ECE 375: Computer Organization and Assembly Language Programming

Lab 1 – Introduction to AVR Development Tools

SECTION OVERVIEW

Complete the following objectives:

- Connect your AVR microcontroller board to a TekBot (optional).
- Create a new Atmel Studio project.
- Download and compile the sample AVR assembly source code given on the lab webpage (`BasicBumpBot.asm`).
- Understand how to connect and operate the `USBASP` AVR programmer.
- Upload the previously-compiled sample program to the flash memory of the TekBots AVR microcontroller board (`mega128`), and observe it running.

PRELAB

For most labs, you will be required to complete a prelab assignment before attending the lab session itself. These prelabs will cover the important concepts and background knowledge necessary to accomplish each corresponding lab.

Prelab assignments are due at the beginning of your lab session each week. If you have not submitted your prelab by the beginning of the lab session, you will receive no credit for that prelab. For general information about assignment submission policies, please refer to the syllabus on the lab webpage.

For this first lab, there is no prelab assignment.

PROCEDURE

Wiring Your TekBot

1. **Use of the TekBot is optional for the ECE 375 lab**. If you choose to use your TekBot, use the following instructions to ensure it is properly wired to the `mega128` board. Otherwise, skip ahead to the next subsection, entitled "Compiling an AVR Assembly Program".

Connection	Port	Pin	Alternate Function
Right Whisker	D	0	External Interrupt 0
Left Whisker	D	1	External Interrupt 1
Right Motor Enable	B	4	PWM Output for TCNT0
Right Motor Direction	B	5	PWM Output A for TCNT1
Left Motor Direction	B	6	PWM Output B for TCNT1
Left Motor Enable	B	7	PWM Output for TCNT2

Table 1: `mega128` Port Usage for TekBot Connection

2. Table 1 shows which port pins of the `mega128` board must be connected to the TekBot. Use this table in conjunction with the wiring diagrams given on the lab webpage to perform the wiring properly.

   To make the wires needed to connect your `mega128` board to your TekBot, you can use the ribbon cable that came with your original TekBot kit, solder male headers to the ends, and add a bit of heat shrink tubing onto each connection. If you are not familiar with this technique, there is a short tutorial on the TekBots webpage that explains this process in detail.

   Since we are working with a modular programmable AVR microcontroller board, you *could* connect the whisker inputs and motor controller outputs to any pins on any port on the board. But, for this lab and future labs to work properly, these specific wires need to be connected in a certain way. Please follow the convention specified in Table 1 and maintain this wiring throughout the course, so that your use of some alternate pin functions in later labs behaves as expected.

3. When you have completed the wiring, be sure to double-check all connections before you turn the power on. If you have wired things incorrectly (especially if you have switched Vcc and ground), you can easily destroy parts of your TekBot. If you aren't confident in the correctness of your wiring, please ask a fellow student, your TA, or the TekBots store to double-check your work.

Compiling an AVR Assembly Program

1. Download the sample code (`BasicBumpBot.asm`) available on the lab webpage. This is a simple AVR assembly program that is well-commented and ready to compile. All code that you produce for this course should be as well-commented as this code. Save this code somewhere you can find it.
2. Atmel Studio is the Integrated Development Environment (IDE) that you will be using to develop your AVR assembly code throughout this course

(with the exception of Lab 2, which uses C). Atmel Studio is a powerful IDE created by Atmel for their line of AVR microcontrollers. You will be using it to write assembly programs for your AVR microcontroller board (`mega128`), which uses an ATmega128 microcontroller. Section 2 of the AVR Starter Guide, which can be found on the lab website, contains a good overview on how to use the IDE, as well as some step-by-step tutorials. Briefly read through this section to gain a basic understanding of the Atmel Studio workflow.

Atmel Studio is already installed on the computers in the lab classroom. If you plan to use your personal Windows computer for some or all of your lab work, the latest version of Atmel Studio is available for download on Atmel's website. It is also possible to use your personal Mac OS X or Linux computer for AVR assembly programming, but you will need to find an alternative to Atmel Studio on your own. Also, a few of the labs in this class rely heavily on the use of Atmel Studio's built-in AVR simulator, so please keep in mind that occasionally you may be relegated to completing your work on one of the lab computers instead of your personal device.

3. Once you have finished browsing through Section 2 of the AVR Starter Guide, follow the specific steps in Section 2.1.2 to create a new project. In most IDE tools, a project is the base starting area to your program. It consists of all files you use and any settings for the program. In this case, the file you'll want to include is the AVR assembly sample program, which you downloaded from the lab website earlier. Continue following the instructions in Section 2.1.2 to associate the sample program with the project you just created.

4. Now that you have loaded the sample program into your Atmel Studio project, take a moment to look over the sample code. Although you haven't seen any AVR assembly code in this class yet, read the comments and see if you can follow the general structure and flow of the program. For reference, the general operation of the program is described in Figure 1.

5. Using the project you created, continue on to Section 2.1.3, and perform steps 1 and 2 to compile the sample program (you can disregard the simulation instructions in this section, as this topic will be covered in a later lab). Note: When an AVR assembly program is compiled, the resulting output is a binary program file (called a HEX file, with a `.hex` extension). This HEX file contains the actual binary instructions that will be executed by the ATmega128 microcontroller.

- Initializes key components of the ATmega128
- Starts the TekBot moving forward
- Polls the whiskers for input
- If right whisker is hit
  - Backs up for a second
  - Turns left for a second
  - Continues Forward
- If left whisker is hit
  - Backs up for a second
  - Turns right for a second
  - Continues Forward

Figure 1: Theory of Operation for Lab 1 AVR Assembly Code

You now know how to create a new AVR project in Atmel Studio, include an assembly program into that project, and then compile the program into usable program code. Next, you will learn how to upload the compiled program to the microcontroller and watch it run.

Uploading a Compiled AVR Program

1. Use the Windows application `Universal_GUI.exe` (already installed on the lab computers, and available for download on the lab webpage) to upload the compiled `.hex` file to the flash memory of the ATmega128 microcontroller. (For detailed instructions, follow the steps given in the Universal GUI User Guide linked on the lab webpage.)

2. With the HEX file uploaded to the microcontroller, you can observe the behavior of the sample program (if using a TekBot, unplug it from the computer and turn it on). The LEDs on the `mega128` board should indicate that the board is performing a basic BumpBot routine (remember, Figure 1 gives a description of the full routine). To receive your implementation points for this lab, demonstrate to your TA that your board is performing this correct BumpBot behavior.

STUDY QUESTIONS / REPORT

For most labs, you will be required to submit a short write-up that details what you did and why, but **there is no write-up required for this first lab**. Instead, you will only need to hand in the answers to the study questions given later in this document.

When putting together each lab's write-up, **you must use one of the templates (`.doc` or `.tex`) provided on the lab webpage**. Include enough detail so another student could recreate the lab by using your write-up as a guide. See the lab webpage and template for specific details on what should be included in the lab write-up. You must submit your write-up and code via **Canvas** by the start of the following lab. NO LATE WORK IS ACCEPTED.

You are responsible for turning in a clean, organized, and professional document free of misspelled words. The code you turn in must include sufficient comments, such that ANOTHER STUDENT could reasonably be able to understand your code. Code that is not well-documented will result in a severe loss of points for the write-up portion of your lab grade. For an example of the style and detail expected of your comments, look at the code you just downloaded. Generally you should have a comment for every line of code.

Study Questions

Most of the labs you do in this class will have study questions that are to be answered within your lab write-up. This lab's study questions are given below, and will be due at the start of lab next week. Although you will be exposed to some information that has not been covered in class, keep in mind that as a student accepted into pro school you will need to get into the habit of being proactive. This involves some independent learning, such as reading ahead and preparing yourself before going to lab or class.

1. Go to the lab webpage and download the template write-up. Read it thoroughly and get familiar with the expected format. **What specific font is used for source code, and at what size?** *From here on, when you include your source code in your lab write-up, you must adhere to the specified font type and size.*
2. Go to the lab webpage and read Syllabus carefully. Expected format and naming convention are very important for submission. If you do not follow naming conventions and formats, you will lose some points. **What is the naming convention for source code (asm)? What is the naming convention for source code files if you are working with your partner?**
3. Take a look at the code you downloaded for today's lab. Notice the lines that begin with `.def` and `.equ` followed by some type of expression. These are known as **pre-compiler directives**. Define pre-compiler directive. What is the difference between the `.def` and `.equ` directives? (HINT: see Section 5.1 of the AVR Starter Guide).
4. Take another look at the code you downloaded for today's lab. Read the comment that describes the macro definitions. From that explanation, determine the 8-bit binary value that each of the following expressions evaluates to. Note: the numbers below are decimal values.

   (a) $(1 << 3)$
   (b) $(2 << 2)$
   (c) $(8 >> 1)$
   (d) $(1 << 0)$
   (e) $(6 >> 1|1 << 6)$

CHALLENGE

Challenge problems are additional tasks that can be performed for several of the labs. By successfully completing a particular lab's challenge problem, you can get extra credit for that lab. To get credit for a challenge problem, you must successfully demonstrate it to your TA, as well as document it in your lab write-up. This lab's challenge problem is given below:

- You have learned to use a simple tool to download a sample program into your `mega128` board. Modify the sample program so the TekBot will reverse for twice as long before turning away and resuming forward motion. Demonstrate this to your TA, and include a copy of your modified assembly program when you submit your answers to the study questions.