**ESE 381 Embedded Microprocessor Systems Design II**

Spring 2016, K. Short
February 21, 2016 9:24 pm

**MODULE # 2a Timer/Counter Basics Using Timer/Counter0**
To be performed the weeks starting Feb. 21st.

**Prerequisite Reading**

1. Lectures 8 and 9 Timer/Counters
2. AVR130: Setup and Use of the AVR Timers (on Blackboard).
3. ATmega128 Data Sheet pages 194 - 216 (on Blackboard).
4. Agilent InfiniiVision 3000 X-Series Oscilloscopes User's Guide (on P Drive)

**Overview**

The ATmega128 has four on-chip programmable timer/counters. Two of these counters are 8-bit timer/counters and two are 16-bit timer/counters. Timer/Counter0 is an 8-bit timer/counter that can be operated using the internal ATmega128 system clock (synchronous operation) or operated using an external clock (asynchronous operation) provided by attaching a crystal to pins TOSC1 and TOSC2. Of the four timer/counters, Timer/Counter0 is the only one that can be operated asynchronously. This laboratory will use Timer/Counter0 in synchronous operation.

The primary objective of this laboratory is to introduce you to the basic operation of timer/counters in AVR microcomputers and use of the oscilloscope to capture single events.

There are four modes of operation of Timer/Counter0:

1. Normal mode
2. Clear Timer on Compare Match mode
3. Fast Pulse Width Modulated (PWM) mode
4. Phase Correct PWM mode

The mode is determined by the values of WGM01:0 in TCCR0. An output waveform can be generated at pin OC0. Two interrupts can be generated. One when the counter TCNT0 overflows setting the flag TOV0. The other interrupt occurs when the count in counter TCNT0 matches the compare value in register OCR0 setting the OCF0 flag.

*Normal Mode*

The simplest mode of operation is the normal mode (WGM01:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter0 Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero.

*Clear Timer on Compare Match (CTC) Mode*

In Clear Timer on Compare or CTC mode (WGM01:0 = 2), the OCR0 Register is used to manipulate the counter resolution. In CTC mode, the counter is cleared to zero when the counter value (TCNT0) matches the OCR0. The OCR0 defines the top value for the counter, hence also its reso-

lution. This mode allows greater control of the compare match output frequency.

*Fast PWM Mode*
The fast Pulse Width Modulation or fast PWM mode (WGM01:0 = 3) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to MAX then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0) is cleared on the compare match between TCNT0 and OCR0, and set at BOTTOM.

*Phase Correct PWM Mode*
The phase correct PWM mode (WGM01:0 = 1) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC0) is cleared on the compare match between TCNT0 and OCR0 while up counting, and set on the compare match while down counting.

Using these various modes of operation, the counter/timers allow you to generate hardware delays, single pulses, periodic waveforms, and pulse width modulated waveforms. You can also measure pulse duration, waveform period, frequency, and count external events.

**Design Tasks**

When directly generating an output signal using Timer/Counter0, the output appears at pin OC0. This is the alternate function for general I/O pin PB4. Since we used this pin to provide the RS signal to the LCD, you will have to disconnect the flat cable to the LCD at the header on the breadboard when running your program in the laboratory.

Simulation of your programs will be limited by the fact that counter TCNT0 does not actually count when doing a simulation. When doing an in-circuit emulation in the laboratory, the counter will operate fully. However, if you are single stepping your program you must disable the Run timers in stopped mode option or your counter will continue to run between your single stepping of instructions.

*Design Task 1: Timer/Counter0 Generated Delay - Normal Mode*
Write a test program named `hardware_delay_test` that generates a hardware delay using Timer/Counter0 in normal mode. Use TOV0 to generate an interrupt at the end of the delay. Make your delay 1.0 ms long and as precise as possible. See the example test programs in Lecture 8. To be able to observe the delay in the laboratory, toggle a previously unused port bit when you start the delay and have the interrupt service routine toggle the same port bit at the end of the delay.

Using Timer/Counter0 in normal mode, what is the longest delay you can directly generate and what is its resolution in microseconds? Using Timer/Counter0 in normal mode, and producing the highest resolution delay, what is the longest delay you can directly generate and what is its resolu-

tion in microseconds?

***Your source file listing is required as part of your prelab.***

*Design Task 2: Timer/Counter0 Generated Pulse - Normal Mode*
Write a test program named `hardware_pulse_test`. This program must generate a pulse at OC0 pin that is 400us in duration using Timer/Counter0. Timer/Counter0 must be operated in its normal mode. Write your code as a loop so that the pulse is generated repeatedly. Use the intrinsic software delay function to provide a delay of about 1000 us between pulses.

***Your source file listing is required as part of your prelab.***

*Design Task 3: Timer/Counter0 Generated Square Wave - Clear Timer on Compare (CTC) Mode*
Write a test program named `hardware_freq_test`. This program must generate a square wave with a frequency of 2 kHz with as high a precision as possible. The square wave must appear at pin OC0.

*Design Task 4: Timer/Counter0 Generated Periodic Wave - Fast PWM Mode*
Write a test program named `hardware_pwm_test`. This program must use the Fast PWM mode to generate a periodic waveform with a 22% duty cycle and a frequency of 500Hz. The pulse width modulated wave must appear at pin OC0.

**Laboratory Activity**
Pin OC0 is the alternate function for general I/O pin PB4. PB4 has been previously used to provide the RS signal to the LCD. You will have to disconnect the flat cable to the LCD, at the header on the breadboard, when running your programs for this laboratory. Do not remove any of the wiring to the LCD display header, as the LCD will be used again later.

*Laboratory Task 1: Timer/Counter0 Generated Delay - Normal Mode*
Load and debug your program `hardware_delay_test`. Configure the MSO-X 3012A oscilloscope to capture single events. If you do not know how to do this, refer to the User's Manual. Using the oscilloscope, demonstrate to a TA that your delay is the correct length.

**Have a TA verify and sign whether the waveform at the pin being toggled meets the design specification.**

*Laboratory Task 2: Timer/Counter0 Generated Pulse - Normal Mode*
Load and debug your program `hardware_pulse_test`. Use a breakpoint in your program to generate a single pulse each time the program is run to the breakpoint. Configure the MSO-X 3012A oscilloscope to capture single events. Using the oscilloscope, demonstrate to a TA that your delay is the correct length.

Reconfigure the oscilloscope's trigger mode to Auto. Run your program and verify the duration of the pulse being generated.

**Have a TA verify and sign whether the waveform at OC0 meets the design specification.**

*Laboratory Task 3: Timer/Counter0 Generated Square Wave - Clear Timer on Compare (CTC) Mode*
Load and debug your program `hardware_freq_test`. Verify that the square wave at pin OC0 has the correct frequency.

**Have a TA verify and sign whether the waveform at OC0 meets the design specification.**

*Laboratory Task 4: Timer/Counter0 Generated Periodic Wave - Fast PWM Mode*
Load and debug your program `hardware_pwm_test`. Verify that the duty cycle and frequency of the pulse width modulated wave at pin OC0 is correct.

**Have a TA verify and sign whether the waveform at OC0 meets the design specification.**

**Leave the hardware that you have wired on the breadboard intact. This hardware will be used again in later laboratories.**