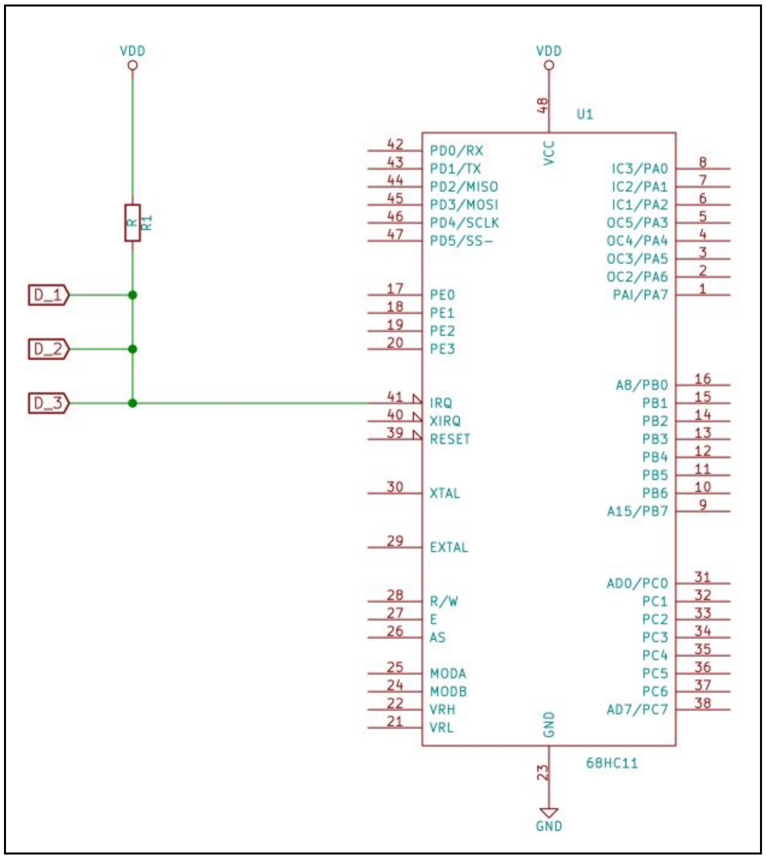
Quiz 1
Grace Livingston

Question 1: If the processor only has one interrupt input line that asserts low, draw a schematic on how you will connect three devices whose interrupt outputs assert low and are open collector? What does "saving the context" mean, when an interrupt occurs?

Answer:



Saving the context includes saving the processor state, such as shared registers, processor status flags, the program counter in some cases, so that when the ISR has completed, it can restore the processor state for the non-interrupt code that was executing before the ISR was entered.

Question 2: Take care of the shared data problem in the code below.

```
static int g_speed[2], g_temperature;
static mutex_t g_engine_mtx;

void interrupt Check_Engine_ISR(void)
{
  /* Lock the data we're updating so that it's not inconsistent for readers. */
  mutex_lock(&g_engine_mtx);
  g_speed[0]       = !! Read engine speed
  g_speed[1]       = !! Read engine speed
  g_temperature  = !!Read engine temperature
  mutex_unlock(&g_engine_mtx);
}

void main (void)
{
  int speed[2], temperature;
 while (TRUE)
 {
   /* Lock the data structures so that the ISR won't change them. */
   mutex_lock(&g_engine_mtx);
   speed[0]       = g_speed[0];
   speed[1]       = g_speed[1];
   temperature  = g_temperature;
   mutex_unlock(&g_engine_mtx);


   if ( (speed[0] != speed[1]) && temperature > 240)
   {
     printf( "System Malfunction");
   }
 }
}
```

Answer:

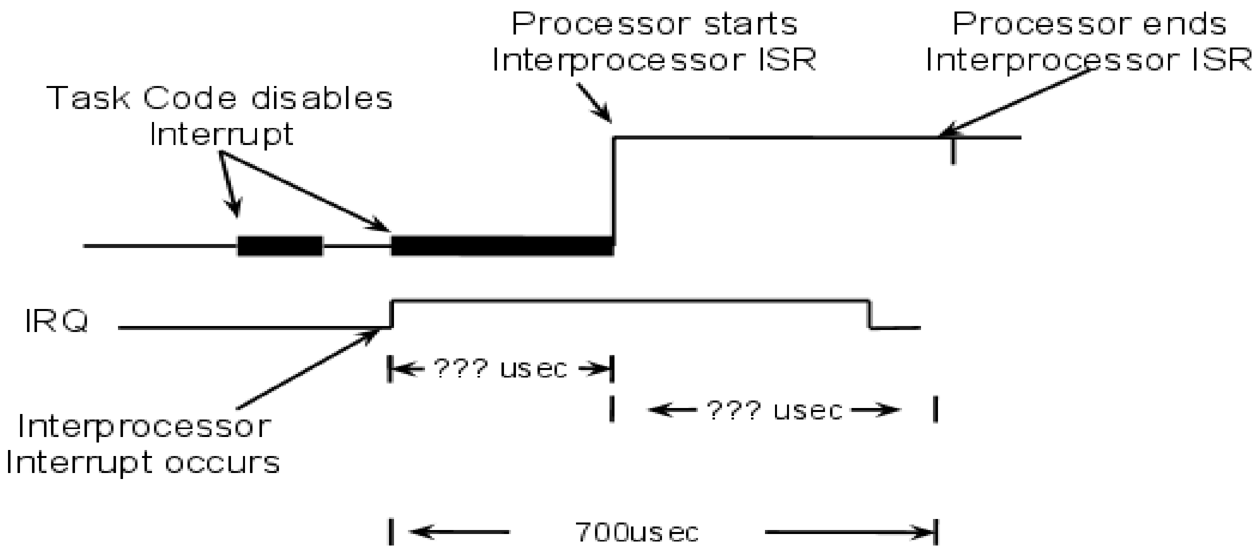
Changes are made in text with light green highlights

Question 3: Your system has the following requirements

A. You have to disable interrupts for 200usec for task code A that updates a pair of temperature shared variables.

B. You have to disable interrupts for 300usec for task code B that gets current time from a shared variable.

C. You must respond within 700usec when you get a high priority interprocessor interrupt


Processor starts Interprocessor ISR
Processor ends Interprocessor ISR
Task Code disables Interrupt
IRQ
??? usec
??? usec
Interprocessor Interrupt occurs
700usec

1. What is the worst case time for the task code to disable the interrupts?
2. What is the maximum time the interprocessor ISR can take to execute?

Answer:

Question 1:

The worst case time for interrupts to remain disabled by the task, assuming it enables them between each time it disables them, is 300 μsec + the disable instruction + the enable instruction. If an interrupt is posted during either of these while they're disabled, the interrupt will be serviced as soon as they're enabled by the processor.

Question 2:

The longest time the ISR would have to execute would then be 700 μsec - 300 μsec - ISR entry latency - context save time - IRQ Enable time, or slightly less than 400 μsec.