Modular Programming—Motor Control Interface

The purpose of this lab is to gain further experience in writing modular assembly programs and debugging programs. This lab builds on the knowledge gained from the previous lab to read the keypad and output characters to the LCD screen. The end result of this exercise will be a fully functional user interface for the DC motor including manual entry of the speed, menu options for the approximation type, and error detection.

Objectives

- Gain further experience in modular programming and documentation.
- Understand how to implement user input from the keyboard from both menus and strings.
- Use error detection to display error messages for invalid control entries.

Turn-In Requirements:

1. Pages 2 and 3 and answers to the 5 questions as specified in the lab policies handout.

References


Equipment for this lab:

- 68HC11 trainer kit, to include 68HC11 EVBU and prototyping strips
- IBM compatible PC to connect to the trainer kit via an RS-232 serial cable
- Floppy disk provided by the student
*In completing this lab, I have abstained from any form of academic dishonesty or deception (e.g. cheating, lying, stealing or plagiarism). This work is entirely of my own origin (beyond the code that was supplied to me by the instructor).

Name: __________________________            Signature: ___________________________

Lab 2 p.2

**Laboratory Exercise**

**Timing**

It is anticipated that this lab will take one **period** to complete.

**Notes**

- In this lab, all numbers such as addresses and data are given in *hexadecimal* format (“hex”) unless otherwise indicated. In completing the lab, record all information in hexadecimal unless directed differently.

- Make sure your development EVBU board is connected to power (green LED on board ON) and the serial port of the EVBU is connected to the serial port of your development PC containing the AXIDE software, configured to the correct port at the correct baud rate, etc.

- Be sure to bring your spd_3.RTF file from last period. One additional subroutine is available on the EE-3306 Web site under **RESOURCES**. The file name is *getdir.RTF* and can be used for option (2) of the “Speed Manual Entry Mode” (see the note on the next page). Feel free to copy this file and append it to the end of the spd_3.RTF file used in Lab #1 (note you’ll need to make strings OPT4, OPT5, and DIR and enter a call to a subroutine to display an error message for the code to function) or to develop your own code. The code you submit should be named *control.RTF*.

**Motor Control Functions**

In this lab you will append code to the first lab’s program and modify the code for the initial display to obtain user input which will determine the control that could be used for a motor. This will include several approximation types, manual entry of the motor speed, and resetting these variables (thus stopping the motor). In addition, you will be expected to determine when the user has entered an invalid input key and display an error message accordingly. The keys corresponding to the functions are as indicated below. When in the two entry modes, the LCD should display enough information for the user to understand what to do. The TA will have a working example of one implementation.

**When in “Display Mode”:**

- Increment Speed by 100………………………………… …….A (same as Lab 1)
- Decrement Speed by 100……………………………….............B (same as Lab 1)
- Double Speed…………………………………………………..C (same as Lab 1)
- Reset Speed and Approximation Type to Defaults……………*  
- Enter “Speed Manual Entry Mode”…………………………….#  
- Enter “Acceleration Type Entry Mode”………………………..D  

**When in “Speed Manual Entry Mode”:**

- Enter Speed and Return to “Display Mode”…………………..# (see note on next page)

**When in “Approximation Type Entry Mode”:**
Use Linear Approximation……………………………………..0
Use Positive Quadratic Approximation………………………...1
Use Negative Quadratic Approximation………………………..2

Note: When entering the speed, you must be able to determine the direction (positive indicating clockwise, negative indicating counterclockwise). You may do this in any manner that you wish. Two possible suggested ways are (1) to read in the entire string and check for a negative sign character, ‘-‘, at the beginning of the string or (2) only allowing positive integers to be input and then using the ‘#’ key to enter a “Direction Mode” in which the user is allowed to choose between Clockwise and Counterclockwise (Positive and Negative). These are, however, just suggestions so if you have a different manner of implementing the functionally, by all means do so.

The keypad input and LCD output program used in the previous lab can be used to read display inputs. As previously noted, an additional partial subroutine is provided:

get_dir: This subroutine takes as input the absolute magnitude of the speed in Register X and displays on the LCD two options for direction: pushing ‘1’ will make the sign/direction negative while pressing ‘2’ will make the sign/direction positive. The updated speed will be then returned in Register X. Error detection is also included but the call to the actual error display subroutine must be entered. Lastly, the strings OPT4, OPT5, and DIR must be created to instruct the user what keys are possible to push with what results.

Requirements of the program:

1. When in the “Display Mode”:
   • The LCD should display the current approximation type (default should be 0 (linear))
   • The LCD should display the current speed (default should be 0)
   • The program should respond appropriately to the keys specified on p.2 (and display an error message for others)

2. When in the “Speed Manual Entry Mode”:
   • The LCD should display a line requesting the user for a new speed (e.g. “Please enter the speed”)
   • The LCD should display the numbers for the speed as they are entered
   • Once a number is validly entered, the speed will update to 100 times the number entered in the “Display Mode”

3. When in the “Acceleration Type Entry Mode”, the LCD should display:
   • The options for the different acceleration types (0 = linear, 1 = pos quadratic, 2 = neg quadratic)
   • If an intermediate mode is used for direction entry, the direction options should be displayed

4. Error Detection:
   • If the user presses a character from the keypad that is not specified as an input for a given mode, the LCD should briefly display an error message and then return to the previous display (you may use a single error message for all types of errors).
   • Extra Credit will be given if the program also is able to detect errors in the “Speed Manual Entry Mode” (e.g. a number greater than 50 is entered, the letter ‘A’ is entered, etc.)

5. For any strings being displayed, you must follow the above guidelines but the exact wording does not matter.

6. Demonstrate to the T. A. that you program is running. T. A. Initials:______________________

Suggestions Your code will likely need the following:
   • New case statements (e.g. check_D) in the main loop of the program for the “Display Mode” key presses.
• A modification to disp_speed to display the approximation type in addition to the speed
• A subroutine that will handle the manual entry of the speed
• A subroutine that will handle the approximation type menu and data entry
• A subroutine that will display an error message and wait for a few seconds

When you are done, do the Following to document your program (Note: comments, structure, and readability counts!!!)

1. Hand-in a complete Flowchart and for the entire program (control.RTF).
2. Hand in the listing of the complete, well-commented source-code that matches the flowchart. Remember, if the TA can not read your code with “reasonable” effort, then it is wrong.

Questions When you turn in the lab report, please answer the following questions. Provide flow charts and/or assembly language programs as necessary to explain your answers.

1. Where is the ‘reset’ vector of the 68HC11 located? Suppose your program is located at address $675f$, what do you need to do to get the 68HC11 to execute your program?
2. How do you modify ‘ascii_to_dec’ subroutine to return an error if the first character it encounters is non numeric?
3. If the input to ‘ascii_to_dec’ subroutine is $312_{10}$, what will acc A contain when the program exits the subroutine?
4. If the ‘dec_to_lcd’ subroutine interprets the number in acc D as a signed 16 bit number, what will it print if acc D has the following:
   • $10675_{10}$
   • $32768_{10}$
   • $62453_{10}$