Lab 2 report – Task based software architecture

Student name: Vu Nguyen (71141)

Date: 06.02.2015

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 should be known.

Description of the task

The task of lab 2 is to implement a program using task based software architecture. The tasks are as follows:

- Blinking the LED
- Reading input from an input pin in which the LED stops blinking when a button/key is pressed
- A variable delay in ms using timers

In task based software architecture, the tasks/ functions are divided into modules by means of header files (.h). The main application will run three tasks: blinking the LED, waiting for a button press, and delay between LED on/off blinking.

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.

Description of the performed work

From the task description, three software components are created: PES_blink_led.h, PES_ports.h, PES_delay.h. The prefix "PES_" will be used throughout the course when naming a user-created header file to distinguish the system header files. The PES_leds.h together with PES_leds.c will be used to implement LED blinking task. The PES_ports.h and PES_ports.c will be used to handler I/O pins of the platform (reading/writing value from or to a port pin). In this lab, reading an input pin is implemented in this component. The PES_delay.h and PES_delay.c component will be used to create delay by using timer 0 of the PIC18f6627 microprocessor.

Implementation of delay function

The delay function is implemented by using Timer0. Initial value is loaded to the timer and the timer starts counting until it overflows and rolls backs to 0 again. Two functions of the delay component are:

void initTimer0(void);

This function turns on timer 0, configures the timer as 16-bit mode, chooses internal instruction cycle clock which is Fosc/4. Note that the internal instruction cycle clock is 40Mhz even though the oscillator frequency is 10Mhz as shown in the schematics of the board. The scaling of the frequency is done by configuring the PLL (Phase Locked Loop). The prescaler is not assigned; therefore, timer 0 clock input bypasses precaler. For some reason, interrupt must be disabled for the overflow to occur. Hence, the interrupt flag of timer 0 is cleared.

void delay_ms(unsigned int ms);

This function, as the name implies, implements a delay interval in millisecond. It takes as input *ms* milliseconds delay interval and hangs there until *ms* milliseconds is passed. There is a for loop inside the function. The for loop iterates every one millisecond until it reaches *ms* value.

Implementation of blinking LED function

The blinking LED function defines three LEDs on the expansion board stacked on top of the main SBC65EC development board. These are red, yellow, and green LEDs. The prototype of the function in this component is as follows

void blink_led(unsigned char led);

This function takes as argument the color of a LED and toggles that LED every time it is called. The parameters for the argument are: LED_RED, LED_YELLOW, and LED_GREEN for red LED, yellow LED, and green LED, respectively.

Implementation of pin reading function

The pin reading function is implemented in the PES_ports.c and PES_ports.h component. This component in this lab only defines two port pins: port B pin 0 and port B pin 1 for the red push button and the blue push button on the expansion board, respectively. The function used for reading a port pin in this component is defined in the following way.

int readPin(unsigned char port_pin);

This function takes as argument the name of the port pin and returns the value read from that port pin. The return value 0 indicates that port pin is in low logic state, while return value of 1 indicates high logic state.

Achieved results

In the infinite loop of the main application, yellow LED is blinked every one second and port B pin 1 (the blue push button on the expansion board) is checked every iteration. If the blue push button is pressed, the yellow LED stops blinking. If not pressed, the yellow LED blinks every one second.

Observed result: The yellow LED blinks every one second when the yellow push button is not pressed. When the button is pressed, the LED stops blinking (but not immediately since it has to go through the loop).