# Lab3: I/O Port Expansion

## OBJECTIVES

Explore and understand the implementation of memory-mapped I/O. Add an 8-bit input port and an 8-bit output port.

## REQUIRED MATERIALS

- EEL 4744 Board kit and tools
- 1 - 74HC574 8-bit 3-state D Flip Flop
- 1 - 74HC573 8-bit 3-state transparent latch
- From a previous lab:
  - 1 - 8-bit LED switch circuit
  - 1 - 8-bit switch circuit
- XMEGA documents
  - doc8331, doc8385, doc0856
- uTinkerer Schematics (on our website)
- Digilent Analog Discovery (DAD) kit

*YOU WILL NOT BE ALLOWED INTO YOUR LAB SECTION WITHOUT THE REQUIRED PRE-LAB.*

## PRELAB REQUIREMENTS

### REMEMBER:

You must adhere to the *Lab Rules and* Policies document for **every** lab.

Read the **ENTIRE** lab handout before starting this Lab. Study the examples on our website, especially *Input_Port.asm*.

For this lab and many future labs, you will need to create a Quartus schematic (bdf) file or VHDL file. Include Quartus files in your lab document as described in the *Lab Rules and Policies*. You can modify the Quartus schematic files to create complete wiring schematic (with pin numbers shown) for the designs described in this lab document.

## INPUT/OUTPUT PORTS

The XMEGA has several input/output (I/O) ports (A-F, H, J, K, Q, and R). The uTinkerer makes access to ports C-F efficient by supplying labeled wire wrap headers for each. Each of the I/O pins on all the XMEGA ports have additional features that we will use later in the semester. Therefore, we can "free up" these ports for their special use by adding **additional** memory-mapped I/O ports. Luckily, the XMEGA comes equipped with an External Bus Interface (EBI) peripheral that allows easy access to memory mapped peripherals that we will add in this and other labs.

1. Carefully read through the EBI section in the XMEGA AU Manual (doc8331, section 27). Make sure you understand the configuration registers, the chip select register, and how it generates the appropriate chip selects. In this lab we will configure XMEGA's CS0 (chip select 0) to enable external devices at addresses that will be specified later. In addition, we will use the CPLD to generate control signals for a tri-state buffer chip and D Flip-Flop (register) chip. These chips are used for input and output ports, respectively.
2. Your uTinkerer board was designed to work with an SRAM 3-PORT ALE1 EBI interface (see doc8331, Section 27.5.2 [and Figure 27-4] and Section 27.9 [and Table 27-4]). This means that address bytes 0 and 1 are time-multiplexed, i.e., they share the same pins. There is an 8-bit latch, already soldered to your board (just below the CPLD), that allows address latching using the ALE1 signal. Also see doc8385, Tables 33-7, 33-8, 33-9, for the port locations of the address, data, and control bus pins (in the columns marked SRAM ALE1).
3. In this lab you will configure the EBI and generate the necessary chip-enable/clock equations to implement one 8-bit input port and one 8-bit output port **both** mapped to address $**9000** and mirrored at all addresses from $9001 to $9FFF in data memory. Tri-state buffers **must** be used for input ports and flip-flop registers should be used for output ports.

# Lab3: I/O Port Expansion

4. How much memory is $**9**000 - $**9**FFF? You will select this as the SRAM address size in the EBI chip select register (EBI_CS0_CTRLA). You must also set the base address appropriately in the base address register (EBI_CS0_BASEADDR). You do not need to worry about wait states or other advanced features of the EBI. NOTE: There are 24 address lines (A23:0) in the data memory of the processor. However, your board only has access to address lines A15:0. As a result, for this and all future labs, you will only have to take address lines A15:0 into account when generating decode equations on your CPLD. (Hint: In this lab, you will NOT need to use address A15-A0 in any CPLD address decoding. All of the address decoding for this lab is done in the XMEGA's CS0.)
5. After CS0 has been properly configured, use Quartus to generate the combinatorial circuits for the input and output ports. Set /WE, /RE, /CS0 as inputs to the CPLD. Use /WE along with /CS0 to drive an output port and use /RE along with /CS0 to read from an input port. Take note of the activation levels in the XMEGA AU Manual (doc8385, Table 33-7) when designing your equations. **HINT:** The output port device's /OE signal can be tied to GND and input port device's LE can be tied to 3.3V.
6. If you have not already done so, solder a double row header into the J1 slot, inserting this header from the top of the board and soldering it on the bottom of the board. It is located at the top of your XMEGA board above the CPLD. The header will have the long legs facing up. The Altera USB Blaster will be connected to this header to program the CPLD, just as was done in EEL 3701.
7. Use jumpers on the double row header below and to the right of the reset switch. The jumpers are used to connect required signals (/WE, /RE, etc) to the CPLD. Figure 1 is a snapshot from the uTinkerer schematic document (available on our website). The numbers boxed in red correspond to the PIN_# on the CPLD. These are the pins that you need to select in the Quartus pin planner so that the desired jumpered signals are connected to the CPLD. Once these pins are verified, program the CPLD. If you start encountering issues with the processor while programming or with various code that worked previously, double check both your wiring and the Quartus design. If there is a problem, it is likely caused by a data bus or address bus conflict, which interferes with the processor's normal operation. **NOTE:** Make sure you set the default value of the pins as input tri-stated before you program the CPLD.

## INPUT PORT

8. After the CPLD is programmed, use the 74HC573 datasheet to **create a wiring schematic** (by hand or with Quartus) for the input port using the 74HC573 8-bit 3-state

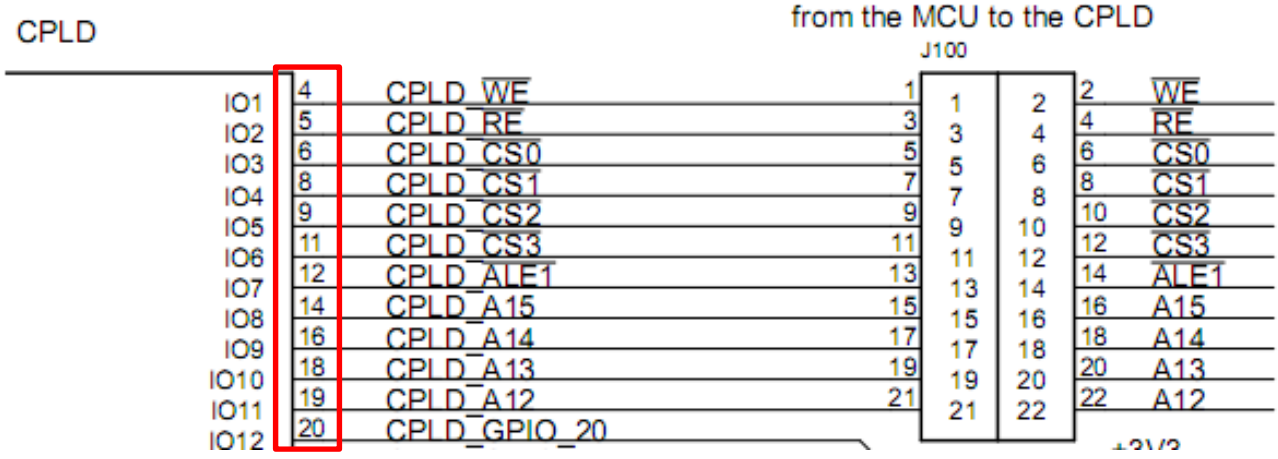


**Figure 1**: CPLD pins utilized to connect to XMEGA. (See also the uTinkerer schematics on our website (uTinkerer-v1.1-sch.pdf).

transparent latch (acting as an 8-bit 3-state device). **Use labels instead of wires to show connections**. (I encourage you to make a computer-generated schematic that you can add to in subsequent labs.) The schematic should show connections to the data bus header and the CPLD header. Do not draw wires. Use pin labels instead. **Include all pin numbers**. You can add the pin numbers by hand on your printouts. (Note: The 74573 is not available in Quartus, but the operation of the 74373 is identical and can be used. The 74373 can be found in the parts library under `others | maxplus2`. The pinouts of the 74‘373 and the 74‘573 are **different**.)

**9.** Wire the input port using your wiring schematic.
   **a.** If you have not already done so, you must solder the 74’573 surface mount chip into place. Place the 74’573 in the left surface mount position just above your LED bank, as shown in Figure 2. Begin by soldering the two corner pins to chip in place. Add two single row 10-pin headers on either side (above and below) the 74’573. You will wire wrap to these header pins. Solder a 0.1μF bypass capacitor between the header pins corresponding 74’573’s Vcc and GND pins, i.e., pins 10 and 20 (as shown in Figure 3). Before soldering the capacitor, trim the leads appropriately. Be careful that the capacitor leads will not touch any non-desired pads or pins.
   **b.** As always, double wrap the VCC and GND pins. Finally, wire wrap the remainder of the signals as shown on your circuit diagram.

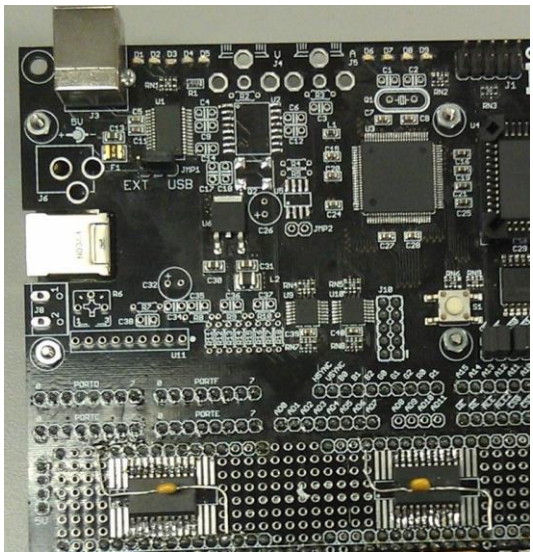


**Figure 2**: Placement of 57’573 (bottom left) and 57’574 (bottom right) on uTinkerer.

**Note:** Power and ground signals are usually distributed using a **bus** to every chip. In this course and in future projects (ECE Design 2, for example) you should solder power bus lines or make wide traces and **NOT** use wire wrap.

**10.** Once the input port is finished, verify that your board still programs via JTAG. If AVR Studio cannot properly read the signature with the programmer, there may be a crossed or loose connection. Verify all of your connections using your multimeter.
**11.** Now that the board programs with the newly wired input port, remove the wire wrap from the switch bank from a previous lab and wire-wrap it to your new input port.
**12.** Verify that your board still programs. If the board does not boot, check the switch bank connections using a digital multimeter.
**13.** Write a short test program to read the value of the port and write it to the LEDs from Lab 2. (Submission of this program is not required, but you will likely utilize this code again later.) If the input port does not work, review the above steps.

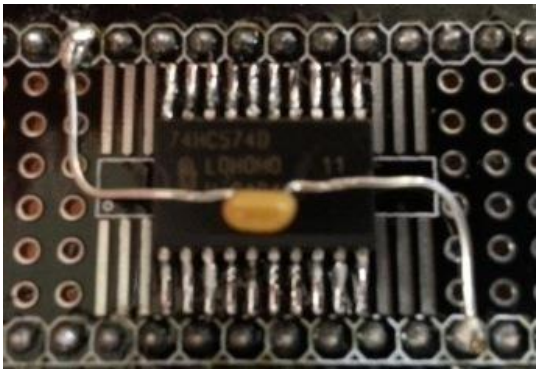

**Figure 3**: Example capacitor placement across a chip.

# Lab3: I/O Port Expansion

## OUTPUT PORT

**14.** Once the input port works, use the 74HC574 datasheet to create a Quartus design for the output port using this 8-bit 3-state D flip-flop. Add your circuit diagram to the one with the input port. (As the semester continues, we will add to this schematic many times, so I suggest that you generate this with a computer.) The schematic should show connections to the data bus header and the CPLD header. Do not draw wires. Use pin labels instead. **Include all pin numbers**. You can add the pin numbers by hand on your printouts. (Note: The 74574 is not available in Quartus, but the operation of the 74374 is identical and can be used. The 74374 can be found in the parts library under `others | maxplus2`. The pinouts of the 74‘374 and the 74‘574 are **different**.)

**15.** Wire the output port using your schematic. In lab 2, you should have already soldered the 74’574 and capacitor in the appropriate location, as repeated in part a below.

a. Place the 74’574 surface mount chip in the middle surface mount position (to the right of your 74’573). Solder the 74’574 as described above for the 74’573. After the chip is in place, solder a 0.1μF bypass capacitor between the header pins corresponding to the 74’574’s Vcc and GND pins, i.e., pins 10 and 20. Add two single row 10-pin headers on either side (above and below) the 74’574.

b. Finally, wire wrap the remainder of the signals as shown on your circuit diagram.

**16.** Once the output port is finished, verify that your board still programs via JTAG. If it does not, verify your connections for the output port.

**17.** If the output port does not work, review the above steps.

**18.** Disconnect the LED bank circuit from lab 2 and connect it to your new output port.

**19.** Verify that your board still programs after moving the LEDs. The port can be tested further by writing a value to the output port with a short program. The output should appear on the LED bank.

## I/O SOFTWARE

**20.** Write a program that reads the input port, places the input port contents on the output port, then **shift the bits on the output port right** the output port contents **once every two seconds** (approximately). After shifting 8 times, read the input port again and repeat the shifting process. Repeat this forever. This is the only program you must submit for this lab.

## PRE-LAB QUESTIONS

1. What is the advantage of mirroring the input and output ports to $**9**000 - $**9**FFF (also known as partial address decoding)? What would be necessary if you wanted to only place the ports at 0x**9**000 and no place else?
2. Write the address decode equation to put the input and output ports at addresses 0x4000-0x6FFF.
3. What is the complete range of external memory on the XMEGA?
4. The uTinkerer is configured to use the EBI in SRAM ALE1 mode, which does not give us address bits A23:16. Which mode(s) can be used to access these higher address bits? Describe the change(s) necessary to use one of these modes. What additional hardware is needed, if any?

## PRE-LAB REQUIREMENTS

1. Answer all of the pre-lab questions.
2. Configure EBI for 0x**9**000 – 0x**9**FFF memory range.
3. Determine the chip-enable equations for the input and the output ports.
4. Program the CPLD.
5. Build the 8-bit input port on your board

# Lab3: I/O Port Expansion

6. Build the 8-bit output port on your board
7. Write and debug the required program.
8. ~~Take a screenshot of the external memory in the memory view with the correct address range.~~
9. Email the required documents as specified in the document *Lab Rules and Policies*

## IN-LAB REQUIREMENTS

1. Demonstrate that the input port works using the switches.
2. Demonstrate that the output port works using the LEDs.
3. Demonstrate the proper functioning of your I/O Software program.

Preparation for Future Labs:

If you have not already done so, solder the 32k x 8 SRAM in place, as shown in Figure 4.

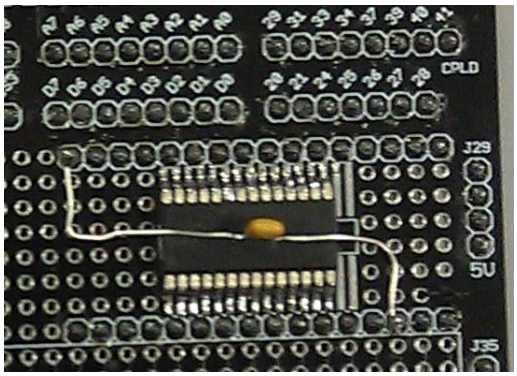


**Figure 4**: SRAM placement on uTinkerer.

## DEBUGGING TIPS:

1. You will **not** able to use the multimeter (as a voltmeter) to measure signals on the memory bus. They are changing too quickly. The voltmeter will show the average value. This **can** be utilized for debugging as follows.
   a) Write a very short program segment that modifies the desired bus appropriately.
   b) Run this program in a continuous loop.
   c) The average voltage of the particular pin will appear on meter. This will be different than the voltage when the bus is not being utilized.
2. Connect the pin to your analog discovery board to observe the waveform using the DAD oscilloscope function. In lab 4, you will learn how to use the DAD LSA (logic state analyzer) function. The LSA can view the logical value of up to 16 pins at the same time
3. Make sure that you set all unused pins in Quartus to be tri-stated input. If you connect signals with jumpers and the do **not** use them, your program may not work.
4. Make sure you understand the required PORT direction bits settings. If they are not set correctly, your code will not work.
5. The following signals are active low: /WE, /RE; while ALE1 is active-high (even though it is shown in some of the manuals as /ALE1). See Figure 27-4 in doc8331.
6. Read the EBI documentation regarding I/O pins (section 27.9). It explains how to handle active low and active high signals.

Note that it is **not** generally possible to view external memory-mapped devices in Atmel Studio. I posted a document at http://mil.ufl.edu/4744/docs/External_Memory_Viewing.pdf, that describes a "trick" that sometimes allow us to view external memory in Atmel Studio. It may work for you, but it has not worked for many people over the last few semesters. (If you try this "trick," please let Dr. Schwartz know if it works or not.)