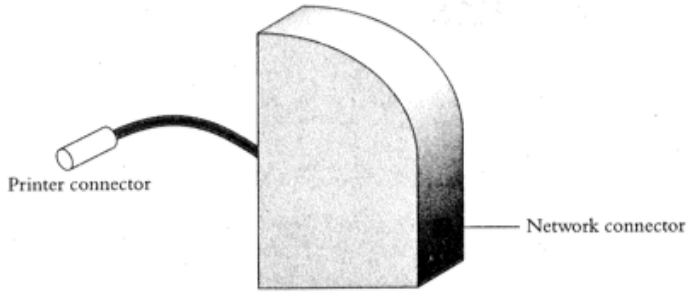
CPE355 - Real time embedded kernels - Spring'11
Prof. Nuno Alves (nalves@wne.edu), College of Engineering
Homework Assignment #7
Due date: 11am, Tuesday, February 14th 2012

WESTERN NEW ENGLAND UNIVERSITY | WNE

Homework turned in after the deadline, will not be graded.

Question #1 (5 points) - Consider a system that controls the traffic lights at a major intersection. It reads from sensors that notice the presence of cars and pedestrians, it has a timer, and it turns the lights red and green appropriately. What architecture might you use for such a system? Why? What other information, if any, might influence your decision?

Question #2 (5 points) - Consider the following telegraph:



The telegraph allows you to connect a printer and also has a high-speed connection to a network. Data comes in the network and that data needs to be printed. However there are several constraints:

1) The arriving data sometimes arrives out of order, sometimes gets lost, and some other times it may arrive late. The telegraph must clean this chaos and send out a clean data stream to the printer.

2) The printer may be connected to a lot of other telegraphs and only one job can be printed at the same time.

3) The printer must provide status to any telegraph that requests it, even if its printing.

4) The telegraph has to work with a number of different types of printers without customer configuration. The telegraph must be able to figure out what type of printer it is attached to.

What architecture might you use for such a system?

Question #3 (40 points) - Based on your answer for question #2, write the pseudo-C code that will be placed in a micro-controller which will run the telegraph.

Question #4 (10 points) - Look at the following code. To which of the architectures we discussed is this architecture most similar in terms of response?

```
static WORD wSignals;
#define SIGNAL_A      0x0001
#define SIGNAL_B      0x0002
#define SIGNAL_C      0x0004
#define SIGNAL_D      0x0008
:
:
void interrupt vHandleDeviceA (void)
{
   !! Reset device A
   wSignals |= SIGNAL_A;
}
void interrupt vHandleDeviceB (void)
{
   !! Reset device B
   wSignals |= SIGNAL_B;
}
:
:
```

```
void main (void)
{
   WORD wHighestPriorityFcn;

   while (TRUE)
   {
      /* Wait for something to happen */
      while (wSignals == 0)
         ;

      /* Find highest priority follow-up processing to do */
      wHighestPriorityFcn = SIGNAL_A;

      disable ();

      /* If one signal is not set . . . */
      while ( (wSignals & wHighestPriorityFcn) == 0)
         /* . . . go to the next */
         wHighestPriorityFcn <<= 1;

      /* Reset this signal; we're about to service it. */
      wSignals &= ~wHighestPriorityFcn;

      enable ();

      /* Now do one of the functions. */
      switch (wHighestPriorityFcn)
      {
         case SIGNAL_A:
            !! Handle actions required by device A
            break;

         case SIGNAL_B:
            !! Handle actions required by device B
            break;
         :
         :
      }
   }
}
```

Question #5 (20 points) - Write pseudo-C code to implement the function queue necessary for the function queue-scheduling algorithm we discussed in class. Your code should have two functions: one to add a function pointer to the back of the queue and one to read the first item from the front of the queue. The latter function can return a NULL pointer if the queue is empty. Be sure to disable interrupts around any critical section of your code.

Question #6 (20 points) - Enhance your code from question 5 to allow functions to be prioritized. The function that adds a pointer to the queue should take a priority parameter. Since this function is likely to be called from interrupt routines, make sure that it runs reasonably quickly. The function that reads items from the queue should return them in priority order.