**ESE 380 Embedded Microprocessor Systems Design I**

Fall 2018 - K. Short
November 26, 2018 2:21 pm

## LABORATORY 10: Frequency Meter System with Auto Ranging

To be performed during your assigned laboratory section the week beginning December 2nd.

### Prerequisite Reading

1. *AVR130: Setup and Use the AVR Timers*, Atmel application note (on Blackboard).
2. ATmega324A data sheet pages 129 through 189 (on Blackboard).
3. *AVR200: Multiply and Divide Routines* (on Blackboard in Atmel Documents Folder).
4. Prof. Short's 32 x 32 Unsigned Integer Division Subroutine (on Blackboard).
5. *AVR204: BCD Arithmetics* (on Blackboard in Atmel Documents Folder).

### Overview

As you know, the ATmega324A contains three independent hardware timer/counters. Timer/Counter0 and Timer/Counter2 are 8-bit timer/counters. Timer/Counter1 is a 16-bit timer/counter. The timer/counters on the ATmega324A are very flexible and can be used to do many different timing and counting functions.

As you saw in Laboratory 9, one of the many things a timer/counter can do is measure time intervals. It can also count external events. The advantage in using a hardware timer/counter over software for these kinds of measurements is that it frees the CPU to perform other tasks simultaneously. In addition, time intervals and frequency can be measured with greater precision using timer/counter hardware.

A timer/counter can be used to directly measure frequency or period. To measure period, the timer/counter is started on a positive edge of the signal whose period is being measured and stopped on the following positive edge. During that time interval the timer/counter counts a prescaled version of the system clock. After the sample period is over, the count is a measure of the signal's period in units of the timer/counter clock period.

To measure frequency, the timer/counter is run for a fixed time interval, called the sample interval. During this interval of time, the timer/counter counts the positive edges of the signal whose frequency it is measuring. If the sample interval is one second, the count at the end of the sample interval is the signal's actual frequency in Hertz. If the sample period is some other interval of time, the frequency in Hz can be computed based on the sample interval and count.

An important consideration in measuring frequency or period is the precision of the measurement and how long the measurement takes to obtain the desired precision. For example, to measure,

with high precision, the frequency of a relatively low frequency signal, the period is measured directly, and the frequency is computed from the period. In contrast, to measure, with high precision, the frequency of a relatively high frequency signal, the frequency is measured directly.

One way to measure period is to connect the signal whose period is being measured to an external interrupt pin, whose sense is configured for positive edge. In response to an interrupt, the ISR clears and then starts the timer/counter. Then the ISR returns. In response to the next interrupt, the ISR stops the timer/counter and reads the count. A memory flag must be used by the timer/counter's ISR to keep track of whether it should be clearing and starting the counter or stopping and reading the counter. Another memory flag must be used by the ISR to signal to the background task when a measurement has been completed. The background can monitor this flag to determine when it should display a new measurement of the period and compute and display the frequency from the measured period.

Another way to measure a signal's period is to use Timer/Counter1's input capture capability. The signal whose period is to be measured is connected to the Input Capture Pin (ICP1). When configured for input capture, The Input Capture Register (ICR1 or ICR1H:ICR1L) captures Timer/Counter1's value at a given external (edge triggered) event on ICP1, if the input capture interrupt is enabled (TICIE1), an interrupt occurs. The ISR has to compute, from the last two values captured, the period. The background software can then display the period and compute and display the frequency.

Computation of frequency from period requires computation of a reciprocal. However, at this point we are constrained to using integer arithmetic. Thus, you need to scale the 1 in the numerator to a larger number, so that after the division the fractional value is not completely lost. When displaying the result, a decimal point can be appropriately placed to unscale the value.

A frequency measurement system can be designed to automatically determine whether to make a measurement as a direct period or a direct frequency measurement based on the period or frequency of the signal being measured.

## Design Tasks

*Design Task 1: Measuring Period and Displaying Period and Frequency*
Write a program named `direct_period_meas` that uses Timer/Counter1 to directly measure the period of a relatively slow periodic waveform. The waveform is connected to INT1. Use the first of the two period measurement techniques described in the Overview. That is, do not use input capture.

The period of the signal ranges from 1000 microsecond to 50,000 microseconds. This corresponds to a frequency from 1 kHz down to 20Hz. Configure Timer/Counter1's prescalar for 1.

Your program must display the frequency and period determined from the measurement on the LCD. Use the following format on the LCD for displaying frequency and period:

```
Frq =   1000 Hz
Prd =   1000 us
```

Compute and display the frequency in units of 1HZ. Display the period in units of 1 us.

**Submit your program listing as part of your prelab. Make sure that your program includes a program header and each subroutine includes a subroutine header and clear comments throughout.**

*Design Task 2: Generating a One Second Gate.*
Direct measurement of frequency requires a precise and accurate gate interval or sample interval during which cycles of the waveform being measured are counted. The longer this sample period the more precise the measurement, but the longer it takes to make the measurement. We would like to have an exactly 1 second sample interval. However, this is difficult because our precscalar values are powers of 2 and not powers of 10. We will try to get a sample period as close to 1 second as possible using only Timer/counter0 and possibly an external flip-flop.

Determine how to configure Timer/counter0 and possibly an external flip-flop to create a sample gate as close to 1 second as possible. The transitions of the sample gate are used to determine when to start and stop Timer/counter1 to count cycles of the waveform being measured. If your approach to generating a 1 second gate requires any hardware external to the ATmega324A, draw a schematic showing that hardware.

Write a program named `sample_interval` that uses Timer/Counter0 to generate a sample period of as close to 1 second as possible.

**Submit your schematic and program listing as part of your prelab. Make sure that your program includes a program header and each subroutine includes a subroutine header and clear comments throughout.**

*Design Task 3: Measuring Frequency and Displaying Period and Frequency.*
Write a program named `direct_freq_meas` that uses Timer/Counter1 to directly measure the frequency of a relatively fast periodic waveform. Use the gate signal from Task 2 to specify the sample interval during which cycles from the waveform being measured are counted by Timer/counter1.

The frequency of the signal being measured is from 1 kHz to 65kHz. Configure Timer/Counter1's

prescalar for 1. Your program must display the frequency and period determined from the measurement on the LCD. Use the following format on the LCD for displaying frequency and period:

```
Frq = 65000 Hz
Prd =    15 us
```

Compute and display the frequency in units of 1 HZ. Display the period in units of 1 us.

**Submit your program listing as part of your prelab. Make sure that your program includes a program header and each subroutine includes a subroutine header and clear comments throughout.**

*Design Task 4: Autoranging - Measuring Either Period or Frequency, Whichever is More Precise, and Displaying Period and Frequency.*
Design a system that measures and displays the frequency and period of signals with frequencies from 20Hz to 65kHz. When the frequency is less than or equal to 1 kHz, the system directly measures period and computes frequency. When the frequency is greater than 1 kHz, the system directly measures frequency and computes period. The display of the value directly measured should have an asterisk suffix to indicate it is being directly measured.

```
Frq = 65000 kHz*
Prd =    15 us
```

Write a program named `auto_range_meas` that determines which range of frequencies the input signal is in and uses the appropriate method to measure the signal.

**Submit your program listing as part of your prelab. Make sure that your program includes a program header and each subroutine includes a subroutine header and clear comments throughout.**

******* The following tasks are optional extra credit tasks. *******

*Extra Design Task 1: Using Input Capture to Measure Period*
Repeat Task 1, except using input capture to measure the period. Name the program `input_capture_meas`.

**Submit your program listing at the end of the laboratory for this extra credit task.**

*Extra Design Task 2: Adding Measurement of Duty Cycle to Period and Frequency*
Write a program named `duty_cycle_meas` that adds the measurement of duty cycle to that of

period and frequency from Task 4.

Use the third line of the LCD to display the duty cycle:

```
Frq =  1000 Hz*
Prd =  1000 us
Dty =  50.1 %
```

**Submit your program listing at the end of the laboratory for this extra credit task.**

**Laboratory Activity**

*Laboratory Task 1: Measuring Period and Displaying Period and Frequency*
Load and debug your `direct_period_meas` program.

Using the function generator, generate a periodic waveform with a duty cycle of 50%. Run your program and verify the values on the LCD display for periods between 1 millisecond and 50 miliseconds... Make observations in steps of 10 ms. For each observation, verify that the LCD displays the correct information.

**When your program runs correctly, get a TA to verify its operation and obtain the TA's sign off.**

*Laboratory Task 2: Generating a One Second Gate.*
Load and debug your `sample_interval` program. Using the oscilloscope, measure the duration of your sample interval signal.

**When your program runs correctly, get a TA to verify its operation and obtain the TA's sign off.**

*Laboratory Task 3: Measuring Frequency and Displaying Period and Frequency.*
Load and debug your `direct_freq_meas` program.

Using the function generator, generate a periodic waveform with a duty cycle of 50%. Run your program and verify the LCD display for frequencies between 1 kHz and 65 kHz. Make observations in steps of 10 kHz. For each observation, verify that the LCD displays the correct information.

**When your program runs correctly, get a TA to verify its operation and obtain the TA's sign off.**

*Laboratory Task 4: Autoranging - Measuring Either Period or Frequency, Whichever is More Precise, and Displaying Period and Frequency.*
Load and debug your `auto_range_meas` program.

Using the function generator, generate a periodic waveform with a duty cycle of 50%. Run your program and verify the LCD display for frequencies between 20 Hz and 65 kHz. Make observations from 20 Hz to 1 kHz in steps of 100 Hz and from 1 kHz to 65 KHz in steps of 10 kHz. For each observation, verify that the LCD displays the correct information.

**When your program runs correctly, get a TA to verify its operation and obtain the TA's sign off.**

*Extra Laboratory Task 1: Using Input Capture to Measure Period*
Load and debug your `input_capture_meas` program.

Using the function generator, generate a periodic waveform with a duty cycle of 50%. Run your program and verify the LCD display for periods between 1 millisecond and 50 miliseconds... Make observations in steps of 10 ms. For each observation, verify that the LCD displays the correct information.

**When your program runs correctly, get a TA to verify its operation and obtain the TA's sign off.**

*Extra Laboratory Task 2: Adding Measurement of Duty Cycle to Period and Frequency*
Load and debug your `duty_cycle_meas` program.

Using the function generator, generate a periodic waveform with a duty cycle of 10%. Run your program and verify the LCD displays the correct duty cycle for duty cycles between 5 and 95%. Make observations in steps of 10%. For each observation, verify that the LCD displays the correct information.

**When your program runs correctly, get a TA to verify its operation and obtain the TA's sign off.**