Real-Time Project

Cell Phone Text Message System

Due Date: Final Lab Period

Introduction:

To understand the capabilities and ease-of-design of using an embedded RTOS, we will need a system that would be a nightmare to program as a foreground-background system. For this project, we will design the portion of a cell phone that receives, maintains, and displays text messages and that operates the clock. This will require a number of different RTOS services, carefully decomposing the system into ISR's and tasks, and it will be awesome when it works.

Assignment:

Many of the drivers needed for this project are available as part of the supplied "board support package", or BSP, targeted at the Dragon12+ board. This includes code for the 7-segment displays, the LCD display, and the 16-button keypad. A demo project that uses these is available in the textMsg project (which is a subset of the OS_Probe_LCD available in the C:\Micrium tree). You have already written the core functions for a SPI interface in Week 3's lab. You will need to write drivers for the Port H edge-triggered interrupt, the SCI1 interface, and use the PWM5 (some files and functions are supplied).

Use MicroC/OS-II to implement an application that meets the following requirements:

- Text messages are received at a standard baud rate (i.e. one available on your computer) on SCI1. There is no upper bound on the time between characters. Hint: the files for the Micrium uProbe are included in the textMsg project but aren't used by the example. They contain example code for using the SCI1 device with an interrupt.
- Characters are limited to basic printable ASCII such as letters, numbers, and punctuation; so the system does not need to process other characters with the exception of a carriage return character, and the carriage return signals the end of the message.
- Text messages are limited to a maximum of 160 characters, not including the carriage return (which does not need to be stored with the string).
- Text messages must be stored in the 23K256 SRAM via the SPI interface.

- If there are no stored text messages, the LCD must either be blank or display an appropriate message indicating such.
- If there is at least one message, the LCD must display two lines (32 characters) of the message.
- 5 buttons (programmers choice) on the 16-button keypad must be used for the following functionality: scroll to the next 16 characters of the message (dropping 16 and keeping 16 that were previously there), scroll to the previous 16 characters (as above), jump to the beginning of the next text message, jump to the beginning of the previous text message, and delete the current text message and display either the previous or the next text message.
- The speaker (controlled with PWM5) must play a ringtone when a new text message arrives. Hint/suggestion: Use a task to control PWM5, not an interrupt. Please ask the instructor if you need assistance generating a ringtone.
- The 7-segment displays must increment as a 12-hour clock in actual time. Pushbutton SW2 should increment the hour by one when pressed, and SW3 should increment the minute by one when pressed. Hint: use an edge-triggered interrupt for Port H for these, and a leading "0" on the time is fine, which allows use of the built-in functions.

Suggested Phases

There are no intermediate deadlines for the project. However, steady progress is a must, and the suggested list of phases below gives one outline of the parts of a project that you can start with the material covered in class.

- Phase 1: Modify the example code and implement the clock. Instead of using interrupts to monitor the pushbuttons (at least for the moment), you can use a task that periodically polls the inputs and uses OSTimeDlyHMSM() between samples.
- Phase 2: Implement the ringtone using PWM5. Use pushbutton SW4 or SW5 to simulate the arrival of a text message that triggers the ringtone.
- Phase 3: Transfer to and retrieve text messages from the 23K256 SRAM. The singly linked list for free blocks and doubly linked list for occupied blocks is strongly recommended. (Messages could be stored as constants)
- Phase 4: Implement ISRs for the pushbuttons and the SCI device. The clock is now complete. Get the ringtone to play if a carriage return is received.
- Phase 5: Display a text message and scroll up and down using the buttons.
- Phase 6: Switch between and delete messages using the buttons.
- Phase 7: Receive messages using the SCI interface.

Deliverables

- Your CodeWarrior project in a zip file.
- A project report detailing the ISRs and tasks used with brief purposes, events used (semaphores, mutexes, etc) and their uses, and important/relevant data structures.
- Demonstration of the working project.