SUMMARY

COURSE LOGISTICS

GENERAL THEORY

Analog-to-Digital Converter (ADC) Systems

Microchip/Atmel Studio

Asynchronous Serial Communication

AVR Assembly Programming

FREQUENTLY ASKED QUESTIONS

COURSE LOGISTICS

FAQ1. *Beyond completing the required assignments, how should I best study for this course?*

ANS. *Ultimately, it is the responsibility of the individual to figure out how they themselves best learn or study something. However, as a start, it will likely be helpful to try to do all of the following, in the same order provided:*

[1] Carefully study all relevant lecture and lab material(s) until you believe that you have a firm grasp on both theory and application. Do not be afraid to revisit some material even after you believe that you thoroughly understand it.

[2] Solve relevant problems provided within the "Practice" page of our course website. Try to impose an appropriate time limit on yourself while solving these problems, e.g., that which is given in the prompt of each problem. Verify any solutions that you are unsure about with a course staff member.

[3] Experiment with applying the relevant material(s) in new ways and try to come up with your own applications. Push yourself to think outside the box. In general, the ability to identify/solve new problems is crucial for an engineer, but note that to refine this skill normally entails continual development; do not expect too much from yourself right away, but also do not wait to start practicing.

Now, here comes the real challenge. While the doing any of above, try to pretend as if you are teaching someone else how and why to do what you are doing. In other words, for anything that you do, try to walk yourself through a detailed explanation of what you are doing, and why. This is often a main catalyst for achieving a true understanding of something — if you can get someone, even just yourself, to thoroughly understand a concept through your own instruction, you almost assuredly have a strong understanding of this concept.

FAQ2. *How should I best prepare for a lab quiz?*

ANS. *First and foremost, make every effort to fully understand the relevant lab material(s). Lab quizzes are made strictly for the purpose of verifying whether or not you individually (emphasis on "individually") understand the relevant lab material(s).*

Do not just memorize the particular examples or requirements of some lab assignment, but actually attempt to understand the underlying concepts. Try to understand how you may be able to extend what was described/required within some lab assignment, and then actually try to implement such extensions.

If you need extra practice problems, you may wish to refer to the "Practice" page of our course website.

If you need further guidance beyond this, seek help from a course staff member.

FAQ3. *Will the use of personal notes be allowed during a lab quiz?*

ANS. *No, but access to the relevant documentation will be provided.*

Recognize that the main purpose of a lab quiz is to determine whether or not an individual (emphasis on "individual") understands how to apply the relevant content. Providing access to personal notes would freely allow an individual to blindly copy and paste work and other content, which would make it impossible to determine [1] if the individual actually completed the relevant work themselves and [2] if the individual even understands what they have included in their notes.

Overall, you should be capable of understanding and remembering all of the content surrounding a single lab, and you should be capable of remembering all relevant programming language syntax. For practical quizzes and the final exam, where there is a much larger set of content that applies, some form of personal work or notes will be allowed.

FAQ4. *Are there additional resources for learning the relevant course content?*

ANS. *Yes! Look through the "Examples" and "Videos" pages of our course website, as well as the "Modules" page of our Canvas site.*

FAQ5. *How should I submit lab assignments on Canvas?*

ANS. *Refer to the lab rules and policies. For convenience, you may access them via the following link.*

https://mil.ufl.edu/3744/admin/lab_rules_and_policies.pdf

Make sure to use the required lab submission template, available at the top of our lab webpage.

FAQ6. *How should I submit homework assignments on Canvas?*

ANS. *In regard to submissions, treat homework assignments just as lab assignments, and refer to the relevant lab rules and policies. For convenience, you may access the lab rules and policies via the following link.*

https://mil.ufl.edu/3744/admin/lab_rules_and_policies.pdf

Make sure to use the required homework submission template, available at the top of our homework webpage.

FAQ7. *Should I include code files within an assignment submission on Canvas?*

ANS. *If the assignment requires that some form of code be written, yes. Treat such assignments as lab assignments and refer to the relevant lab rules and policies. For convenience, you may access the lab rules and policies via the following link.*

https://mil.ufl.edu/3744/admin/lab_rules_and_policies.pdf

FAQ8. *When creating a report for some assignment, when should I take a screenshot of something?*

ANS. *Only when the assignment explicitly requires you to do so. Make sure to attach this screenshot within an appendix of the relevant report, as documented in the lab rules and policies. For convenience, you may access the lab rules and policies via the following link.*

https://mil.ufl.edu/3744/admin/lab_rules_and_policies.pdf

GENERAL THEORY

Analog-to-Digital Converter (ADC) Systems

FAQ9. *Is there a mistake in Table 28-13 in the XMEGA AU Microcontroller Manual?*

ANS. *Yes! The caption should say CONVMODE=0 (not CONVMODE=1).*

FAQ10. *For an ADC system with n-bit precision, is the linear transfer equation based off a digital value range of 2^n values, or 2^n+1 values?*

ANS. *Generally, 2^n+1 values. Obviously, an ADC can only generate 2^n digital values, so a digital value with all bits being set would generally correspond to some voltage slightly less than the maximum voltage, rather than exactly the maximum voltage.*

For more information, it may be helpful to refer to the following sources:

[1] https://masteringelectronicsdesign.com/an-adc-and-dac-least-significant-bit-lsb/

[2] http://ww1.microchip.com/downloads/en/appnotes/atmel-8456-8-and-32-bit-avr-microcontrollers-avr127-understanding-adc-parameters_application-note.pdf

[3] http://www.cse.psu.edu/~kxc104/class/cse577/11s/lec/lecture/ADCsamplingCKTs.pdf

For this course, if you are asked to derive a linear transfer equation for an ADC (or DAC) system and it is not clear if the equation should be based off of a digital value range of 2^n values or 2^n+1 values, an answer that follows from either of these should be fine, i.e., unless otherwise noted, this one additional digital value is not crucial for the purposes of this course.

Microchip/Atmel Studio

FAQ11. *How should I create an "AVR Assembler Project" within Microchip/Atmel Studio?*

ANS. *Refer to the following two links.*

[1] https://mil.ufl.edu/4744/docs/Create_Simulate_Emulate_Atmel.pdf

[2] https://youtu.be/GHIpDQMssAo

FAQ12. *How should I create a "C Executable Project" within Microchip/Atmel Studio?*

ANS. *Refer to the following two links.*

[1] https://mil.ufl.edu/4744/docs/Create_Simulate_Emulate_Atmel.pdf

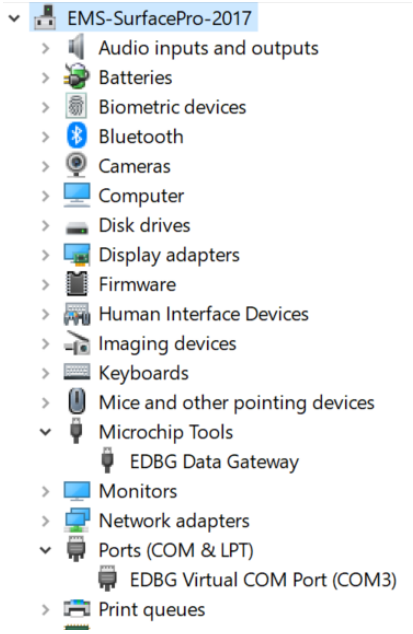
[2] https://youtu.be/gVYo4Awlp2w

FAQ13. *I am having trouble getting Microchip/Atmel Studio to recognize my hardware under the "Tool" section of "Project Properties". What should I do?*

ANS. *Perform the following, in the same order provided. If you still have issues after doing so, contact a course staff member.*

[1] Ensure that your hardware is connected to your computer via the relevant USB cabling.

[2] Try opening up the "Device Manager" program of the Windows operating system and see whether or not the hardware is listed under a section titled "Atmel"; it should be labeled something such as "EDBG Data Gateway". (See the Figure to right.)



[3] If it is listed under "Atmel" (it should be), right click on the device listing and select "Uninstall device". Then, in the dialog box that pops up, select the "Delete the device driver for this device" checkbox, and then click "Uninstall". Also, if the EDBG Virtual COM Port is listed under Ports (COM & LPT), uninstall and delete that as well.

[4] Once the device has been uninstalled, make sure that Microchip/Atmel Studio is closed and disconnect the hardware from your computer by unplugging the relevant USB cabling.

[5] Restart your computer.

[6] Once you restart your computer and log back into Windows, connect the hardware to your computer.

[7] Open up Microchip/Atmel Studio.

At this point, Microchip/Atmel Studio should be able to detect your hardware. (However, again, if you experience further issues after performing these steps, contact a course staff member.)

FAQ14. *Is there a reason for why the time measured by the "Stop Watch" feature of Microchip/Atmel Studio (within the "Processor Status" debug window) can be quite different than that of some time measured by WaveForms?*

ANS. *The simulator and emulator within Microchip/Atmel Studio are imitators; more specifically, they are software programs that utilize theoretical calculations to estimate the execution of a created computer program. When some program is actually executed within the physical world, there will always exist some discrepancy.*

FAQ15. *With a "Memory" debug window, is there any way to see what is stored within external memory?*

ANS. *There is no readily available manner for you to view external memory with a Memory window. Instead, it may be helpful to [1] write a program that reads data from external memory and places this data into the internal SRAM, and then [2] use a Memory window to view the contents of the internal SRAM.*

FAQ16. *How and why do I turn off the optimizer tool for the "C" compiler of Microchip/Atmel Studio?*

ANS. *Refer to the following video link.*

https://youtu.be/gVYo4Awlp2w

FAQ17. *When I delete a breakpoint in Microchip/Atmel Studio, it appears to be there. How can I fix this?*

ANS. *Go to the "Debug" dropdown menu and select "Delete All Breakpoints."*

Asynchronous Serial Communication

FAQ18. *Why would my serial terminal program be displaying garbage when the microcontroller sends data?*

ANS. *There could be several possible reasons for this, but it is most probable that the baud rate has not be properly implemented by either the serial terminal program or the microcontroller. If you are confident of your settings, try utilizing a different combination for the 'BSCALE' and 'BSEL' values. Another possibility is that you have different parity settings on the microcontroller and serial terminal program.*

AVR Assembly Programming

FAQ19. *How should I successfully write an application with the AVR assembly language?*

ANS. *This takes practice, and it ultimately depends on the application at hand. To get started for the purposes of this course, refer to the "Examples" page of our course website, as well as to the following video link.*

https://youtu.be/P78lWaRaY14

[1] Break down the application into small and testable subroutines or program fragments. The key is to make each subroutine do only one or two things, and for each to be as independent as possible from any other subroutines.

[2] Make a main routine to call the first subroutine. Test it. If it works, add another subroutine.

[3] Add a second subroutine to the previously designed main and first subroutine. If it works, continue this process until you have solved the required application.

FAQ20. *Where should I access the 'ATxmega128A1Udef.inc' include file?*

ANS. *After an "AVR Assembler Project" has been "built" at least once (e.g., by navigating to "Build | Build Solution", via the top toolbar of Microchip/Atmel Studio), this include file should be accessible by way of the "Dependencies" tab of the "Solution Explorer" window. (To open the "Solution Explorer" window, navigate to "View | Solution Explorer", via the top toolbar of Microchip/Atmel Studio.)*

In addition to accessing the file within Microchip/Atmel Studio, this file may also be accessed by way of our course website, via the following link.

https://mil.ufl.edu/4744/docs/XMEGA/ATxmega128a1udef.inc.txt

FAQ21. *Why is the assembler of Microchip/Atmel Studio not recognizing the symbols 'RAMPX', 'RAMPY', and 'RAMPZ'?*

ANS. *As defined within the 'ATxmega128A1Udef.inc' include file, these registers are addressable by the symbols 'CPU_RAMPX', 'CPU_RAMPY', and 'CPU_RAMPZ', respectively.*

Become accustomed to searching through the 'ATxmega128A1Udef.inc' file. For information on how to access this file, refer to the previous FAQ.

Other CPU_ registers include 'CPU_SREG' (the status register) and 'CPU_SPL' and 'CPU_SPH', the low and high bytes of the stack pointer.

External Bus Interface

FAQ22. *Regarding the EBI system within the ATxmega128A1U, why and how is the EBI chip select hardware able to compare against address signals A[23:8], with it only being possible to specify the twelve most-significant bits of a base address (i.e., those that correspond to A[23:12]) via the BASEADDR registers?*

ANS. *Although the BASEADDR registers only allow that a programmer specify the most-significant twelve bits of a base address (where these bits are meant to be continually compared against address signals A[23:12]), the chip select hardware itself can also compare against address signals A[11:8].*

"Wait a second…," you may say, "if the BASEADDR registers can only be used to specify the most-significant twelve bits of a base address, then what would address signals A[11:8] be compared against?"

Simple. Zeros!

"What?! How could it possibly be meaningful for address signals A[11:8] to only ever be compared against zeros??"

Well, if it is always enforced that a chip select base address be on a 4K-boundary, as required by the relevant documentation, then the value of the least-significant twelve bits of the base address (A[11:0]) should automatically be zero! Indeed, we can further deduce that this must be why the documentation even sets forth such a requirement. (Separately, yet similarly, remember that it is also required that any chip select configured to have a size greater than 4K must have its starting address on a boundary equivalent to its size, e.g., a chip select configured to have a size of 8K must have its base address on an 8K-boundary, a chip select configured to have a size of 16K must have its base address on a 16K-boundary, etc.)

An important consequence of the above is that every sequence of addresses accounted for by some chip select, even those that contain less than 4K addresses (i.e., those corresponding to a size of 256, 512, 1K, or 2K), must start with an address that is on a 4K-boundary. As an example, a chip select can itself account for the first consecutive sequence of 256 addresses starting at address 0x4000, but not the second consecutive sequence of 256 addresses, not the third consecutive sequence of 256 addresses, etc. Following address 0x4000, the first consecutive sequence of 256 addresses that a chip select could account for would start at address 0x5000, the next 4K-boundary. (To account for any intermediate sequences of 256 addresses, or any other sequence of addresses that a chip select could itself not account for, external address decoding is required.)

Overall, to my knowledge, the main reason for performing the above, i.e., the main reason for representing a base address with less than twenty-four bits, is that a lesser amount of flip-flops is needed to implement the BASEADDR registers, which in turn should ultimately lower production costs for the microcontroller. (Savings created by this lesser amount of flip-flops would likely just be on the order of pennies, but even such a small amount of savings could quickly add up over many thousands of production units.) In any event, the particular choice that only twelve bits should be utilized to represent a base address (rather than fourteen, sixteen, etc.) may be, more or less, arbitrary. Hopefully, the designers of the microcontroller came to this decision after some careful analysis, but, unfortunately, that is something that we have not yet been apprised of.

Interrupts

FAQ23. *Is it good practice to use a "jump" instruction inside an interrupt service routine (ISR)?*

ANS. *No. You should aim to make interrupt service routines as short as possible. Doing any shenanigans within an interrupt service routine can lead to really hard-to-detect software bugs.*

In general, try to build the habit of doing a minimal number of things within an interrupt service routine. (Recognize that, whenever there are multiple interrupts enabled, you do not usually wish for one interrupt to prevent the triggering of another.) What is considered "minimal" is application-dependent, but as a personal rule of thumb for our system, spending less than 25-50

clock cycles within an interrupt service routine would be ideal. (Note that this is just a heuristic; there can, of course, be exceptions.)

One common, very useful strategy for minimizing how much is performed within an ISR is to set some memory location with a specific value, so that, once you return from the ISR, some other routine (e.g., a main routine) can be signaled to perform some series of actions. Such a signal provided by a memory location is often denoted by the term "software flag".

For example, suppose that you wish to perform some extensive set of operations at a periodic rate. Rather than do all these operations within an interrupt service routine corresponding to something like an OVF interrupt signal, you could set a memory location with some predefined value (e.g., you could set a single bit within some available register or SRAM memory location), and then within a main routine, you could continually check to see if the relevant memory location contains the relevant value. If the memory location does contain the value, you would perform the desired series of actions; if not, you would just continue to wait and potentially perform other operations in the process.

Overall, remember that the general point of an interrupt is to handle the occurrence of some event as quickly as possible. So, in general, you should first determine whether or not everything related to the interrupt can be performed within the relevant ISR in a reasonable amount of time, like that which is given above. Only if this is not the case should you consider utilizing something such as a software flag.

WaveForms

FAQ24. *The measurements taken by the Scope feature of WaveForms contain a lot of noise. Is there a way to alleviate this?*

ANS. *Make sure that the negative terminal of the relevant channel pin on the Analog Discovery device is connected to a GND pin of the OOTB µPAD.*

COURSE ASSIGNMENTS

Lab 1 – AVR Assembly

FAQ25. *For the skeleton code provided with this lab, why is the '.db NULL' statement listed on a separate line?*

ANS. *Essentially, the '.db NULL' is given on a separate line within the template just to show how the end-of-table (EOT) marker may be referenced within code. Having it on the first line, without other data first, would have probably been confusing to most students. However, you should place the EOT marker in such a way that the '.db' assembler directive(s) incorporate an even number of*

elements, so that "padding" does not occur. (When storing data within program memory, the assembler of Microchip/Atmel Studio enforces that whole words of data, i.e., multiples of two bytes, be stored for every '.db' command; if not, the assembler will pad the given data with an extra byte of zeros, so that an even number of bytes are ultimately stored. For more information, refer to the AVR Assembler User Guide.)

Lab 2 – I/O, Timing

FAQ26. *For the last part of the lab, if the internal SRAM is wholly allocated for animation frames (0x2000 – 0x3FFF), where will stack memory be placed? If I use a '.byte' directive that allocates 0x2000 bytes for the animation, wouldn't this prevent the stack from being used?*

ANS. *Refer to the third footnote on the first page of the relevant lab document.*

Allocating memory with the assembler will not prevent the CPU from utilizing some subset of data memory for the stack. In general, the CPU can manipulate any set of data locations for the stack (or for anything), even when doing so is erroneous.

Overall, we allocate memory with the assembler just so that the programmer can be relatively confident that there is indeed enough memory to support all the data relevant to some application. As you should be able to readily imagine, if we are to omit any knowledge about some relevant data when allocating memory with the assembler, the assembler will not know to consider it. Thus, in the context of our application, the assembler considers allocating 0x2000 bytes of data for the animation to be just fine, since that is the only data segment that we specify to the assembler. However, if we were to specify anything in addition to these 0x2000 bytes (e.g. some segment for stack memory), the assembler should throw an error, since it knows that the relevant data memory of our microcontroller contains only 0x2000 bytes.

Really, if we wish to truly perform safe practices, we should allocate data memory for both the animation table and the stack, but since it is very likely that no one will store close to 8K animation frames for this particular application, there shouldn't be any issues with not doing so. Also, determining exactly how much memory should be utilized for the stack is tedious (and, in general, sometimes nondeterministic), so we usually do not require that you consider this for our course.

Lab 3 – Interrupts

FAQ27. *I can press and release a tactile switch somewhat fiercely and a switch press is only registered once, but if I hold the switch down and sort of roll my finger on it, a switch press gets registered more than once. Is this normal, or am I "debouncing" incorrectly?*

ANS. *Pressing a switch in that form seems to sometimes cause our switches to mechanically open and close multiple times. If you can both press straight down and release straight up pretty roughly without unintended switch presses being registered, your debouncing should be fine.*