ECET 20900

Laboratory 12-13
C Language ADC with Stepper Motor Integrations

Objective: Design and Write a C language program to drive the stepper motor and control its speed based on the potentiometer setting read from the ADC. The motor will also have two directions (forward or backward) controlled by bit-zero of the DIP-switches.

****
Do not code anything Yet!
Read **all** instructions carefully and
Complete the Program Development Cycle worksheets.
*****

Program Description:
Bit 0 from the dip switches connected to Port B is used to check the desired motor direction. If bit 0 is high the motor should turn in a forward direction and if bit 0 is low the motor should turn in the backwards direction. The speed of the motor is controlled by the input read from the potentiometer (ADC channel 2) scaled to a value passed to a delay function.

The following sequence must be sent to the motor, each byte command sent will step the motor one step:
0x05, 0x09, 0x0A, 0x06

There must be a delay between each command sent to the motor. The length of the delay determines the speed the motor will turn. The shorter the delay time the faster the motor will turn. The delay time sent to a Delay function is based on the ADC value read from the potentiometer (ADC channel 2) scaled to an appropriate value. As the potentiometer is turned from minimum (full CCW) position up to maximum (fully CW) position the motor speed will increase from minimum speed to maximum speed respectively.

Program Files and Code Organization:
Your program should have no global variables and at least five or more C language functions.

- One function should step the motor:
- One function should initialize the ADC
- One function should read the potentiometer
- One function should delay
- The main function should call these functions.

You must start a **new program**, add additional files to your program and use code from your last three previous labs (08_09, 10_11 Part I and 10_11 Part II) to complete this lab. Put the stepper motor and ADC functions in a file named MotorControl.c. Put function prototypes in a file named Protos.h. Put calls to your functions in main.c file (**Refer to instructions section below: How to add .c and .h files to your project).**

- The step function: is called from main function. The step function must define a character array with the values to control the stepper motor and a static char index variable so that its value will be retained between calls. The index will be incremented if the motor is going in the forward direction, decremented if going in reverse direction, and appropriately reset when outside of the array limits. A C-language *if* statement can be used in the function to check bit 0 of port B and execute the forward direction block of code if it is high and the reverse block of code if it is low.

- Two ADC functions:
    1. An ADC initialize function should: power-up the ADC, wait for power to stabilize, if necessary, set 8-bit mode, 4 conversions on a single channel (register).

2. A function to read the potentiometer from the ADC should: display 8-bit reading to LEDs, then the code should scale a delay value (count) based on the 8-bit ADC reading, then return the scaled delay value (unsigned int) back to main.
   Both of these functions are called from main function.

- The delay function is called from main function and passed a delay value (count) that is type *unsigned int*. In the function, use a C-language *for* loop or *while* loop that loops the number of times passed by the argument.

- The main function runs an endless loop. Use the endless *for* loop provided by CodeWarrior. At start-up the main function must call the ACD initialize function. Then the main function must call the step function. Then main function must call an ADC function that reads the pot and returns a scaled delay value (count) back to main. Next main function must call the delay function with the scaled delay value. The main function must start the motor upon program startup and keep it going regardless of direction or speed until the program closes.

Write you program such that only code that requires continuous operation are in the endless *for*-loop. Put one-time operations and initialization in the appropriate program locations. All code must follow good programing practices as taught in this class.

Program Development Cycle worksheet (C-language) steps 1-3:
Thoroughly design/outline your program and each of the required functions. Include flow charts for your program including each functions tasks. Also, include your logic equations to scale.
Refer to **Program Files and Code Organization** section above to Plan how to organize and then start your program solution. **Testing:** Debug the program by stepping through it and examining registers. Set the potentiometer to a low value until you are positive your code will work while debugging. Once the code is debugged, the potentiometer can be varied, and the code/processor run at speed. The motor should not miss-step when reversing direction.

The program must use standard C language conventions, follow code organization format as outlined in lecture and posted examples, and be fully documented

Submit: your design worksheets .docx**,** main.c, MotorControl.c, Protos.h and your entire zipped project file.

Program design Outline Notes:

1. Complete all design worksheets. Include your pot step calculations. Include all design items in one well formatted word docx.
2. Start a new C language CodeWarrior program: TeamName_Lab12_13
3. Add the two new files as instructed below. **How to add .c and .h files to your project**
4. Do not continue until you have successfully added the two new files.
5. Refer to your previous Stepper Lab
   a. copy the appropriate functions into your MotorControl.c file,
   b. copy the appropriate function calls into your main.c file,
   c. copy all needed function prototypes into your Protos.h file.
   d. Delete all unnecessary code, edit the appropriate code to meet this labs requirements!
6. ****!Run your program to ensure it works.!****
7. Then copy the ADC functions from your ADC potentiometer lab to your MotorControl.c File.
   Only add the necessary code, edit the appropriate code to meet this labs requirements!

8. ****!Run your program to ensure it works.!****
9. Then write the code to scale the delay value (this is done in your ADC function that reads).

Theory to scale ADC reading for delay value: Slowest speed ~60,000 Fastest speed ~4,000
Then, our range is 60,000 – 4,000 = 56,000 and our ATD pot range was 0 to your highest reading from lab 9.II, possible values to scale.
This gives us a 56,000 range to divide by 127 different settings. 56,000/127 = 440 (Drop any remainder and use the quotient as the multiplier). Therefore, our delay value step is ~440. Of course, 127 is an example, you must use your highest reading from lab 9.II.

So if the pot is turned completely to counter clock wise CCW (minimum)

ADC reading is 0 and delay value is	60,000	(60,000 - (0*440))

If the pot is turned one bit-step up,

ADC reading is 1 and delay value is	59,560	(60,000 - (1*440))

If the pot is turned 3 bits up,

ADC reading is 3 and delay value is	58,680	(60,000 - (3*440))

If the pot is turned N# bits up,

ADC reading is N# and delay value is	###	(60,000 - (N# * 440))

If the pot is turned completely to clock wise CC (up)

ADC reading is 127 and delay value is	4120	(60,000 – (127*440))

Note: do not use 127, you must use your highest reading from lab 10_11 Part II.

Of course this is just an outline of the theory as to how to scale motor speed based on pot reading/turn position. You will need to figure out how to write this outlined logic as code in your program. Before you try to add this logic, make sure your program runs all the logic from just the stepper part. Be sure to include your calculation for the ADC in your outline.

How to add .c and .h files to your project:
Once you have CodeWarrior C project properly started, named and saved.

- Add a .h file to your project:
  all .h files should be stored in the projects Includes folder
  1. Open a new file:
     From the project menu bar select File | *New Text File*. An untitled file will open.
  2. Name and Save the file:
     With the new file selected (cursor blinking in the empty file), select File | Save As… , In the Save As dialog window ensure your current project folder is displayed. For the *Object name* enter: *Protos.h, this will be the name of your new .h file. C*lick on the save button. You should now see the file name at the top of your new file, and its path is in your project's folder.
  3. Add the file to your project:
     Next, you MUST ADD the file to your project: In the project explorer tree on the left side of the CodeWarrior IDE screen, Right-click on "Includes" folder | Select Add Files.., In the Select files to add… dialog window, Scroll down the displayed list and select *Protos.h* file from the list, then click the Open button.
     If added correctly, file should now appear in explorer folder tree, inside the Includes folder.
  4. Next: In the Protos.h file, type your commented header, later you will add the function prototypes for all the functions in you program.
     Note: when you add .h files to your project, you do not do step 5.

- Add a .c file to your project:
  all .c files should be stored in a projects Sources folder
  1. Open a new file
     From the project menu bar select File | New Text File, an untitled file will open.
  2. Name and Save the file:
     With the new file selected (cursor blinking in the empty file), select File | Save As… In the Save As dialog window, ensure your current project folder is displayed, then navigate to and double click to open the Sources folder. For the Object name enter: MotorControl.c This will be the name of your new .c file. Click on the Save button. You should see the file name at the top of your new file, and its path is in your project's folder.
  3. Add the file to your project:
     Next you MUST ADD the file to your project. In the project explorer tree on the left side of the CodeWarrior IDE screen, Right click on "Sources" folder | Select Add Files… In the Select files to add… dialog window, scroll to and select MotorControl.c file from the list, then click the Open button. You should see the MotorControl.c file display in the Sources folder in the explorer folder tree.
  4. Next: In the MotorControl.c file, add the following #includes at the top left side of the file:
     ```
     #include <hidef.h>      /* common defines and macros */
     #include "derivative.h"      /* derivative-specific definitions */
     ```
     Then type your commented header for this file.
  5. Then in the main.c file, directly below the last #include, you will include *Protos.h* by typing:
     #include "Protos.h"
     (Note: when you add .h files to your project, you do not do step 5.)
  6. Build your program to ensure all is still working. Once your code is working….Now you can return to step 3 in ***Program design Outline Notes*** section above.

For pre-lab work submission:

See Module Week 12_13 Pre-lab assignment details.

For Lab 12_13 submission
Submit:

Each student will submit the following:

- **In** one well formatted word .docx:
  - **All** Design documents, complete and professional
  - including your complete calculations with comments as to their purpose.
- main.c,
- MotorControl.c,
- Protos.h
- And your entire zipped project file.