Lab 4 Notes

Task 1: PWM synchSM (3 points).

Design and implement a PWM signal on B0: having the duty cycle 60%, and the period of 2000 ms.

Question(s)

1.1 Use the RIBS to draw the PWM synchSM diagram. (2 points)

1.2 Generate timing diagram (in RIMS, Tools->Generate timing diagram). (1 point)

Task 2: Use RIMS' UART (5 points).

UART stands for universal asynchronous receiver/transmitter. When dealing with UARTs, receive is typically written as rx, and transmit as tx. RIMS has a UART that can send data over an additional tx pin, and receive data over an additional rx pin. The UART must first be activated by the RIMS built-in function UARTOn().

For receiving, instead of showing the rx pin, RIMS has a "UART input" text box in which a user may type characters, whose 8 bits (per ASCII) are received serially into a special global variable R. The UART informs the program when new data has been received into R by automatically calling an ISR function "void RxISR()" that the programmer must define. Our convention is to have that ISR set a flag that the program may then read to determine that new data was received by the UART.

Question(s)

2.1 Write a UART function to receive data. Type characters into the UART text box in RIMS, and note that the character's ASCII value appears on B until the next character is typed. (2 points)

2.2 Additionally, when A0 is on, if you type a~z, convert it as uppercase A~Z, and if you type A~Z, convert it as lowercase a~z, but for all the other characters, they stay the same (2 points).

2.3 Print out the character you type in, converted character and its ASCII in “Terminal” window. (1 point)

Task 3: Task scheduler (7 points).

Refer to ***Figure 8.3.3: The LedShow system implemented using task scheduler code*** on the type definition of Task and the scheduler code

3.1 Modify the code Figure 8.3.3 to use a programmer-assigned priority: highest-priority to shortest-period tasks, i.e. **Shortest Period First**. This will include sorting the tasks in the tasks array before entering main()'s while(1) loop. Note: make sure the code works for any number of tasks. (3 points)

3.2 Modify the code Figure 8.3.3 to use a programmer-assigned priority: **Higher Priority First**. This will include updating the task structure to include an unsigned char field named priority; having the user initialize that field when declaring a task; and sorting the tasks in the tasks array before entering main()'s while(1) loop. Write a simple routine for such sorting. Assume a higher unsigned char value means higher priority. Note: make sure the code works for any number of tasks. (4 points)