ESE381 Embedded Microprocessor Systems Design II

Spring 2024 - K. Short revised February 4, 2024 1:18 pm

Laboratory 03: Clock Control Module CLKCTRL and Software Delays
To be performed the week starting February 11th.

Prerequisite Reading
1. *AVR128DB48 Data Sheet*, Section 12 CLKCTRL - Clock Controller (on Brightspace).
2. *avr-libc 2.0.0* Sections 23.30 and 23.31 (on Brightspace).
3. *Saleae Logic Analyzer User's Manual* (on Brightspace).

Overview
Microcontroller clocking has changed over the years from single fixed frequency crystal oscillator based clocks of the past to multiple clock sources with clock prescaling and sophisticated clock distribution and control networks. The AVR128DB48 has powerful clocking capabilities controlled by its CLKCTRL module. During this laboratory, you will become familiar with the operation of the CLKCTRL module. You will also have your first experience with configuration change protection (CCP) registers used to protect critical special function registers (SFRs) in the microcontroller from inadvertent change.

You will also get your first chance to use the Saleae Logic Analyzer to measure digital and analog waveforms. For the tasks you are doing in this particular laboratory, the Saleae Logic Analyzer's performance cannot compare with that of the oscilloscope that you have used in the past. You will also use the oscilloscope to make the same measurements you make with the Saleae. However, when we start looking at serial protocols in the next laboratory you will see that the Saleae has "protocol aware" features that are extremely useful when analyzing serial data.

You can download the Saleae software and run it in simulator mode at home to learn how the software operates, you can accomplish this without having the Saleae hardware.

https://www.saleae.com/downloads/

If you finish the required laboratory tasks early, you should spend any extra time you have becoming more familiar with the Saleae's operations and capabilities while you have the actual hardware. You will be required to use the Saleae Logic Analyzer extensively throughout the semester

For all the programs that you write for this laboratory use the predefined macro names from avr128db48.h for bit masks, bit fields and group configurations.

Design Tasks

Design Task 1: Generating a Clock Frequency at CLKOUT
Write a program named `one_MHz` that generates a 1 MHz CLK_PER using the internal high frequency 4 MHz oscillator and brings CLK_PER out at the CLKOUT pin. The pin that is CLKOUT, based on the particular IC package, can be found by looking in Section 3. I/O Multiplexing and Considerations in the Special column. For the AVR128DB48 this is pin 3 of the IC, or PA7.

Submit your program as part of your prelab.

Design Task 2: Blink LED0 Using an avr-libc Delay Function
Write a program named `blink_LED0_1sec` that blinks LED0 at a rate of 1Hz or period of 1 second. Use one of the avr-libc delay functions to accomplish this. LED0 is driven by pin PB3. Use the oscilloscope to measure the period of the signal at PB3 to determine how accurate is your 1Hz output.

Submit your program as part of your prelab.

Design Task 3: Determining Whether Changing CLK_OUT (or CLK_PER) Using The Main Clock Prescaler Also Changes CLK_CPU
Looking at the CLKCTRL block diagram, fig. 12.2.1 of the data sheet, the input to the Main Clock Prescaler is CLK_MAIN, the selected clock source frequency. There are two branches out of the Main Clock Prescaler, one of them is labeled CLK_CPU and the other is labeled CLK_PER, which is also CLKOUT. Since there are two branches shown out of the Main Clock Prescaler, you would expect that the two branches can be controlled independently by the Main Clock Prescaler. This would be advantageous if it allowed CLK_CPU to not be prescaled, but CLK_PER to be prescaled. This would allow one to run the CLK_CPU at the CLK_MAIN frequency and run CLK_PER at a slower clock speed for the peripherals.

In my reading of the data sheet I could not find a definitive statement about the relationship between CLK_CPU and CLK_PER.

Ascertain the relationship between CLK_CPU and CLK_PER empirically. Write a program named `CLK_CPU_vs_CLK_PER` that you can use to determine, in the laboratory, whether these two clocks frequencies can be controlled independently. Also write a brief explanation, your verification strategy, of how you will use your program and the laboratory equipment to define the relationship.

Submit your program listing, and verification strategy as part of your prelab.

Design Task 4: Toggling a Bit Every xxx us Using a Library Function
You need to toggle a bit at a port pin at one of three different rates: every 52.08333, 104.1666, or 208.333 us. Write a program `toggle_every_xxx_us` that accomplishes this using one of the

delay functions provided in the avr-libc library. CLK_MAIN for the AVR128DB48 must be set for 4 MHz. You can toggle any pin you like. Use two bits of the DIP switch to select the toggle rate.

Submit your program listing, and program verification strategy as part of your prelab.

Design Task 5: 32.768 kHz CLK_MAIN

The AVR128DB has an internal 32.768 kHz oscillator that can be used to provide CLK_MAIN. This oscillator is optimized for ultra low-power (ULP) operation, but has decreased accuracy compared to using an external 32.768 kHz crystal.

You must write a program named `clk_main_32768Hz` that makes CLK_MAIN equal to the output from the 32.768 kHz internal oscillator. Your program must also bring this signal out to pin CLKOUT.

Submit your program listing, and program verification strategy as part of your prelab.

Design Task 5 Extra Credit (+5 points)

Modify your program `clk_main_32768Hz` to create the program `clk_main_ext_32768Hz` that uses the external 32.768 kHz crystal oscillator to provide CLKMAIN and brings it out to pin CLKOUT.

To do this there is an extra step after you select the external 32.768 kHz oscillator as the source. External oscillators must be enabled. Since the external and internal 32.768 kHz frequencies are so close, just measuring the output frequency is not enough. To get the extra credit in Laboratory Task 5, you must also be able to show the TA how you can be certain that you are actually getting the output from the external 32.768 kHz crystal oscillator.

Laboratory Activities

Laboratory Task 1: Generating a Clock Frequency at CLKOUT

Create a project named `clk_per_to_clkout`. Type in either of the versions of the programs in Lecture 6 that map CLK_PER to CLKOUT. Compile the program with the optimization level at -Og and run the program. Measure the signal at clock out and verify that it is 4 MHz using the oscilloscope.

Recompile this program with the optimization level set to -O0. Run the program. What is the frequency of the waveform at CLKOUT?

Create a project named `one_MHz`. Load and debug your program named `one_MHz`. Verify that your program generates a 1 MHz square wave at the CLKOUT pin.

Have a TA verify that your 1 MHz signal is correct and get that TA's signature.

Laboratory Task 2: Blink LED0 Using an avr-libc Delay Function
Create a project named `blink_LED0_1sec`. Compile your `blink_LED0_1sec` program. Using the oscilloscope, measure the signal at CLKOUT (PA7) and verify that it is 4 MHz.

Verify that your program generates a 1 Hz square wave at pin PB3.

Have a TA verify that your 1 Hz signal is correct and get that TA's signature.

Laboratory Task 3: Determining Whether Changing CLK_OUT (or CLK_PER) Using The Main Clock Prescaler Also Changes CLK_CPU
Create a project named `CLK_CPU_vs_CLK_PER`. Add your `CLK_CPU_vs_CLK_PER` program and compile it. Load, debug, and run your program. Debug your system if it does not operate properly.

Have a TA verify that you have compiled and run this program and get that TA's signature.

Laboratory Task 4: Toggling a Bit Every xxx us Using a Library Function
Create a project named `toggle_every_xxx_us`. Add your program `toggle_every_xxx_us` and compile it. Load, debug, and run your program. Measure the width of the output pulses at each of the different rates using your oscilloscope. Take measurements for each rate and place the measurement results in a table. Compute the percent error in your output pulse width at each rate and also place that information in your table. Finally, determine the error in us for the pulse width at each rate and also place that information in your table.

Have a TA verify whether your three pulse widths are correct and get that TA's signature.

For any one of the toggle rates capture the CLKOUT waveform using the Saleae Logic Analyzer. Use Saleae's measurement capability to add the pulse width measurement information to your captured waveform.

Have a TA verify whether you were successful in capturing the CLKOUT waveforms with the Saleae Logic Analyzer and get that TA's signature.

Laboratory Task 5: 32.768 kHz CLK_MAIN
Create a project named `clk_main_32768Hz`. Add your program `clk_main_32768Hz` and compile it. Load, debug, and run your program. Measure the frequency of the signal at CLKOUT. Compute the percent error in your output compared to the specified frequency.

Have a TA verify that your output frequency is correct and get that TA's signature.

Laboratory Task 5 Extra Credit (+5 points)
Create a project named `clk_main_ext_32768Hz`. Add your program `clk_main_ext_32768Hz` and compile it. Load, debug, and run your program. Measure the frequency of the signal at CLKOUT. Compute the percent error in your output compared to the specified frequency.

To get the extra credit in Laboratory Task 5, you must also be able to show the TA how you can be certain that you are actually getting the output from the external 32.768 kHz oscillator.

Have a TA verify that your output frequency is correct and explain how you know without a doubt that the clock source is the 32.768 kHz external oscillator. Get that TA's signature.

Questions
1. Explain why you got the waveform you did at CLKOUT when you compiled and ran the program `clk_per_to_clkout` with the optimization level at -O0.

2. Explain your strategy for determining whether changing CLK_OUT (or CLK_PER) using the Main Clock Prescaler also changes CLK_CPU. What was your determination as to the independence or dependence of these two clocks?

3. Can you tell from the data you took in Task 3 whether either the percent error or the absolute error stay the same as you change the argument of the delay function for the different toggle times. Please explain your answer.

4. In Design Task 1 you had to configure the CLKCTRL module to generate a 1 MHz output at CLKOUT. Briefly describe two different ways that you could do this.

5. Briefly explain how you could, in software, improve the accuracy of the output in Laboratory Task 4. Would this solution work in production when you are going to manufacture 1,000,000 copies of the system? Explain.