OBJECTIVES

- Understand how to utilize an external bus interface (EBI) system effectively.
- Design a memory-mapped hardware expansion consisting of external I/O ports and an external SRAM component.

LAB STRUCTURE

In this lab, you will explore and utilize the External Bus Interface (EBI) system within the *ATxmega128A1U*, which ultimately provides a generic, configurable interface for connecting external components to the data memory space of the microcontroller, by way of the data bus, address bus, and control signals provided by the microcontroller.

In § 1, you will research various information so that you may design a hardware expansion consisting of a series of external components; namely, an input port, an output port, the SRAM component available on the *OOTB Memory Base*, and some switch and LED circuits. In § 2, you will begin to implement and test the design created in § 1. More specifically, you will physically construct the relevant input and output ports, connecting the input port to a series of switch circuits and the output port to a series of LED circuits, and then create an assembly program to test this portion of the design. Finally, in § 3, you will finish implementing and testing the hardware expansion by connecting the relevant SRAM to the data space of your microcontroller and writing another assembly program.

REQUIRED MATERIALS

- Atmel *ATxmega128A1U* AU Manual (doc8331)
- Atmel *ATxmega128A1U* Manual (doc8385)
- *OOTB µPAD v2.0* kit, with accompanying schematics
- 1 – 74HC573 8-bit 3-State Transparent Latch
- 1 – 74HC574 8-bit 3-State D Flip-Flop
- 1 – DIP switch component
- 1 – DIP LED component
- Some programmable logic device (PLD) kit given in *3701* and related software
- Some *Analog Discovery* (*AD*) kit, along with the *WaveForms* software
- `sram data asm.txt`

SUPPLEMENTAL MATERIALS

- The *3701* Software/Docs webpage, for PLD and Quartus information

PRE-LAB PROCEDURE

REMINDER OF LAB POLICY

You must re-read the *Lab Rules and Policies* before submitting any pre-lab assignment and before attending any lab.

1. INTRODUCTION TO EBI AND HARDWARE EXPANSION

In this section, you will begin researching the relevant information regarding the EBI system.

1.1. Read § 27 (*EBI – External Bus Interface*) of the 8331 manual to learn how to configure and utilize the EBI system within the *ATxmega128A1U*. Additionally, peruse any and all other related course content, e.g., relevant lectures, supplemental videos, etc.

PRE-LAB EXERCISES

i. For each SRAM configuration within the EBI system of the *ATxmega128A1U* microcontroller, to which address lines do you have external, physical access? Additionally, for the *SRAM 3-PORT ALE1* configuration, is it possible to have external, physical access to any address lines above A_{15}? Why or why not?

ii. Describe what performing full address decoding and partial address decoding signifies. Provide examples of both types for [1] an SRAM chip and [2] either an input port or output port.

iii. In theory, how many 8-bit I/O ports could be mapped to the *external* data memory space of an *ATxmega128A1U* microcontroller, assuming that at most one port utilizes any particular address range? Explain your answer, utilizing your response from Exercise ii.

Now, for the remainder of this section, you will design a memory-mapped hardware expansion for the *OOTB µPAD*. Later in this lab, you will physically construct the relevant design using the EBI system within your microcontroller and some other appropriate components.

The first two components to be added by the hardware expansion will be an 8-bit input port and an 8-bit output port. Designing external input and output ports is advantageous as it allows an increase in the number of external entities to which a microcontroller can connect, e.g., backpacks, base boards, etc., and it also allows any pre-existing internal I/O ports to be utilized by other internal peripheral systems, e.g., TC, EBI, and other systems that we will explore throughout this semester. To see which ports, and specifically which pins, are utilized by internal peripheral systems, see § 33.2 (*Alternate Pin Functions*) within the 8385 manual.

For this lab, upon constructing an external 8-bit input port and 8-bit output port, you will connect eight switches within a DIP switch package (along with the appropriate resistors to make switch circuits) to the input port and eight LEDs within a DIP LED package (along with the appropriate resistors to make LED circuits) to the output port. To learn more about the physical construction of I/O ports, see our class notes/discussion and your notes from EEL3701.

Following these two components, you will add an external memory device to the microcontroller via the data memory space. More specifically, you will connect the microcontroller to the static random-access memory (SRAM) chip available on the *OOTB Memory Base*. See Appendix A for some general information regarding SRAM components.

1.2. To know how to perform the relevant hardware expansion for this lab, complete the *Canvas* quiz titled *"Lab 4 – Hardware Expansion"*. (Note that this quiz is separate from, but will be very similar to, the lab quiz that you will take following the due date of this lab assignment.) This "pre-lab quiz" provides a series of steps that should generally be performed when designing a hardware expansion. Overall, this quiz will account for a sizable portion of the grade for this lab assignment. An unlimited number of attempts will be given to complete the quiz, and only the last (hopefully highest) score will be considered. Note that the due date for this quiz is several days before the lab is due. Failure to turn it in by its due date/time will result in a five-point penalty (of the 100 point available in the lab). The reason that it is due earlier than the rest of the lab is that we think that this will help encourage you to start early and thus (hopefully) finish this lab before the due date/time. **You must verify that all your solutions for the quiz are fully correct before continuing with this lab assignment.**

PRE-LAB EXERCISES

iv. Assume that an SRAM component with the same size as the one on the *OOTB Memory Base* has to be added to a different computing system containing an *ATxmega128A1U* but no *OOTB Memory Base* (i.e., this is not the *OOTB µPAD* computing system), with the first address of the SRAM starting at address 0x17 C000, instead of the address specified in the *"Lab 4 – Hardware Expansion"* quiz. Design, on paper, a hardware expansion for this new system, utilizing the same structure of steps laid out in this quiz. Use the same "overall" memory-mapping constraints presented within the quiz.

v. Assume that some external 128 KB SRAM must be fully mapped to the data memory space of the *ATxmega128A1U*, with the first memory location of the SRAM corresponding to data memory address 0x320000. Design, on paper, a hardware expansion for this new system, utilizing the same structure of steps laid out in the relevant pre-lab quiz. Use the same "overall" memory-mapping constraints presented within this quiz. Hint: Consider how many address signals are needed to make the SRAM fully addressable, and then consider how you will gain access to all of these signals.

vi. Assume that some 8-bit input port should be accessible via the 256 consecutive addresses starting at 0x20580 within

the *ATxmega128A1U* data memory space. Design, on paper, a hardware expansion for this port, utilizing the same structure of steps laid out in the relevant pre-lab quiz. Use the same "overall" memory-mapping constraints presented within this quiz

vii. Do the same as in pre-lab exercise vi, but assume that the 8-bit input port should only be accessible via data memory address 0x4744.

viii. Assume that some external 1 MB SRAM must be fully mapped to the data memory space of the *ATxmega128A1U*, with the first memory location of the SRAM corresponding to data memory address 0x100000. Design, on paper, a hardware expansion for this new system, utilizing the same structure of steps laid out in the relevant pre-lab quiz. Use the same "overall" memory-mapping constraints presented within this quiz, except that which is concerning the usage of the *SRAM 3-PORT ALE1* EBI configuration – consider the use of some other EBI configuration.

2. INTERFACING WITH EXTERNAL I/O PORTS

In § 1, you designed a memory-mapped hardware expansion containing both an 8-bit input port and 8-bit output port. In this section, you will utilize your design to physically construct the relevant I/O ports, and then write an assembly program to test your design.

2.1. If you have not already done so, connect the *µPAD* to the *OOTB Memory Base*.

2.2. Use a breadboard, a 74HC573 8-bit 3-state transparent latch, a 74HC574 8-bit 3-state D flip-flop chip, a dual-inline package (DIP) switch component, a DIP LED bank, some programmable logic device (PLD) kit given in *3701*, the relevant *OOTB* components, and anything else appropriate to build the 8-bit input and output port circuits designed in § 1. Study the relevant schematics before building your circuits, so that you do not duplicate circuits already built. For any external component constructed with a PLD, remember to submit any relevant *Quartus* project(s), each archived to a `.qar` file, as specified in item ii of § 5 within the *Lab Rules and Policies* document. Additionally, for all PLD designs, either [1] include a screenshot of the relevant block diagram file (BDF) within the Appendix of your pre-lab report or [2] include the relevant hardware description language (HDL) code within the "Program Code" section of your pre-lab report. (You are not expected to have both [1] BDFs and [2] HDL code, only one of them.)

2.3. Create an assembly program **`lab4_2.asm`**. Within the program, first configure the EBI system such that the I/O ports will only be enabled for addresses within the relevant memory range specified in the *"Lab 4 – Hardware Expansion"* quiz. Then, continually write [1] the digital value specified by each DIP switch connected to your external input port to [2] a corresponding LED on the LED bank connected to your output port.

3. INTERFACING WITH EXTERNAL SRAM

In this section, you will utilize the design created in § 1 to connect the SRAM component available on the *OOTB Memory Base* to your microcontroller, and then you will write an assembly program to test your design.

3.1. Verify that the *OOTB Memory Base* is connected to the appropriate components.

3.2. Create an assembly program **`lab4_3a.asm`**. Within the program, first configure the EBI system such that the external SRAM chip will only be enabled for addresses within the relevant memory range specified in the *"Lab 4 – Hardware Expansion"* quiz. Then, sequentially write the data available in the `sram data asm.txt` file (available on our course website) to the external SRAM, starting at the first address within the SRAM. After this, to be able to verify that all the relevant data was correctly stored, your program must sequentially read back **all relevant data** (i.e., only that which was stored) at a rate of one byte per 500 ms, writing each read byte of data to the external I/O port constructed in § 2. Utilize a timer/counter to achieve the necessary delay.

3.3. Create an assembly program **`lab4_3b.asm`** to allow you to readily measure both [1] a *complete* EBI write cycle and [2] a *complete* EBI read cycle when communicating with the external SRAM. (You will use an *Analog Discovery* kit and the *WaveForms* software to measure these write/read cycles, as discussed in the following item of this document.) For simplicity, it is recommended that the program simply perform an infinite loop that writes and reads from at least two *meaningful* SRAM locations. (Recognize that the ALE signal will only be set for memory accesses that require the latched "middle-byte" address value to change, as described in § 27.5.5 of the 8331 manual. Thus, to be able to measure both a *complete* EBI write cycle and a *complete* EBI read cycle, at least two separate memory addresses will need to be accessed.) Additionally, when writing data to the SRAM, write some **nonzero four-bit value**. It should be clear why this requirement was chosen after reading the following item of this document.

3.4. Use the relevant *OOTB* components, *Analog Discovery* kit, and the *Logic* feature of the *Waveforms* software to record both a *complete* EBI read cycle and *complete* EBI write cycle for the SRAM component. Include at least the following signals in your waveform: an appropriate chip select (*CS*) signal, *WE*, *RE*, *ALE1*, A_7-A_0, and D_3-D_0. (There is, of course, a limit to how many digital signals can be measured by the *Analog Discovery*. This is why only a

subset of the data bus is to be measured.) When using the *Logic* feature, make sure to group the appropriate signals into busses, and use the appropriate time base for your displayed waveforms such that only a single read/write cycle, or, alternatively, a single read/write cycle pair, is visible. Take at most two screenshots to capture the relevant measurements. **Annotate the screenshot(s), meaningfully describing every change in signal or bus.**

NOTES:

- The `sram data asm.txt` file contains an XMEGA AU assembly definition for a read-only array of data. Since this text file is fairly large and has no information directly relevant to the programmer, do **NOT** include the text within the `sram data asm.txt` file directly in your asm file, but rather use an *include* assembler directive to include this file in your asm file.
- When EBI applications are to be written with the "C" programming language later in the semester, certain macro functions defined within the *ebi_driver.h* file provided on the class website must be used whenever a RAMP register is needed to access a memory location.

PRE-LAB PROCEDURE SUMMARY

1) Answer pre-lab exercises, when appropriate.
2) In § 1, learn about the EBI system within the *ATxmega128A1U*, and complete the *Canvas* quiz titled *"Lab 4 – Hardware Expansion"*.
3) In § 2, use the relevant schematic designed in § 1 to physically construct the appropriate external input and output ports, connecting them to your microcontroller. Additionally, connect the input port to a set of DIP switches and the output port to an LED bank. Write an assembly program to test both systems. For any external component constructed with a PLD, either [1] include a screenshot of the relevant block diagram file (BDF) within the Appendix of your pre-lab report or [2] include the relevant hardware description language (HDL) code within the "Program Code" section of your pre-lab report. Remember to submit any relevant *Quartus* project(s), each archived to a `.qar` file, as specified in item ii of § 5 within the *Lab Rules and Policies* document.
4) In § 3, connect your microcontroller to the external SRAM chip located on the *OOTB Memory Base*. Write an assembly program to test this interface. Write another assembly program to measure both a complete EBI write cycle and a complete EBI read cycle for the SRAM component. Take at most two screenshots to capture the relevant measurements. Annotate the screenshot(s), meaningfully describing every change in signal or bus.

APPENDIX: STATIC RANDOM-ACCESS MEMORY (SRAM)

By design, static random-access memory is volatile, i.e., information stored within these devices is lost if power is removed from the chip, although, unlike dynamic random-access memory (DRAM), any information stored is retained simply by maintaining power. Additionally, most SRAM chips are generally designed to have a parallel bus interface for address and data signals, where the encoded binary value of the address signals specifies which address within the SRAM is to be written to or read from, and where the encoded binary value of the data signals specifies either the data available within the currently specified SRAM address or the data received from a connected device.

When interfacing with an SRAM component, there are often a few control signals to consider. For any SRAM with a bi-directional data bus, the direction of data transfer is generally specified by signals known as output enable (often written as *OE*) and write enable (often written as *WE*). When an *OE* signal is true, an SRAM is to output data from within an SRAM address currently specified by the address bus, and when a *WE* signal is true, an SRAM is to write any data currently specified by the data bus to an SRAM address currently specified by the address bus. (If both *WE* and *OE* are true, it is often that *WE* is made to have priority.) Finally, to enable an SRAM component, it is generally the case that a third control signal, often known as a chip select (*CS*) or chip enable (*CE*), is used.