Programming Embedded Systems 2014 / JB

Exercise 3 / 29/31.1.2014 / Deadline for submitting report 14.2.2014

Return report electronically on address: https://abacus.abo.fi/ro.nsf. If you do not have an ÅA account, please email report to Åke Syysloiste <agustavs@abo.fi>.

Advisor/labs: Åke Syysloiste. Åke will be available during lab hours, at other times he can be found in room A5031.

Equipment and tools

Equipment used: Modtronix SBC65EC single-board computer + daughter board
PC with Microchip MPLAB IDE / MCC18-compier (both can be downloaded for free from microchip home page: www.microchip.com)

Task

Implement a simple traffic light system as a Multi-State machine. The system should work the following: should initialize to amber blinking, after 5 seconds it should go to operation. Red (5 s) -> Amber+Red(0,5 s)-> Green (5 s) -> Amber (0,5 s)->Red. A button press will switch to green (via Red+Amber) immediately (if in Red state).

Using exercise 2 as a starting point, this time the system will be enhanced in the following ways:

- Implement a Multi-State machine
  - Implement the states
  - Timed / input based transitions between states

- Interrupt based task updating
  - Empty super loop
  - Interrupt service routine activates the update of state machine
- Low power mode
  - Low power mode can be tested using Microchip command SLEEP (for details, see the datasheet).

Registers for I/O:

TRIS – register for controlling I/O-port direction

```
TRISBbits.TRISB6 = 0; // pin 6 on port B is  set as output
```

LAT – register: latch for output ports

```
LATBbits.LATB6 = 1; // pin 6 on port B is set high
```

PORT – value on port (for reading input)

```
myval = PORTBbits.RB6;
```

Note that register NAME-ADDRESS mapping is found in the via the “p18cxxx.h”, which dependent on your architecture is mapped to the file representing the actual hardware (depending on precompiler definitions). In this case the file “18f6627.h”, which is found in the MCC18 installation folder h-directory (often “c:\mcc18\h”).

Programmable timers on the Microchip 18F6627. The microcontroller has 4 16/8 bit timers, which are programmable. The T0CON register controls the behavior for timer 0. The TMR0L and TMR0H are the timer 0 counter values (low and high bytes). The timers can be controlled for interrupts, but here only overflow (TMR0L/H == 0).

In order to enable interrupts at regular intervals, a timer must be programmed. For timer 0:

```
TMR0L – Low byte for 16-bits timer
TMR0H – High byte for 16 bits timer
T0CON – Timer control (ce set to 0b00000001)
T0CONbits.TMR0ON  - Timer on/off
INTCONbits.TMR0IE – Enable interrupts on timer 0
INTCONbits.TMR0IF – Set if overflow
```

The timers has to be reset on each interrupt by rewriting to the registers TMR0L, TMR0H, and resetting TMR0IF.

Interrupts:

Define in the main routine so that the interrupt vector is in the right place. In a Microchip 18F6627 PIC with bootloader, the high priority interrupt vector is at position 0x808.

(In a bootloader-enabled PIC, everything is shifted from position 0x0 to 0x800).

```
#pragma interrupt HighISR save=section(".tmpdata")
void HighISR(void) {
  if (INTCONbits.TMR0IF) {  /* Check: Is it a timer interrupt?
          ...
 }
}

#pragma code highVector=0x808
void HighVector (void)
{
    _asm goto HighISR _endasm
}
#pragma code /* return to default code section */
```

SLEEP / idle mode in Microchip.

The Microchip uC:s have different power saving modes. Sleep mode means that all oscillators are off, and is entered by executing the assembler instruction SLEEP. The sleep requires that some systems is enabled for getting the uC out of sleep mode. Easier is to use device idle mode, by setting the IDLEN bit to 1 (OSCCONbits.IDLEN=1), and after this execute the SLEEP instruction. Now oscillators are still running, and for instance timer interrupts work normally. The assembler instruction SLEEP can be achieved by using inline assembler: `_asm sleep __endasm`

Document what you have done, and submit the documentation and the code you have produced electronically to the address give above.

General rules for documenting projects:

Each report should include:

- Title
- Name
- Date / timeframe when exercise performed
- Group (if not done individually)
- Assumptions on knowledge of the reader
- Own contribution (if performed in group)
- Description of the task / exercise
- Description of the equipment used
- Description of performed work
- Achieved results