Lab Assignment 1—Ready, Set, Go!
Introduction to the AVR Microcontroller

Overview

In this lab you will get acquainted with the **ReadyAVR-64** development board (a.k.a. the *ReadyAVR*). This board is based on the popular ATmega128A microcontroller from Atmel. In this lab you will use the **Atmel Studio** Integrated Development Environment along with the **Atmel-ICE JTAG programmer** to compile and download an assembly language program into the ATmega128A, and you'll use **PuTTY**, a very popular serial-port terminal emulator, to communicate between a PC and6 the code running on the ReadyAVR.

Administrative Information

You'll be issued a lab kit to use this semester. **You are responsible for the items in your kit and will need to return everything at the end of the semester, so please handle with care**. You can store your kit in F103 when not in use, or you can take it with you and work on projects outside of lab. To work outside the lab you'll need to download and install **Atmel Studio** on your own computer. Version **v7.0.1645** is installed on the lab computers, and can be downloaded at https://www.microchip.com/mplab/avr-support/avr-and-sam-downloads-archive, and you may have to register with Atmel before downloading the software. If you decide to use a newer version, **be sure it's backward compatible with the version in the lab**! Also, Studio 7 is a Windows-only product—there is currently no support for OS X or Linux. ☹

Background: ATmega128A Specifications

Using the ATmega128A datasheet, the ReadyAVR-64 user's manual, and the Atmel 8-bit AVR Instruction Set (available on Canvas in the Files | Docs folder), answer the following questions.

1. Define the following acronyms:
    a. ADC -
    b. PWM -
    c. USART -
    d. EEPROM -
    e. SRAM -
2. What is the operating voltage range of the ATmega128A microcontroller?
3. What is the allowable input power supply voltage for the ReadyAVR-64 board?
4. In theory, how many `ADD` instructions can be stored in ATmega128A Flash memory?
5. List at least **two different types** of ***input*** **devices** available on the ReadyAVR-64.
6. List at least **two different types** of ***output*** **devices** available on the ReadyAVR-64.

Lab Procedure:

The steps below will show you how to create a new **Atmel Studio "solution"** named `EECE3624Labs` to hold your lab projects this semester. Each lab assignment will be a separate Atmel Studio **"project"** stored within the `EECE3624Labs` solution. You'll also use **GitHub Classroom** to create a remote Lab01 "starter code" repo, and you'll use **Git Shell** to **clone** that repo to your local storage device. Once the lab code is saved locally, you'll configure your ReadyAVR board, connect it to your computer along with the Atmel-ICE programmer, and use Atmel Studio to assemble, download, and run the Lab01.asm program on your ReadyAVR.

1. Open Atmel Studio and select **File | New | Project…** and select **Atmel Studio Solution**. Name the solution **EECE3624Labs** and set the Location field to the root of your personal storage device (e.g. D:\, **where "D" is replaced by the drive letter of *your* device**.) Make sure the "Create directory for solution" box is checked, and click "OK" to create the empty solution. **Remember:** You'll use a ***single solution*** all semester containing ***separate projects*** for each lab.

2. You should have received an email from your instructor containing a GitHub link associated with this lab. If you haven't already done so, click that link to be taken to a **GitHub Classroom** page where you'll see an "Accept this assignment" link. Click that link, providing your GitHub username and password if needed, and GitHub Classroom will create a remote Lab01 assignment repository for you, pre-populated with some starter code. Once the repo has been created, a link similar to `https://github.com/EECE3624-S2021/lab01-YourGitHubUserName` will be shown, where your GitHub user name will appear as part of the repo name. Click that link to view your new remote GitHub repo. (Remember, "remote" means in your GitHub cloud account.)

A few things to note. First, you have full read/write access to this repo but the owner is an organization named **EECE3624-S2021**, not you. Future assignment repos will also be owned by **EECE3624-S2021** and will have URLs similar to the one shown in the previous paragraph but with new lab numbers (e.g. `https://github.com/EECE3624-S2021/lab02-YourGitHubUserName`). Second, assignment repos are **private**, and only you and your instructor have access to them, so don't share the repo with other students. Third, these repos are remote so **you'll need to create local repos** to stage and commit any code changes you make (described below). And, of course, you should periodically add, commit, and push local changes to the remote assignment repo.

Once you've completed a lab, the remote repo becomes part of your submission. Therefore, **do not rename or delete the remote assignment repository**, as it will become **ungradable**.

3. With GitHub opened to your remote assignment repo, use the Clone or download ▾ button to copy the repo's URL to your paste buffer. The easiest way to do this is to click the 📋 icon.

Next, open **Git Shell** on your computer and navigate to your `D:\EECE3624Labs` Atmel Studio solution directory. Enter the following command to create a new `Lab01` directory inside the solution directory and populate it with the contents of the remote repo. (Hint: R-click the mouse to copy the repo's URL to the command line. And don't forget the directory name at the end.)

```
D:\EECE3624Labs> git clone remote-repo-URL Lab01
```

4. With the `EECE3624Labs` solution open in **Atmel Studio**, R-click on the solution name (in Solution Explorer), click **Add | Existing Proje**ct…, browse to `D:\EECE3624Labs\Lab01` and select the `Lab01.asmproj` file. This will add the `Lab01` project to the current solution. R-click on the newly added `Lab01` project and select **Set as StartUp Project**. Then R-click on the `Lab01.asm` assembly file and select **Set as EntryFile**. In later labs, be sure to do this for the latest lab project and startup file or you'll end up compiling and running an older lab's code!

5. Locate the JP2 pin block on the ReadyAVR board and make sure the two PD2 pins are connected via 2-pin jumper. Ensure the two PD3 pins are connected via a 2-pin jumper as well.

6. Attach the Atmel-ICE programmer to the ReadyAVR board's JTAG port. Pay very close attention to the connector orientation. **THIS IS IMPORTANT! When in doubt, ask!** Next connect the Atmel-ICE to the PC using a micro-USB cable, and connect the ReadyAVR board to the PC using a USB A/B cable.

7. In **Atmel Studio**, double-click on the `Lab01.asm` filename in Solution Explorer to open the file for editing. Click on **Project | Lab01 Properties**… and choose **Tool** from the left-hand menu. Select your **Atmel-ICE** device as the debugger/programmer, and set the Interface to JTAG. Finally, select **File | Save All**… to save any unsaved settings.

8. Open **PuTTY** on the PC, choose **Connection | Serial**, and set the proper COM channel (use the Windows Device Manager to determine this, or run Dr. Nordstrom's `comports.bat` file found in the Lab01 repo and/or the class share). Set the remaining PuTTY communication parameters for 9600N81 (**9600** baud, **N**o parity, **8** data bits, **1** stop bit). Select **Terminal** and set **Local echo to "Force off"** and **Local line editing to "Auto"**). Consider strongly saving this PuTTY configuration for future use—very handy for future labs!

9. Click the green ▶ triangle on the AVR Studio menu bar to assemble, download, and run the code on the ReadyAVR. When it runs, you'll see a welcome message displayed on PuTTY.

10. Begin typing characters in the **PuTTY** window and notice that all printable characters are displayed and that non-printable characters (e.g. Enter, Backspace, etc.) operate as expected. Also, LED0 on the ReadyAVR board should blink whenever you type a character.

10. Experiment with type characters into PuTTY. Briefly explain what happens on screen and on the ReadyAVR board when the following keys are pressed:

   a. Letters, numbers, punctuation, spacebar, tab, and Enter

   b. Arrows, backspace, and ESC

   c. The RESET button on the ReadyAVR board

Study the Lab01.asm program (especially the comments!) and, using the space on the next page, **draw a flowchart with 10 or less shapes** that describes the logic of this echo server application. Be sure to capture the "infinite loop" nature of the code and use **legal flowchart symbols**. See the *gliffy* link below for flowchart help. Also see the *draw.io* tool link if you'd rather do the diagram electronically (draw.io is ***great***). In that case, just append your flowchart to this handout.

Report:

This is a **one week lab**. Due to the introductory nature of this lab, no separate lab report is required. However, you must turn in this completed lab worksheet via Canvas (PDF format) and your code via the lab's GitHub repo, **before the start of lab on January 29, 2021**. Have fun!

Helpful documents and links:

www.avrfreaks.net – A great user group and source of **much** information on AVR processors. Signing up for an account (free, and required to post messages) is highly recommended.

http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&t=70673 – A very good starting point for understanding and programming AVR microcontrollers. Very highly recommended.

http://www.youtube.com/user/AtmelCorporation – The Atmel channel. Many great videos.

https://www.gliffy.com/blog/how-to-flowchart-basic-symbols-part-1-of-3/ – Flowchart help.

http://draw.io – A great, free charting tool.