CG2271 Real-Time Operating Systems

Tutorial 3

1. There is a new microcontroller from ARM called the Latex M1 that runs at 20MHz. This controller doesn't have PWM capability but it does have a 16-bit Timer module with Interrupt capability which behaves like the Periodic Interrupt Timer in the Cortex M0+. The Timer module is directly clocked by the core clock of 20MHz.

   a. Design the Pseudo Code to generate a 50% Duty Cycle PWM signal using the Timer Interrupt.
   b. What is the period of the PWM waveform?
   c. Another microcontroller Latex M0 doesn't have any Timer module and wishes to still generate a PWM signal with the same period as the Latex M1. Show the Pseudo-Code on how this can be achieved.

2. In your project, you decide to use Serial Interrupts (you don't have a choice anyway…) to capture the data coming in through the Bluetooth interface. The following pseudocode shows a possible implementation.

```
volatile char rx_data; // Global Variable

Serial_ISR
{
  rx_data = Serial_Read_Buffer();
  rx_new_data = 1;
}

Main()
{
      if(rx_new_data == 1)
      {
           rx_new_data = 0;

           if(rx_data == 0x00)
                  move_robot_forward();
           if(rx_data == 0x01)
                  move_robot_right();
           if(rx_data == 0x02)
                  move_robot_left();
           else
                  stop_robot();
      }
      else
           Do_Other_Things();
}
```

a. Describe some issues with the implementation above.

b. The global variable rx_data is declared as volatile. How does it affect the behavior of the robot?

3. To overcome the challenges earlier, you decide to use the circular queue that you learnt in class. In your application, you have the following code for the Serial_ISR and the Main() routine.

```
//Queue declared with a size of 10 (characters)

Serial_ISR
{
  rx_data = Serial_Read_Buffer();

  if(!Queue_Full())
          Q_Enqueue(rx_data);
}

Main()
{
      if(!Queue_Empty())
      {
           my_data = Queue_Dequeue();

           if(my_data == 0x00)
             move_robot_forward();
           if(my_data == 0x01)
             move_robot_right();
           if(my_data == 0x02)
             move_robot_left();
           else
             stop_robot();
      }
      else
           Do_Other_Things();
}
```

a. What does the new implementation resolve? Are there still things to be concerned about?
b. How can we resolve this issue?