EXPERIMENT 5: The Interrupt and Polling System.

Objective:
- To Study the Interrupt and Polling methods
- To apply the interrupt and polling concept in a program.

Pre-lab: Procedure 1

Introduction

Polling

Polling method uses a status flags to indicate whether it has a valid data for the microprocessor to proceed with the next instruction. This method will wait in a loop for the flag to be raised. By checking the flag bit continuously, the microprocessor can determine whether the interface chip has received a new data from the input device. In the polling method, the system needs to wait until the next loop in order to check the flag.

Interrupt

Interrupt is a mechanism provided by a microprocessor or a computer to synchronous I/O operation, handle error condition and emergency, coordinate the use of share resources and so on. An interrupt is an event that requires the CPU to stop normal program execution and perform service related to the event. An interrupt can be generated internally or externally. An external interrupt can be generated when external circuitry assert an interrupt signal to the CPU. An internal interrupt can be generated by the hardware circuitry inside the chip and caused by software errors. Abnormal situations that occur during program execution, such as illegal opcodes, overflows, divide-by-zero, and underflow, are referred to as software interrupts. Software interrupts are also called traps and exceptions.

The PIC18F458 Interrupts

The PIC18F458 devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level or a low priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h. High priority interrupt events will override any low priority interrupts that may be in progress.

There are 13 registers that are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

Each interrupt source has three bits to control its operation. The functions of these bits are:
1. Flag bit to indicate that an interrupt event occurred
2. Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set
3. Priority bit to select high priority or low priority.
The interrupt priority feature is enabled by setting the IPEN bit (RCON register). When interrupt priority is enabled, there are two bits that enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts. Setting the GIEL bit (INTCON register) enables all interrupts that have the priority bit cleared. When the interrupt flag enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 000008h or 000018h, depending on the priority level. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PICmicro® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. The PEIE bit (INTCON register) enables/disables all peripheral interrupt sources. The GIE bit (INTCON register) enables/disables all interrupt sources. All interrupts branch to address 000008h in Compatibility mode.

When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High priority interrupt sources can interrupt a low priority interrupt.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

The “return from interrupt” instruction RETFIE, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used), which re-enables interrupts. For external interrupt events, such as the INT pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set regardless of the status of their corresponding enable bit or the GIE bit.

Below is a template to write interrupt program:

```assembly
#include <p18F458.inc>
org 0x00

goto MAIN

ISR_hi RCALL HI_ISR ;address high priority interrupt = 0x08
RETFIE
NOP
NOP
NOP
NOP
CLRF 0x00
CLRF 0x1A

ISR_lo RETFIE ;address low priority interrupt = 0x18

MAIN ;type the main program here ...
...
...
RCALL POLLING
...
WAIT BRA WAIT
POLLING Type the POLLING program here
...
RETURN

HI_ISR ;type the interrupt program here
...
RETFIE FAST
```
Procedure

1. Write a program based on the flowcharts given in Figure 5.1 on page E5-5 as a guide. The program is divided into 4 sections and should follow the following specifications:
   - Main program – the main program cycles through a continuous loop and outputs to PORTC with interfacing wires connected to a 7-Segment display. The 7-segment display must display the numeral number as shown below. The main program must contain Interrupt initialization to facilitate an interrupt request.
   
   0          1                 2  … …             9

   - Delay routine – to provide time delay of 5-second (in-between the display output, please refer lab 4 for the delay routine).
   - Interrupt service routine – a routine that will service the INT0 request. The interrupt routine request shall be made through SW1. It shall turn ON LED one by one from bit 0 to bit 7 at PORTD. The routine should also contain a delay of 0.5-second in between in order to on the LEDs. When the Bit 7’s LED is on, program control shall be passed back to the main program.
   - Polling service routine – the polling service routine shall be call by pressing SW2.

   Note: The flow charts given are simplified; you will need to expend it to more detail. Also you may want to modify’ the sequence, or design an altogether different program flow, but do keep to the program specifications described above.

   Tips: You’ll need a 5-sec timer. You can use modify delay routine from experiment 4.

2. Set-up the circuit in Figure 5.2 (refer to page E5-6).

   Note: Common the ground point of the board external DC supply and circuit's external DC supply. (Important: please check all the connections are connected accordingly before power up - you may ask your instructor to check).

3. Assemble the program you’ve written and then download it.

4. Test your program on the board. Observe and copy the monitor responses.

   Q1. Is the output response the same as before pushing the button SW1?
   Q2. At the end of interrupt, does the number continues or restart from ‘0’?
   Q3. What is the different when pushing SW1 and SW2?
   Q4. Discuss your answer in Q3.

5. Turn off external power supply and logoff.

Important note:
(Plagiarism: Adopting, reproducing or copying others' work, ideas, words or statements without an appropriate acknowledgement constitutes plagiarism and will be heavily penalized.)
Components List:
1. Breadboard
2. 2 x pushbutton
3. 2 x 4.7kΩ resistor
4. 9 x 330Ω resistor
5. 1 x 7447 7-segment decoder
6. 1 x 7-Segment (common anode)
7. 8 x LED

Figure 5.1: The Flowchart
Figure 5.2: Schematic for the connection