LAB 0: INTRODUCTION / PROGRAMMING, SIMULATING, AND EMULATING WITH THE ATMEL XMEGA

OBJECTIVES

- Before arriving to lab, you will write your first program for the Atmel **ATXMEGA128A1U** processor. You will understand how to simulate your program in *Atmel Studio* **before** receiving and then eventually programming your board in lab.
- Upon arriving to your lab, you will meet your TA. Get to know them, and understand the lab policies (as described in the *Lab Rules and Policies* document). After introductions, your TA will give your first lab quiz. Lab quizzes are related to the material presented in the lab.
- After the quiz, your TA will present a short demonstration on soldering; these skills will aid you in building your ***µPAD 1.4* (ATXMega128A1U) *Development Board*** kit (including four other PCBs, referred to as the single *memory base* PCB and the three *backpack* PCBs). After observing your TA demonstrate proper soldering techniques, you will be required to practice soldering on some practice boards.
- After assembling some of your board, you will program your processor and demonstrate the required program debugging/simulating/emulating using Atmel Studio and your uPAD.
- If time permits, you will continue to assemble your kit until the lab session is over.

REQUIRED MATERIALS

- As stated in the *Lab Rules and Policies*, do the following:
  - Submit your entire pre-lab report and **YOUR** asm file(s) through Canvas **BEFORE** entering the lab.
  - Print parts 14 a through d of the pre-lab report for submission to your TA as you enter lab. In **future** labs, your TA **may** not require you submit this paper part of the report, but assume that this is required unless your TA explicitly tells you that it is not required for future labs.
- Read/save the following documents:
  - *Out of the Box µPAD Assembly Guide*
  - *µPAD 1.4 Parts List*
  - *Atmel Studio Installation Instructions*
  - *Electronic Assembly handout*
  - *Class syllabus*
  - *Create Simulate Emulate on Atmel* tutorial
  - *AVR instruction set (doc 0856)*
  - *Assembly Language Conversion: GCPU to Atmel Assembly*
- Toolbox, DAD/NAD, and multimeter (from 3701)
- Soldering iron (available for use **in lab**)
- uPAD kit (distributed in lab)

DISCUSSION

The *Electronic Assembly handout* discusses how to solder, a technique to make a good electrical and mechanical connection between two parts.

If you have a soldering iron and solder from *Intro to ECE*, do **not** use it without explicit instruction from your TA. (The soldering irons from *Intro to ECE* have unregulated heating and may get too hot for our boards. Also, the solder that is distributed in this course is different than what we use. Mixing solder may lead to unreliable connections; poor soldering irons and soldering technique can burn PCB traces.)

Prior to the summer 2017 semester, we wire wrapped in 3744. Wire wrapping is another technique to make a good electrical and mechanical connection between two parts by wrapping wires around pins. We will no longer wire wrap in our course.

PRE-LAB PROCEDURE

Read **all** the documents listed in the following sections. Answer the pre-lab questions and complete your first pre-lab report that you will submit to your Canvas account (as specified in the *Lab Rules and Policies* document). Note that your first (Lab 0) pre-lab report will be very short and will only include a few of the items that will be included in all future lab reports. You **MUST** include sections a) through d) for all lab reports (even if there is not much to write for these sections). For this lab, answers to the prelab questions can be found in the posted lab documents.

You **must** make a **flowchart** or write pseudo-code **before** writing **any** program in this course. This will help you formulate a plan of attack for the code. The flowchart or pseudocode for parts B and C should be included (as stated in the *Lab Rules and Policies*, part 14 g) in your pre-lab report.

Note: Pre-lab requirements **MUST** be accomplished **PRIOR** to coming to your lab.

PART A - ATMEL Studio Installation

1. Go through the *Atmel Studio 7.0 Installation Instructions* to install the necessary software on your laptop (or tablet) computer.
2. Obtain a screenshot of Atmel Studio running on your computer that **also shows your full name** in big letters on the same screen. To do a screen shot in Windows, press Ctrl-PrtScrn (i.e., select Ctrl and PrtScrn at the same time). **Note:** The built-in *Snipping Tool* program in Windows is another great feature to print your screen. Copy your screen shot into MS Word (or a similar program) and include this in the Appendix (see section 14h) of your pre-lab report that you will submit to Canvas.

PART B - ATMEL Studio Tutorial

1. Go through the *Create, Simulate, and Emulate a Project* tutorial found on the website. Obtain a screen shot on your laptop of the results of step 11 that **shows your name** in big letters on the same screen. Save this screen shot in the Appendix of your pre-lab report.
2. Read the rest of the tutorial, given that you will perform the emulation portion of the tutorial during your lab.

PART C – Write, Debug/Simulate a Project

In this part, you will write an **assembly language program** (**`lab0.asm`**) that filters data from a given table in program memory. (It is required that you make a flowchart or write pseudo-code **before** writing **any** program in this course. This will help you formulate a plan of attack for the code.) The data is given in decimal, hexadecimal, binary, octal, or ASCII. This is to demonstrate that your assembler can read and interpret all these given formats. You do NOT need to convert these values to hex. For each byte in the data table, if the byte matches the filter condition, write the filtered data to the specified memory locations. Include this program in your pre-lab report (as stated in the *Lab Rules and Policies*, part 14 g). Your program will assume that the data is already placed in **program memory** prior to execution, i.e., you will use assembler directives like ".db" to put the constant table values into memory. See the following webpage for more information on using assembler directives:
http://www.avr-tutorials.com/assembly/avr-assembler-directives

Also see Section 4.5 of the below document:
http://mil.ufl.edu/3744/docs/XMEGA/doc1022_Assembler_Directives.pdf

The input table is shown in Table 1; you should place this table in **program memory** starting at **word address 0x9000**

Table 1: Memory Table

Data (ASCII)[1]	Data
NUL (.)	0x00
l	108
G	'G'
space	0b00100000
I	'I'
F	0106
:	':'
B	0x42
ñ	0b11110001
J	74
!	'!'
÷	0xF7
carriage return (.)	0b00001101
line feed (.)	0x0C

[1] The values in parenthesis are shown as visualized in a *Memory Window* of Atmel Studio.

(using assembler directives). Make sure you understand key differences between program memory and data memory. Do our processor's instructions reference memory locations in terms of words, or bytes? **Note:** ASCII is an 8-bit coded version of numbers, letters and symbols. An ASCII table can be found at www.asciitable.com.

Your program should filter the data given in the Table 1 based on the given criteria, and potentially store each of the resulting values in successive **data memory** locations, starting at address **0x3744**. You must use a pointers to accomplish this task.

1. If the table value is greater than or equal to 109 (109 = 0155 = 0x6D = 0b01101101), then it should be ignored, i.e., do **NOT** store anything, and move onto the next value.
2. If the value is greater than 0x39 (0x39 = 57= …), add 0x09, and then store the result.
3. If the value is less than or equal to 0x39 then
   - If bit 5 is **clear**, then the value should be ignored, i.e., do **NOT** store anything, and move onto the next value.
   - If bit 5 is **set**, do not alter the given value, and store it.

Your filtering should end after your program detects the "line feed" character (0x0C). The data in the original table should **not** be corrupted. The new table should end by adding the NUL character (0x00).

In order to make your code modular and re-locatable, use assembler directives. This will make it easy to change the location of both your input and output tables, the filter values (109, 0x39), and the end of table (EOT) value. Use the **`.byte`** assembler directive within a data segment (after a `.dseg` assembler directive) to reserve enough space for the output table.

A summary of the steps required for this lab are as follows.

1. Make a flow chart or write pseudo-code for the program that you will create. (It is required that you make a flowchart or write pseudo-code **before** writing **any** program in this course. This will help you formulate a plan of attack for the code.)
2. Create the assembly language program (**`lab0.asm`**).
3. Test your program using the Atmel Studio Simulator. Verify that the program works as specified.
4. Take a screen shot of a memory view window that shows the filtered values located at the correct addresses in data memory. Also, include any registers you used in the ***watch window*** (see https://mil.ufl.edu/3744/docs/Watch_Atmel_Studio.pdf) A watch window allows you to see the memory locations and registers change when stepping through the code. Make sure the screenshot **displays your name** in big letters on the same screen. Save this

screen shot in the Appendix of your pre-lab report (as stated in the *Lab Rules and Policies*, part 14h).

PRE-LAB QUESTIONS

1. What is the lab makeup policy if you miss a single lab?
2. Can you drop this lab if ... a) you overslept? b) a project for another class is due?
3. What minimum lab average is required to be **eligible** to pass the course?
4. How late can you arrive for lab and still be admitted? How late can you arrive for lab and still be allowed to take the lab quiz?
5. When soldering a wire to a pin, what should the soldering iron touch and what should the un-melted solder touch?
6. What are the key differences between program and data memory? See section 7 in the ATxmega128A1U manual.
7. What instruction(s) can be used to read from program memory (flash)? Can you use any registers with this instruction?
8. When using RAM (not EEPROM), what memory locations can be utilized for the `.dseg`? Why? What `.dseg` did you use in this lab and why?

PRE-LAB REQUIREMENTS SUMMARY

1. Answer all pre-lab questions.
2. Make a flowchart or write pseudocode before writing your program. This will help you formulate a plan of attack for the code.
3. Create your program for parts B and C (in a file on your computer) using the Atmel Instruction Set (doc8096). Include this program in your pre-lab report (as stated in the *Lab Rules and Policies*, part 13h).
4. Print the required parts of your pre-lab report to turn in to your TA (as stated in the *Lab Rules and Policies*, part 14a-14d).

Note: All pre-lab requirements **MUST** be accomplished **PRIOR** to coming to your lab.

IN-LAB PROCEDURE

Lab Rules and Policies / Introductions:

Upon entering lab, your TA will discuss the lab rules and policies (that you should have already read and agreed to as part of Homework 0) and then give you a general introduction to the laboratory and what will be expected from you for the semester.

Quiz:

After introductions, your TA will give your first lab quiz. Prepare by understanding the concepts used in this lab, do your own lab preparation, and read through your processor's entire Instruction Set (doc8096).

Parts Kit:

After your quiz, you will obtain your 3744 lab kit from your TA. These parts will be used in the labs throughout the semester. **Verify** that your kit has **all the parts** listed on the checklist provided. **Immediately notify** your TA about any missing parts. You will be responsible for any parts missing after you attend your first lab.

Board Construction:

Your TA will demonstrate proper soldering techniques. After you practice your soldering, if necessary (i.e., if this is not already done) perform step 1 of the *µPAD Assembly Guide*, i.e., solder the female header through which you program your µPAD and emulate your program. For every major component, solder two pins (on opposite corners of the component) and **have your TA check your work before completing the soldering for that component**. Do **NOT** be overconfident; failure to follow this procedure will result in unnecessary errors that might cost significant **time, money, and LAB POINTS)**. If you are not sure, ask first **before** soldering.

Simulation Demo:

Demonstrate (to your TA) that your program from Part C creates the correct table. The TA will ask you to change the data values and/or filtering conditions and then re-simulate your program. Your TA will also ask you to single step through your program and use breakpoints. Be prepared to answer questions about your program and the simulation/debug procedures.

Emulation Demo:

After simulating, you will now run your program on your processor.

Debugging with hardware is called **emulation**. Whenever possible, you should simulate your design first, before emulating; this eliminates any chance of mistaking hardware bugs for software bugs.

Load the program that you created for pre-lab Part C onto your board and verify that it functions properly. Demonstrate this emulation to your TA.

Board Construction, Continued:

After you finish this demonstration (or while waiting for the TA to check your work) continue working through the *µPAD Assembly Guide*. As stated above, be sure to check with your TA regularly to verify that you are correctly building your PCBs.

You are expected to remain in lab until you are finished with the required soldering (or until your lab ends). If you do not finish soldering, you need to finish before your next lab (Lab 1) begins; this means going to a TA office hour, or soldering at home.

REMINDER OF LAB POLICY

Please re-read the *Lab Rules & Policies* so that you are sure of the deliverables that are due prior to the start of your lab.