Lab 3 report – I/O and scheduling

Student name: Vu Nguyen (71141)

Date: 20.02.2015

Assumptions on knowledge of the reader

It is assumed that a reader should know some basic programming skills in C/C++ in order to understand the source code. A basic knowledge of hardware such as processors, LEDs, pin ports, interrupt, and hardware timers should be known.

Description of the task

The task of lab 3 is to implement a simple traffic light system as a multi-state machine. The system should work as follows:

- Initialize to amber blinking and go to operation after 5 seconds.
- Stay in Red state for 5 seconds and change to Green state and stay there for 5 seconds.
- After 5 second passes, change to Amber state and stay there for 0.5 second
- After 0.5 second passes, change Red state again.
- When a button on the expansion board is pressed, change to Amber state immediately if the system is in Red state.
- The multi-state machine is to be implemented using time/input based transitions between states.
- Updating of the task is done by timer interrupt. In other words, interrupt service routine updates the state machine
- The super loop does nothing but only enters low-power mode (e.g sleep, idle, …)

Description of the equipment used

The equipment used consists of a Modtronix SBC65EC embedded platform with built in web server and bootloader. A PC with Microchip MPLAB IDE/MCC18 compiler is installed to write and compile the C source code. Besides, Modtronix Network Bootloader must be installed in the development PC to program the executable to the PIC microcontroller of the platform. An Ethernet cable is used to connect the platform to the local area network. A regular 2.1mm jack plug for power supply (7-35VDC) is used to power up the platform. The blue button on the expansion board will be used. This button is configured as pull-up.

Description of the performed work

From the task description, three software components are created: PES_eos, PES_ports, and PES_traffic_lights. A software component means it consists of a header file and a .C file for detailed implementation. The PES_ports component is already implemented in lab 2. Therefore, in this lab only two components will need to be implemented. The PES_eos component acts as a simple embedded

operating system for the Modtronix SBC65EC. It will interrupt every 50 ms, and each time an interrupt occurs the state machine will be updated. The PES_traffic_light implements the actual state machine of the system.

Implementation of the embedded operating system

The embedded operating system is implemented using Timer0 of the PIC18f6627. It is configured in such a way that when the timer overflow occurs, an interrupt will ignite. In the interrupt service routine of Timer0, it will reload the timer0 again and call traffic_lights_update() function to update the state machine of the traffic light system. It is important to note that Timer0 does not reload itself automatically. Therefore, in the ISR the high byte and low byte registers of Timer0 are reloaded every time a timer interrupt occurs. Besides, the reloading order is an important point to note. High byte register of Timer0 must be reloaded first, and then the low byte register of Timer0. The PES_eos component also has a function call PES_eos_go_to_idle(). This function put the microcontroller into idle mode to save energy consumption.

Implementation of the traffic light system

The PES_traffic_light implements the state machine of the system. It will be called every 50ms by the interrupt service router (ISR) of the PES_eos component. Each call will increment the internal state counter which determines how long the system is in that state. Based on the requirements of the traffic light system, a detailed implementation of the PES_traffic_light component is done accordingly. This can be seen in the PES_traffic_light.h header file and the PES_traffic_light.c implementation file.

Achieved results

The system works correctly the way it was specified. Initially, it blinks the yellow LED for 5 seconds. After that it goes to green state for 5 seconds. After 5 seconds passes, it changes to yellow state and stays there for 0.5 second. After that it changes to red state and state there for 5 seconds before coming back to the green state. When the system is in the red state, if the blue button on the expansion board is pressed, the system will change to yellow state immediately for 0.5 second before changing to the green state.