Lab 3. LED Control and Interruptions.
Lab Sessions

Lab 1. Introduction to the equipment and tools to be used in the lab, which include the development board (PICDEM 2 Plus), the microcontroller (PIC 16F1937), and the USB Oscilloscope (Analog Discovery).


Lab 3. Experiment 7: LED Control and Interruptions.

Lab 4. Experiment 10: Stepper Motor Motion Control.

Lab 5. Experiment 11: DC Motor Speed Control Using PWM.

Lab 1. Outline

- Objective.

- Lecture
  - Introduction.
  - Oscillator Module.
  - Input/output Ports.
  - Interrupts
  - Programming on MPLAB X

- Time for the experiments.
Objectives

- Learn how to program the PIC 16F1937 using MPLAB X.

- Design a program to control 4 LEDs using the microcontroller (PIC) and implement it in the PICDEM 2 Plus board.

- Design a program with an Interrupt Service Routine and implemented in the PICDEM 2 Plus board.
Every computer, no matter how simple or complex, has at its heart two things: a **CPU and some memory**. Together, these two things are what make it possible for your computer to run programs.
A microcontroller is a simplified version of a similar architecture, placed on one chip.
Data Memory (RAM)

- Memory: 512 bytes
  - Divided in 32 banks with 128 bytes in a bank.
    - 12 core registers
    - 20 Special Function Registers (SFR)
    - Up to 80 bytes of General Purpose RAM (GPR)
    - 16 bytes of common RAM
Oscillator Module

- Clock sources can be external or internal.
  - **External clock sources:** Quartz crystal resonators, ceramic resonators, and RC circuits.
  - **Internal clock sources:** Two internal oscillators, with a choice of speeds selectable via software.
Clock source modes are selected by the FOSC <2:0> bits in the configuration Word 1. The FOSC bits determine the type of oscillator that will be used when the PIC is first powered.

The oscillator module can be configured as:

1. **ECL** - External Clock Low-Power mode (0 MHz to 0.5 MHz)
2. **ECM** - External Clock Medium-Power mode (0.5 MHz to 4 MHz)
3. **ECH** - External Clock High-Power mode (4 MHz to 32 MHz)
4. **LP** - 32 kHz Low-Power Crystal mode.
5. **XT** - Medium Gain Crystal or Ceramic Resonator Oscillator mode (up to 4 MHz)
6. **HS** - High Gain Crystal or Ceramic Resonator mode (4 MHz to 20 MHz)
8. **INTOSC** - Internal oscillator (31 kHz to 32 MHz).
**REGISTER 4-1: CONFIGURATION WORD 1**

<table>
<thead>
<tr>
<th></th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC0EN</td>
<td>IES0</td>
<td>CLKOUTEN</td>
<td>BOREN1</td>
<td>BOREN0</td>
<td>CP0D</td>
<td>CP0</td>
<td></td>
</tr>
</tbody>
</table>

bit 13

<table>
<thead>
<tr>
<th></th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCLRE</td>
<td>PWRTE</td>
<td>WDTE1</td>
<td>WDTE0</td>
<td>FOSC2</td>
<td>FOSC1</td>
<td>FOSC0</td>
<td></td>
</tr>
</tbody>
</table>

bit 6

---

**FOSC<2:0>: Oscillator Selection bits**

- **111**: ECH: External Clock, High-Power mode: CLKin on RA7/OSC1/CLKin
- **110**: ECM: External Clock, Medium-Power mode: CLKin on RA7/OSC1/CLKin
- **101**: ECL: External Clock, Low-Power mode: CLKin on RA7/OSC1/CLKin
- **100**: INTOSC oscillator: I/O function on RA7/OSC1/CLKin
- **011**: EXTRC oscillator: RC function on RA7/OSC1/CLKin
- **010**: HS oscillator: High-speed crystal/resonator on RA6/OSC2/CLKOUT pin and RA7/OSC1/CLKin
- **001**: XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT pin and RA7/OSC1/CLKin
- **000**: LP oscillator: Low-power crystal on RA6/OSC2/CLKOUT pin and RA7/OSC1/CLKin
The system clock speed can be selected via software by setting up the bits IRCF<3:0> of the OSCCON register.

One of the following frequencies can be selected:

- 32 MHz (requires 4X PLL)
- 16 MHz
- 8 MHz
- 4 MHz
- 2 MHz
- 1 MHz
- 500 kHz (Default after Reset)
- 250 kHz
- 125 kHz
- 62.5 kHz
- 31.25 kHz
- 31 kHz (LFINTOSC)
REGISTER 5-1: OSCCON: OSCILLATOR CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPLLEN</td>
<td>IRCF&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SCS&lt;1:0&gt;</td>
<td></td>
</tr>
</tbody>
</table>

bit 7
To set up the clock of the PIC we need to set some bits of the CONFIG1 Register and the OSCCON Register.

1. **CONFIG1**: FOSC bits (Choose type of clock Internal/external)
2. **OSCON**: IRFC bits (Choose oscillation frequency)
Input/output Ports

- 5 ports (A, B, C, D, and E). Each port has different characteristics and number of pins.

- Ports are bidirectional (Input/Output).

- Multi-functional. Some pins are multiplexed with an alternate function. When a peripheral is enable, that pin may not work as a general input/output.
Each port has three standard registers for its operation:
- TRISx registers (data direction)
- PORTx registers (reads the levels on the pins)
- LATx registers (output latch)

Some ports may also have one or more of the following registers:
- ANSELx (analog select)
- WPUx (weak pull-up)
- INLVLx (input level control)
Example Port A

- TRISA register (PORTA Data Direction Register):
  - Bit = 1 (pin on port A = Input)
  - Bit = 0 (pin on port A = Output)

- PORTA reads the status of the pins.
  - Example: x = PORTA

- LATA puts the content of the output latch on the port A.
  - Example: LATA = 0; Make all the pins of port A equal to 0 V.

**Note:** Writes to PORTA are actually written to corresponding LATA register. Reads from PORTA register is return of actual I/O pin values.
### REGISTER 12-3: TRISA: PORTA TRI-STATE REGISTER

<table>
<thead>
<tr>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
</tr>
</tbody>
</table>

- **TRISA<7:0>:** PORTA Tri-State Control bit
  - 1 = PORTA pin configured as an input (tri-stated)
  - 0 = PORTA pin configured as an output

### REGISTER 12-2: PORTA: PORTA REGISTER

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA7</td>
<td>RA6</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</td>
</tr>
</tbody>
</table>

- **RA<7:0>:** PORTA I/O Value bits\(^{(1)}\)
  - 1 = Port pin is > VIH
  - 0 = Port pin is < VIL

### REGISTER 12-4: LATA: PORTA DATA LATCH REGISTER

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>LATA7</td>
<td>LATA6</td>
<td>LATA5</td>
<td>LATA4</td>
<td>LATA3</td>
<td>LATA2</td>
<td>LATA1</td>
<td>LATA0</td>
</tr>
</tbody>
</table>

- **LATA<7:0>:** PORTA Data Latch bits
Interrupts

- The interrupt feature allows certain events to pre-empt normal program flow.

- The following events happen when an interrupt event occurs while the GIE bit is set:
  - Current prefetched instruction is flushed
  - GIE bit is cleared
  - Current Program Counter (PC) is pushed onto the stack
  - Critical registers are automatically saved to the shadow registers
  - PC is loaded with the interrupt vector 0004h
Interrupts are disabled upon any device Reset. They are enabled by setting the following bits:

- GIE bit of the INTCON register
- Interrupt Enable bit(s) for the specific interrupt event(s)
- PEIE bit of the INTCON register (if the Interrupt Enable bit of the interrupt event is contained in the PIE1, PIE2 and PIE3 registers)
bit 7  GIE: Global Interrupt Enable bit
       1 = Enables all active interrupts
       0 = Disables all interrupts

bit 6  PEIE: Peripheral Interrupt Enable bit
       1 = Enables all active peripheral interrupts
       0 = Disables all peripheral interrupts

bit 5  TMR0IE: Timer0 Overflow Interrupt Enable bit
       1 = Enables the Timer0 interrupt
       0 = Disables the Timer0 interrupt

bit 4  INTE: INT External Interrupt Enable bit
       1 = Enables the INT external interrupt
       0 = Disables the INT external interrupt

bit 3  IOCIE: Interrupt-on-Change Enable bit
       1 = Enables the interrupt-on-change
       0 = Disables the interrupt-on-change

bit 2  TMR0IF: Timer0 Overflow Interrupt Flag bit
       1 = TMR0 register has overflowed
       0 = TMR0 register did not overflow

bit 1  INTF: INT External Interrupt Flag bit
       1 = The INT external interrupt occurred
       0 = The INT external interrupt did not occur

bit 0  IOCIF: Interrupt-on-Change Interrupt Flag bit
       1 = When at least one of the interrupt-on-change pins changed state
       0 = None of the interrupt-on-change pins have changed state

Note 1: The IOCIF Flag bit is read-only and cleared when all the Interrupt-on-Change flags in the IOCBF register have been cleared by software.
Interrupt-On-Change (IOC)

- Port B pins can be operated as Interrupts-On-Change pins.
- An interrupt can be generated by detecting a signal that has either a rising edge or falling edge.
- To enable the IOC pins the IOCIE bit of the INTCON register must be set.
Individual Configuration

- IOCBPx bit in the IOCBP register has to be set to detect a rising edge.

<table>
<thead>
<tr>
<th>REGISTER 13-1: IOCBP: INTERRUPT-ON-CHANGE POSITIVE EDGE REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0/0</td>
</tr>
<tr>
<td>IOCBP7</td>
</tr>
</tbody>
</table>

- IOCBNx bit in the IOCBN register has to be set to detect a rising edge.

<table>
<thead>
<tr>
<th>REGISTER 13-2: IOCBN: INTERRUPT-ON-CHANGE NEGATIVE EDGE REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0/0</td>
</tr>
<tr>
<td>IOCBN7</td>
</tr>
</tbody>
</table>

- The IOCBFx bits located in the IOCBF register are status flags that correspond to the interrupt-on-change pins of port B.

<table>
<thead>
<tr>
<th>REGISTER 13-3: IOCBF: INTERRUPT-ON-CHANGE FLAG REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W/HS-0/0</td>
</tr>
<tr>
<td>IOCBF7</td>
</tr>
</tbody>
</table>
Programing on MPLAB X

- MPLAB X Interface.
- Programing on C language.
- Compiler XC8
- Programing the PIC via PicKit 3 Programer
Assembly VS C Language

EXAMPLE 12-1:  INITIALIZING PORTA

; This code example illustrates
; initializing the PORTA register. The
; other ports are initialized in the same
; manner.

BANKSEL PORTA  ;
CLRF PORTA     ;Init PORTA
BANKSEL LATA   ;Data Latch
CLRF LATA      ;
BANKSEL ANSELA ;
CLRF ANSELA    ;digital I/O
BANKSEL TRISA  ;
MOVLW B'00111000' ;Set RA<5:3> as inputs
MOVWF TRISA    ;and set RA<2:0> as
               ;outputs

// Initializing Port A
TRISA = 0b00111000
LATA = 0x00
Programming the PIC
Thanks