CSE 466 Exam I 11 February 2008

1. General concepts (40 points)

Define the following terms related to features of your Atmega16 microcontroller. Provide a sentence or two explaining the concept, which unit in your microcontroller embodies this feature, and an example of when the feature was used in your lab assignments. If you didn't use it, explain why it was not needed.

a) Input capture

Input capture is used to "capture" the time at which an event occurs on a GPIO pin. When the pin changes value, the current state of a counter is recorded in the input capture register and an interrupt is generated. This was used in the lab assignments to measure the timing of pulse width modulated signals coming from the accelerometer.

b) Pulse-width modulation

Pulse-width modulation enables us to communicate a more or less continuous value by varying the width of a pulse on a GPIO pin. The average value is the duty-cycle of the pulse. Pulse-width modulation was used by the accelerometer to communicate its values of acceleration in two dimensions using only two pins. It was also used to vary the intensity of a tri-color LED.

c) Interrupt enable flag

Every interrupt source in our microcontroller has a corresponding interrupt enable flag. By enabling this flag, we tell the microcontroller we will field the interrupt with an appropriate interrupt service routine. By disabling the flag, we ignore the interrupts caused by that source. This was used throughout the lab assignments.

d) Serial-peripheral interface (SPI)

The SPI serial communication interface was used to communicate between the microcontroller and the FTDI USB interface chip. It is a 4-wire interface (two signals, clock, and source select) used to exchange data between the devices – usually, a microcontroller and a peripheral chip.

2. A-to-D Conversion and Interrupts (30 points)

a) What is free-running mode on the analog-to-digital converter? When might you want to use it? When might you not want to use it? Cite specific examples from the labs (this should require on the order of 1-2 paragraphs to explain).

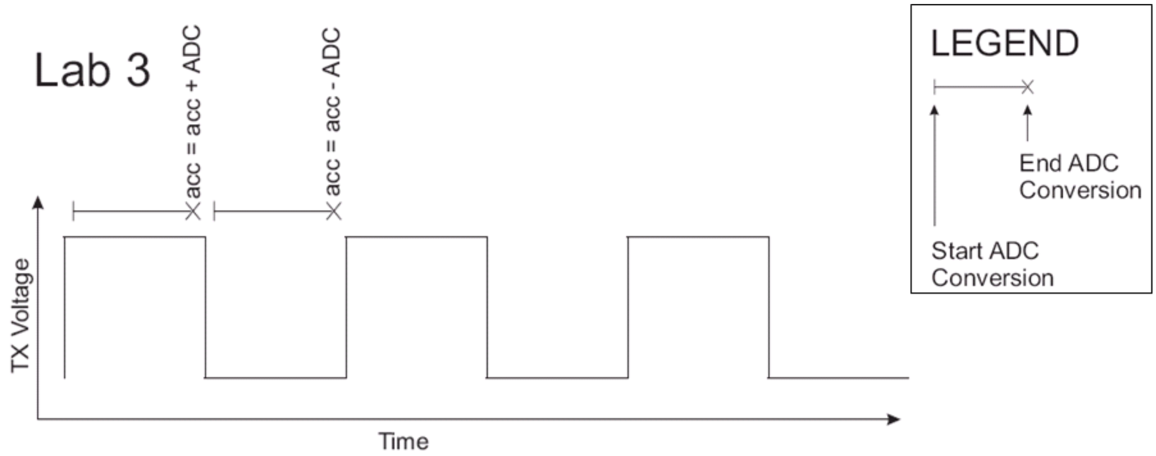
Free-running mode on the ADC enables conversions continuously and as fast as they are completed. As soon as one conversion completes, an interrupt is signaled, and a new conversion begins. This is useful when we want to continuously monitor an analog input. However, it may generate too many interrupts in some situations – monopolizing the microcontrollers' time servicing the ADC and not allowing enough cycles for other functions. In the lab, free-running mode was used when you first connected the potentiometer to the microcontroller but then abandoned when you also had to start generating pulse-width modulated signals for the LED. At that point, you used an explicit trigger for the ADC conversions rather than continuously converting the signal. Also, when building your electric field sensor, ADC had to be performed at a specific time linked to how you were driving your transmit electrode. Finally, in free-running mode you can't switch between different analog inputs easily.

b) Why is it important for interrupt service routines to be short? What problems might occur if they are not?

Interrupt service routines should be short because we often must disable an interrupt flag during their operations. This means we may miss another interrupt or fall behind servicing other interrupts. By keeping the ISR as short as possible we minimize the potential for these sorts of problems. However, we should keep track of how many cycles interrupts will require and their rate of occurrence so that we do not run out of available microcontroller cycles.

3. Data Communication (30 points)

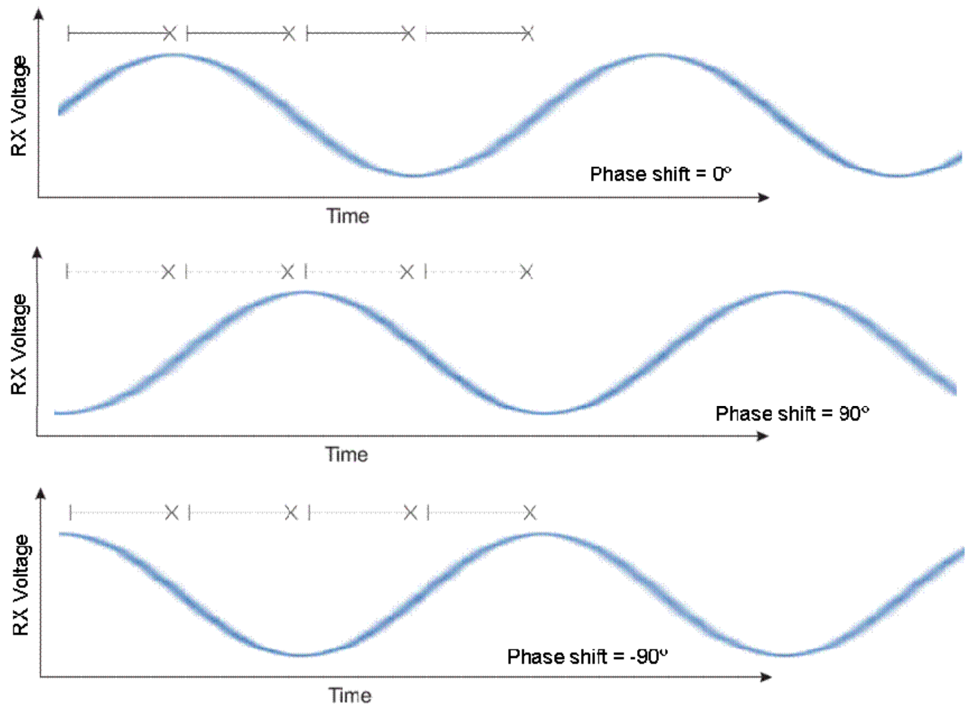
In lab 3, you generated a square wave, and demodulated on the same phase and frequency as the transmitted square wave.



The phase offset between the transmitted signal and the received signal (demodulation channel) is 0 degrees.

It is also possible to demodulate with a different phase offset between transmit and receive. And, it is possible to take the same received signal and demodulate it in more than one way. By measuring on the same phase (0 degrees) and also on a different phase (90 degrees off), it is possible to measure phase shifts: for example, a 45 degree phase shift will make equal contributions to the 0 degree and the 90 degree channel. This can allow you determine whether the impedance between the transmit and receive signal electrodes is capacitive, inductive, resistive, or a combination of these. (If there is no phase shift between transmit and receive signals, this corresponds to a real impedance, i.e. a pure resistance. If the received current is phase shifted by +90 degrees, it corresponds to a pure inductance [positive imaginary impedance]; if the received current is phase shifted by -90 degrees, it indicates a pure capacitance [negative imaginary impedance.]) The simultaneous use of the 0 degree and the 90 degree channel (called “in-phase” and “quadrature” channels) is also used to increase the throughput of communication systems.

The figure below shows three versions of the received signal – shown as a sinusoid to approximate the received waveform (the sinusoid shown has the same frequency as the square wave) – the top one has 0 degree phase shift, the second has +90 degrees shift, and the third has -90 (or 270) degrees phase shift.



Write pseudo-code to generate a TX square wave (at half the transmit frequency of lab 3) and demodulate it using two separate accumulators, one for a 0 degree phase shift (the "inphase" accumulator), and one for a 90 degree phase shift (the "quadrature" accumulator). Write this code on the **NEXT PAGE**.

The result of your pseudo-code should be accumulated samples in these two accumulators.

Recall that a resistance is called a real impedance, and causes no phase shift between the transmitted voltage and received current. An inductance is called a positive imaginary impedance (+90 degree phase shift). A capacitance is called a negative imaginary impedance (-90 degree phase shift). For example, let's agree that when the impedance between the TX and RX is purely capacitive, the in-phase signal will be approximately zero, and the quadrature signal will be large and negative.

At the end of your sampling loop, *in terms of the two accumulator variables*, specify how you would determine (you can use terms such as "approximately 0" and "large and positive" as in the example above) that the impedance is:

a) mostly resistive – *inphase is large and positive and quad is approximately 0*

b) mostly inductive – *inphase is approximately 0 and quad is large and positive*

These can be obtained by sampling the waveforms above for each of the three cases of +90, 0, and -90 degrees phase shift.

FOR YOUR PSEUDO-CODE: Do not be concerned with syntax. In fact, you can use English (e.g., "read timer0 value). Assume that you have a "readADC" function that returns the current value on the receive electrode and that the ADC is already configured appropriately. The complete answer should easily fit on this single page.

We can generate the transmit waveform and sample the receive electrode as follows:

```
for (i = 0; i<200; i++) {

SET PORTB HIGH

Sample0 = READADC

SET PORTB HIGH  // This is optional but makes the timing more uniform
                   and the code more symmetric

Sample90 = READADC

SET PORTB LOW

Sample180 = READADC

SET PORTB LOW  // This is optional but makes the timing more uniform
                  and the code more symmetric

Sample270 = READADC

inphase = inphase + Sample0 - Sample180

quad = quad + Sample90 - Sample270

}
```

This is basically the same code as you used in lab 3 except that we know sample the high portion and low portion twice each instead of only once. This generates a waveform that is half the frequency as we now take twice as long to change from high to low and low to high. `inphase` *is the same as our accumulator variable and we now add the similar* `quad` *variable for the additional two samples we are collecting that are 90 degrees out of phase.*