Quang Nguyen

CPE 409 Lab 9

# Goals

* To obtain a basic understanding of how a Real Time Operating System works.
* To construct a home-brew kernel that will handles 2 tasks.

# Equipment used

## Hardware

* Microchip Explorer 16 board
* PIC kit 3

## Software

* MPLAB X IDE 2.00

# Design Specifications

* The program must use the provided template
* All functions of the program must be perform within the three provided tasks:
  + Unsigned int ButtonPushCounter (void)
    - This task must have the highest priority and scans for the user’s input of a button press on RD6 (S3) and RD7 (S6) every 10 ms.
    - When RD6 is pressed:
      * The task must increment a 16 bits counter and load the LCDDisplay array with the printout of the counter
        + The print out for the LCD must take the form of “Count = XXXXX”
        + XXXXX – the value for of the counter in decimal
    - When RD7 is pressed:
      * The task must throw a flag indicating that the time of the RTC must be display on the LCD
    - Once any of these buttons are pressed:
      * The task must change the scan period from 10 ms to 100 ms.
      * The task must keep this 100 ms period until the button is released.
      * Once the button is released, the task will switch back to a 10 ms scan period.
  + Unsigned int RTCTimer(void)
    - This task must have a mid-priority and runs every 0.5 second
    - Once the flag indicating that the time needs to be printed on the LCD, the task will load the LCDDisplay array with the printout for the time.
      * The print out for the LCD must take the form of “RTC HH:MM:SS.Z”
        + Z indicates the half second resolution
        + SS indicates the value of seconds
        + MM indicates the value of minutes
        + HH indicates the value of hours
        + The indicated time must be in military time.
  + Void WriteLCD (void)
    - This task must have the lowest priority
    - This task must print the content of the LCDDisplay array on the LCD
      * It must print one character at a time.

# Design

* Refer to Figure 1 for the block diagram of the ButtonPushCounter () task

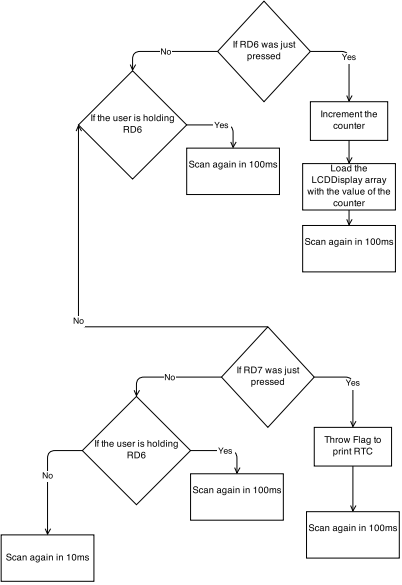


Figure 1: Block diagram for ButtonPushCounter() task

* Refer to Figure 2 for the block diagram of the RTCTimer() task

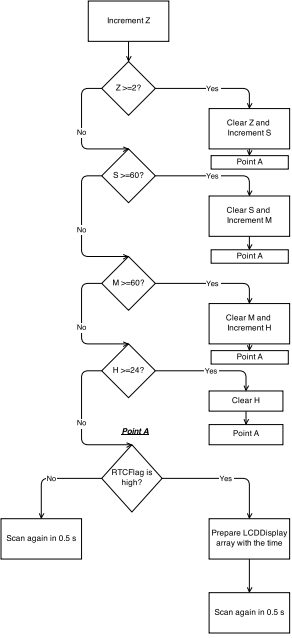


Figure 2: Block diagram for RTCTimer() task

* Refer to Figure 3 for the block diagram of the WriteLCD () task

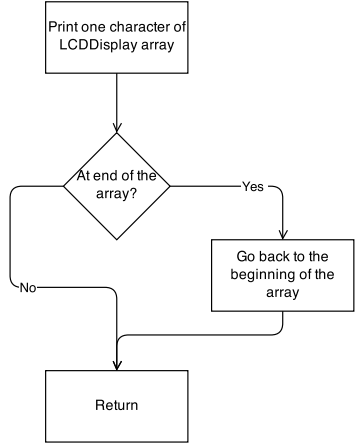


Figure 3: Block diagram for WriteLCD () task

* See attached zip file for the code of the design

# Verification

The program was first tested by pressing RD6. After 100 presses, the LCD did indeed display “Count = 00100”, which confirm that the counter portion of the program is working. The program was then tested by pressing RD7. Once pressed, the LCD did indeed switched to the display for the RTC. RD6 was then pressed to confirm that the LCD display switches back to the counter display. The program was then allow to run for over 1 hour. Once this 1 hour was up, it was confirmed that the LCD output for the RTC display does display the correct value.

# Questions

1. What would happen if the WriteLCD() function took a long time to complete?
   1. The program will not record the presses of the buttons and the RTC time will not be updated.
2. What is the maximum time that WriteLCD () can execute before effecting the other tasks?
   1. 9.99 ms
3. Is our homemade RTOS a preemptive or cooperative scheduler?
   1. Cooperative scheduler
4. Will the RTC have an error due to the ISR that will accumulate over time?
   1. Yes, but this should not be much a problem since the program only sits in the ISR for 2 instruction cycles.

# Conclusions and Limitations

* The program worked as it should and I did not see any limitation to the program