Embedded Real-Time Systems (AME 3623) Homework 4 Solutions

April 30, 2009

Question 1

1. (10pts) Briefly explain why *polling* can be undesirable when performing input/output operations.

   Polling is undesirable because the processor can potentially spend a long time waiting for something to happen (we refer to this as busy waiting*). This time could otherwise be used to do other things.*

2. (10pts) Briefly outline what the microprocessor does in response to an interrupt.

   The microprocessor stops what it is currently executing, remembers the location of the next instruction, executes the associated interrupt service routine, and then returns to the next instruction in the main program.

Question 2

Suppose we want to produce a regular interrupt frequency of approximately 30.49 Hz. Assume that we are using a 16 MHz crystal for our clock.

1. (5 pts) Which timer should we use?

   Timer 1.

2. (5 pts) Which prescaler should we use?

   Prescaler: 8

Question 3

Suppose we want to produce a regular interrupt every 512 μs. Assume that we are using a 16 MHz crystal for our clock.

1. (5 pts) Which timer should we use?

   Timer 2.

2. (5 pts) Which prescaler should we use?

   Prescaler: 32

Question 4

1. (15pts) Suppose we want a function – called $donow()$ – to be executed once every $0.79s$. Assume a system clock of $16MHz$. What is the timer1 prescaler configuration and the (pseudo)code for the interrupt routine (the code does not need to be syntactically correct)? Also - show the code in your main function that configures the timer.

   We will use a prescaler of 64. This gets us down to an interrupt every 0.2621 s. We then need an interrupt routine with an additional counter that expires at 3. So, we are left with an interrupt interval of: $3 * 64 * 256 * 256/16000000 = 0.7864s$.

```
ISR(TIMER1_OVF_vect) {
    static uint8_t counter = 0;

    ++counter;
    if(counter == 3) {
     donow();
     counter = 0;
   };
};
```

```
Somewhere in the main program:


// Interrupt occurs every
//     (64*256*256)/16000000 = 0.2621 sec
timer1_config(TIMER1_PRE_64);
// Enable the timer interrupt
timer1_enable();
// Enable global interrupts
sei();
```

Question 5

Consider the following code.

```
volatile uint8_t duration;

ISR(TIMER0_OVF_vect) {
   static uint8_t counter = 0;

   ++counter;
   if(counter == 0) {
    donow1();
   }
   if(duration == counter) {
    donow2();
  };
};

Somewhere in the main program:

// Interrupt occurs every
//     (256*256)/16000000 = 4.096 ms
timer0_config(TIMER0_PRE_256);
// Enable the timer interrupt
timer0_enable();
// Enable global interrupts
sei();

while(1)
{
   <change the value of duration>
}
```

1. (5 pts) What does the ISR do?

   Assuming that the global variable duration is not changing: the ISR calls donow1() and donow2() at regular intervals (both are called once every $(256 * 256 * 256)/16000000 = 1.0486s$*). The second function is called* $duration * 4.096ms$ *after the first.*

2. (5 pts) What does the main program do (in the while() loop)?

   It is responsible for setting the timing difference between calls to donow1() and donow2().