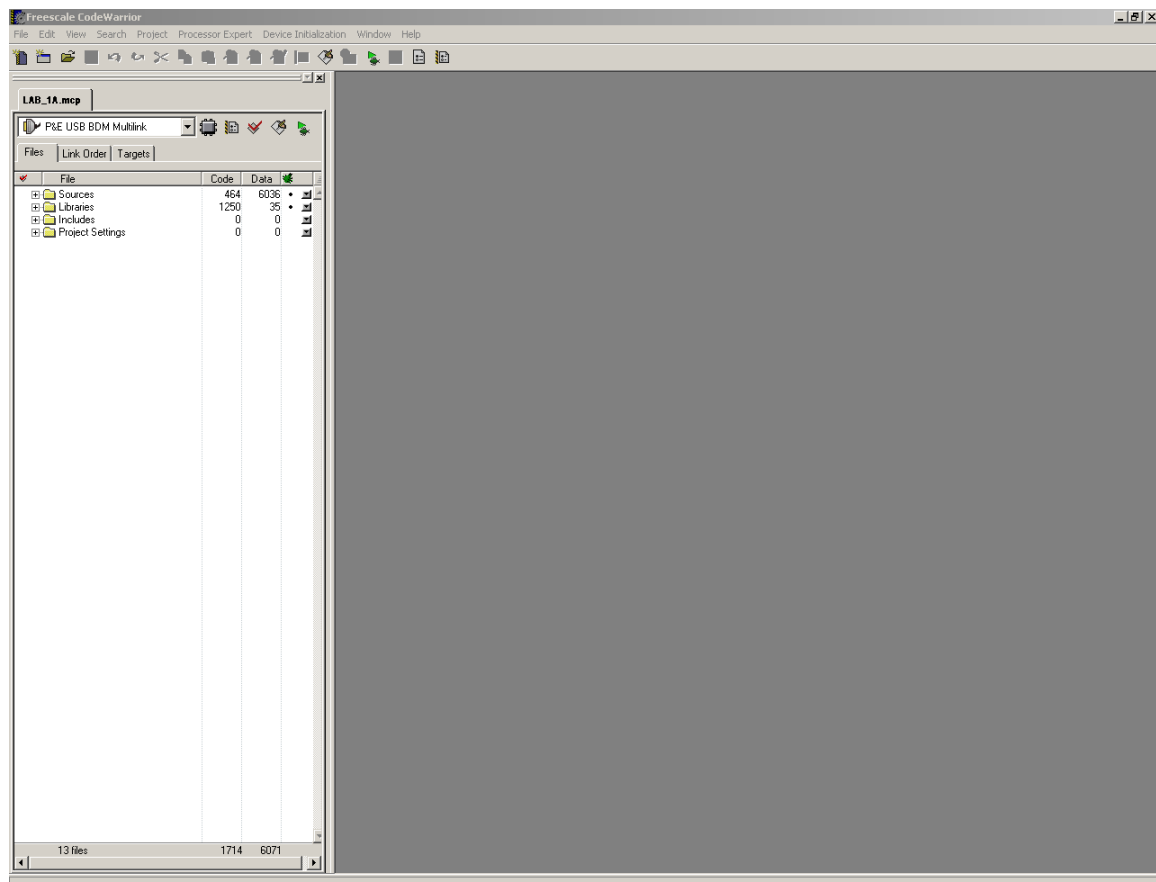
1. Familiarize yourself with the ME 305 Benchtop Unit. Locate the power connectors, BDM, and the Adapt9S12XD module, which contains an MC9S12XDP512 microcontroller.
2. Adjust the +20V output of the power supply to 15 volts. On the power supply, connect a black banana cable to the COM terminal and a red banana cable to the +20V output terminal. Then connect the free end of these banana cables to the appropriate color-coded terminals on the Benchtop Unit.
3. Connect the USB cable to the ME 305 Benchtop Unit. The USB cable from the PC connects to the maroon BDM Multilink box on the right underside of the ME 305 Benchtop Unit. The BDM provides a two-way communication interface between the PC and the Adapt9S12XD module, which is necessary for programming and debugging.
4. Open your OneDrive folder, which you should be able to access by double-clicking the OneDrive desktop icon or by opening File Explorer, then opening OneDrive from the left-hand menu. Go through the OneDrive sign-on procedure.
5. Launch CodeWarrior, which is the IDE/compiler/assembler/debugger that we will use to program the 9S12XD microcontroller. This step can be accomplished by double-clicking the CodeWarrior IDE desktop icon. When opened, the main window should look as shown here.



6. Open the ME 305 Drive: module on the Canvas page for your ME 305 lab section. Locate the blank_project.zip file and download that zip file. Make a copy of the downloaded zip

file and paste that copied zip file into the folder on your OneDrive in which you are intending to work. Right click on your copy of the zip file, click OK, and then select 'Extract All ...'. At this point, you should have a blank_project folder in addition to and separate from the blank_project.zip file; if so, delete the blank_project.zip file to avoid any subsequent confusion because you are done with it. You may work from a flash drive if you prefer, but this is not recommended because there are some problems that occur occasionally when working from a flash drive. You are strongly discouraged from working from the desktop or the ThawSpace(T:). **BEWARE:** uploading and subsequently downloading a text file to/from Google Drive may change the file in subtle ways that cause significant problems with ME 305 source files. **NOTE:** Blank spaces may cause problems in filenames. **Also, NOTE: If you use your OneDrive, you must reboot the computer when you are done to remove all traces of your files from the lab computer on which you were working.**

9. Rename your blank_project folder to Lab_1A. Within that folder rename the following two files: rename blank_project_Data to Lab_1A_Data, and rename blank_project.mcp to Lab_1A.mcp.
10. The easiest way to open the Lab_1A Project in CodeWarrior is to double click on the filename corresponding to Lab_1A.mcp. Alternatively, to open the Lab_1A Project from within CodeWarrior, use the following CodeWarrior menu options: "File -> Open..." and select Lab_1A.mcp. A window similar to the following one should open.



11. Open the main.asm file and you will see shell code to which you can add code to convert this file into fully functional Lab 1A code, that is, add code to this file and/or edit and update where necessary. When done, your Lab_1A code should look as follows (except updated with your particular information where appropriate):

```
*****
; * Blank Project Main [includes LibV2.2]
; *****
; * Summary:
; * -
; *
; * Author: YOUR NAME
; * Cal Poly University
; * Spring 2022
; *
; * Revision History:
; * -
; *
; * ToDo:
; * -
; *****

; /-----\
; | Include all associated files
; \-----/
; The following are external files to be included during assembly

; /-----\
; | External Definitions
; \-----/
; All labels that are referenced by the linker need an external definition

        XDEF    main

; /-----\
; | External References
; \-----/
; All labels from other files must have an external reference

        XREF    ENABLE_MOTOR, DISABLE_MOTOR
        XREF    STARTUP_MOTOR, UPDATE_MOTOR, CURRENT_MOTOR
        XREF    STARTUP_PWM, STARTUP_ATD0, STARTUP_ATD1
        XREF    OUTDACA, OUTDACB
        XREF    STARTUP_ENCODER, READ_ENCODER
        XREF    INITLCD, SETADDR, GETADDR, CURSOR_ON, CURSOR_OFF, DISP_OFF
        XREF    OUTCHAR, OUTCHAR_AT, OUTSTRING, OUTSTRING_AT
        XREF    INITKEY, LKEY_FLG, GETCHAR
        XREF    LCDTEMPLATE, UPDATERLCD_L1, UPDATERLCD_L2
        XREF    LVREF_BUF, LVACT_BUF, LERR_BUF, LEFF_BUF, LKP_BUF, LKI_BUF
        XREF    Entry, ISR_KEYPAD

; /-----\
; | Assembler Equates
; \-----/
; Constant values can be equated here

; /-----\
; | Variables in RAM
; \-----/
; The following variables are located in unpagged ram

DEFAULT_RAM:  SECTION

FIRST:        DS.B  1          ; reserved space for addend
SECOND:       DS.B  1          ; reserved space for augend
RESULT:       DS.B  1          ; reserved space for result
```

```

; /-----\
; | Main Program Code |
; \-----/
; This code implements Lab_1A for ME 305

MyCode:      SECTION

main:  movb  #$04, FIRST      ; initialize addend in FIRST
       movb  #$06, SECOND    ; initialize augend in SECOND
       ldaa  FIRST           ; load accumulator A with addend
       ldab  SECOND          ; load accumulator B with augend
       aba   ; add contents of B to contents of A
       staa  RESULT          ; copy contents of A to result
spin:  bra   spin             ; endless horizontal loop



; /-----\
; | Subroutines |
; \-----/
; General purpose subroutines go here

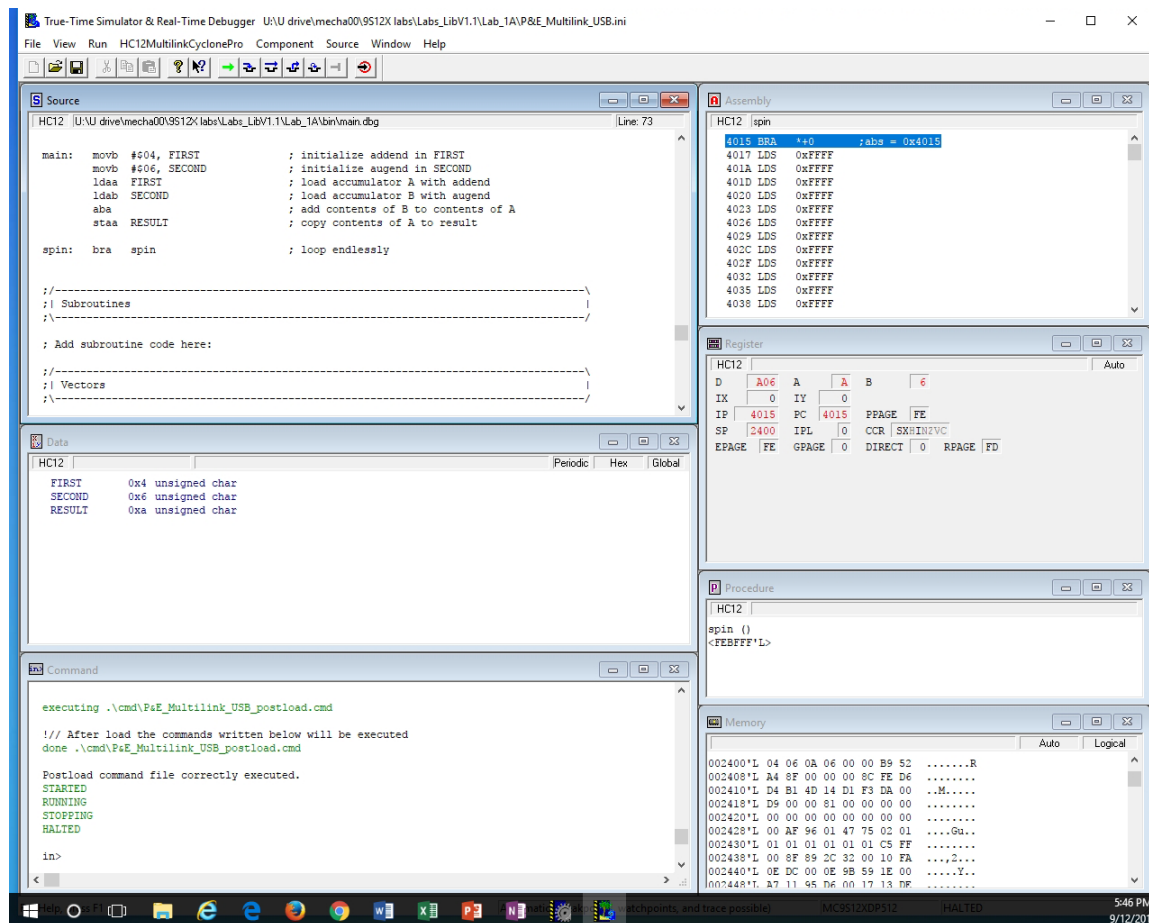
; /-----\
; | ASCII Messages and Constant Data |
; \-----/
; Any constants can be defined here

; /-----\
; | Vectors |
; \-----/
; Add interrupt and reset vectors here

      ORG    $FFFE            ; reset vector address
      DC.W   Entry
      ORG    $FFCE            ; Key Wakeup interrupt vector address [Port J]
      DC.W   ISR_KEYPAD

```

12. Assemble your source code. With your main.asm file in the active window, select the menu option "Project -> Make". Other options to assemble your source code are to press the Make icon , or press the hotkey <Ctrl+F7>.
13. If errors showed up in the assembly process, fix these errors, then save and re-assemble your file. Examine the file "project.map", which is the map file. This file can be found in the "Linker Files" subfolder in the "Project Settings" folder of the Project Lab_1A directory.
14. To load the code onto the 9S12XD you must use the "Debug" command, which can be found from the menu option "Project -> Debug". Alternative methods of opening the debugger are to use the hotkey <F5> or press the debug icon . Any of these options will open a new window called "True-Time Simulator & Real-Time Debugger", which should be similar to the following window.

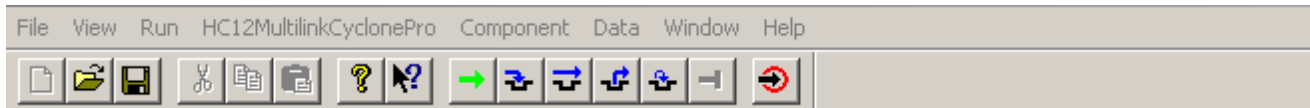


Note: the "Debug" command erases flash in the 9S12XD, loads the new code into flash in the 9S12XD, and enters the debugging mode all in rapid succession.

WARNING: the "Debug" command will cause your source code to be assembled, but it does not automatically cause the assembled code to be saved, whereas the "Make" command causes your source code to be assembled and then the assembled code is automatically saved. Clearly, the safer, smarter option is to use the "Make" command.

15. Across the top of this window are buttons for the most frequently used debugger commands. Below the commands bar are various sub-windows. More specifically, there is the Register window, showing the CPU registers; the Data window, where the contents of selected memory locations may be shown in various 8-bit, 16-bit, and 32-bit formats; a Memory window, where memory contents may be viewed and changed; two Code windows, one showing source code and one showing disassembled code; and a debugger Command window.
16. In the debugger, variables can be observed in pseudo-realtime by adding them to the Data window. Double-click inside the Data window to add variables. Using the name of the variables (`FIRST`, `SECOND`, and `RESULT`), view the contents of the memory locations. Most likely the three memory locations for these variables will be `0x2400`, `0x2401`, and `0x2402`, respectively, but check the ".map" file to be sure.

17. The debugger toolbar is shown in the figure below. Locate the following command buttons in the debugger toolbar: Start/Continue <F5>, Single-Step <F11>, Step-Over <F10>, Step-Out <Shift+F11>, Assembly-Step <Cntrl+F11>, Halt <F6>, and Reset <Cntrl+R>.



18. Run your program by clicking the solid green arrow named Start/Continue on the debugger toolbar. By examining the contents of memory, see if you can determine whether the program did what you expected.
19. Use the Data window to alter the addend and augend. Rerun your program. By examining the contents of memory, see if you can determine whether the program did what you expected. Remember to close the Debugger window when you are through debugging your code. Only one Debugger can be open at any given time: our software/hardware setup debugs one microcontroller through one USP connection to one BDM on a single benchtop unit.
20. Add a **bgnd** instruction, which is the background instruction, as the first executable line of your program. Reassemble your program, load your reassembled code into the microcontroller and into the debugger. Then, execute your code from the debugger again.
21. When program execution stops at the line after the **bgnd** instruction, use the debugger command Single-Step to step through the two **movb** commands, then use the Data window to alter the addend and augend. Use the debugger commands to continue to single step through your program. By examining the contents of memory, see if you can determine whether the program did what you expected.

Lab 1B: More Hardware and Software Familiarization

22. After completing the first portion of the assignment, create a duplicate project directory and name it LAB_1B. Browse inside this directory and rename the appropriate files or directories to correspond to the new project name; i.e., rename "LAB_1A_Data" and "LAB_1A.mcp" to "LAB_1B_Data" and "LAB_1B.mcp", respectively.

Enter, assemble, and execute the Lab 1B code, with debugging as necessary to get a properly functioning program. Shell code to get you started on Lab_1B can be found on the last pages of this handout.

23. Use the oscilloscope to determine the duration of subroutine DELAY as accurately as possible. Explain in detail to your lab instructor the approach you used to accurately measure the duration of subroutine DELAY and have your instructor verify the duration you obtained.
24. Obtain, save, and print an oscilloscope screenshot of the command signals to the LEDs.

Detailed Information for Lab 1B

- A protoboard containing an electrical circuit that drives two separate LEDs with two separate BS170 MOSFETs will be provided to you. For convenience, call these LEDs G_LED and R_LED, for green and red, respectively. Two output bits on port P will be used to switch these MOSFETs ON and OFF independently. Additional information on how to connect the 9S12XD to these components will be provided in lab.
- Write the main section of a program that turns the LEDs ON and OFF according to the following cycle:

G_LED ON	and	R_LED OFF
G_LED OFF	and	R_LED OFF
G_LED OFF	and	R_LED ON
G_LED OFF	and	R_LED OFF
G_LED ON	and	R_LED ON
G_LED OFF	and	R_LED OFF

- Wire the 8 bits of Port T to be the integer number of tenths of a second that you wish your program to delay between each step of the LED lighting cycle.

Again, the timing of the LED lighting cycle will be determined according to the state of Port T. To turn the LEDs ON and OFF, you should use the bit set (**bset**) and bit clear (**bclr**) commands, respectively.

The code shell that you are to modify for Lab 1B can be found beginning on the following page. This code shell includes the following features:

- Initialization of Port T for input.
- Initialization of two pins of Port P for output.
- A time delay routine that provides a 0.1 second time delay to be used in conjunction with the 8-bit number read from Port T. The total time delayed will be 0.1 seconds times the integer read from Port T.

Modify this program so that the tasks listed above loop endlessly. Then, use the oscilloscope to determine your best measurement of the duration of subroutine DELAY. Discuss your measurement with your lab instructor.

Open the Tektronix OpenChoice Desktop application by double-clicking on its desktop icon. Use the Get Screen and Save As features in OpenChoice Desktop to capture and save a screenshot from your TDS 2022B oscilloscope that verifies the correct operation of your Lab_1B code and that you can use to support the quality of your measurement of the duration of subroutine DELAY. For subsequent post processing in later labs, you will need to use the Waveform Data Capture feature of OpenChoice Desktop to save a .csv file containing data from the oscilloscope, where 'csv' stands for 'comma separated variable'. Your .csv file may then be opened in a wide variety of programs, including Matlab and Excel.

; Paste the following code into your Lab 1B in the "Assembler Equates" section

```
PORTT      EQU    $0240          ; input port for DELAY_CNT
DDRT       EQU    $0242
PORTP      EQU    $0258          ; output port for driving LEDs
DDRP       EQU    $025A
LED_MSK    EQU    %00110000      ; LED output pins
G_LED      EQU    %00010000      ; green LED output pin
R_LED      EQU    %00100000      ; red LED output pin
```

; Paste the following code into your Lab 1B in the "Variables in RAM" section

```
DEFAULT_RAM: SECTION
DELAY_CNT  DS.B  1
```

; Paste the following code into your Lab 1B in the "Main Program Code" section

```
; followed by the addition of your own code
MyCode     SECTION
main:      jsr     SETUP          ; jump to SETUP subroutine
```

; YOUR CODE GOES HERE!

; Paste the following code into your Lab 1B in the "Subroutines" section

```
;-----Delay-----
DELAY:
    ldaa  PORTT          ; (3) load 8-bit DELAY_CNT from PORTT
    staa  DELAY_CNT      ; (3)
OUTER:
    ldaa  DELAY_CNT      ; (3)
    cmpa  #0             ; (1)
    beq   EXIT           ; (1)
    dec   DELAY_CNT      ; (4)
    ldx   #$0005         ; (2)
MIDDLE:
    cpx   #0             ; (2)
    beq   OUTER          ; (1)
    dex   #0             ; (1)
    ldy   #$7710         ; (2)
INNER:
    cpy   #0             ; (2)
    beq   MIDDLE         ; (1)
    dey   #0             ; (1)
    bra   INNER          ; (3)
EXIT:
    rts                 ; (5) exit DELAY
```

; Paste the following code into your Lab 1B in the "Subroutines" section

```
SETUP:
; setup IO ports

    clr   DDRT           ; set PORTT to input
    bclr  PORTP, LED_MSK ; initialize LEDs to off
    bset  DDRP, LED_MSK  ; set LED pins to output
    rts                 ; exit SETUP

; end subroutine SETUP
```

; Make sure that the following code is in your Lab 1B in the "Vectors" section

```
ORG    $FFFE          ; reset vector address
DC.W   Entry
ORG    $FFCE          ; Key Wakeup interrupt vector address [Port J]
DC.W   ISR_KEYPAD
```