Hashemite University

Faculty of Engineering

Mechatronics Engineering Department

Microprocessors and Microcontrollers Laboratory

Experiment 1
PIC18F452 Assembly Instructions

This experiment focuses on the Assembly Language Instructions for PIC18F452 device.

Objectives:

1. To make the student familiar with the assembly instructions of PIC18F452 device.
2. To know the way of choosing the appropriate instructions and how to arrange them according to the application requirements

Apparatus:

The devices used in this experiment are:

1. Programmer
2. PIC18F452 IC
3. Breadboard and electronic components as shown in Figure 1.1

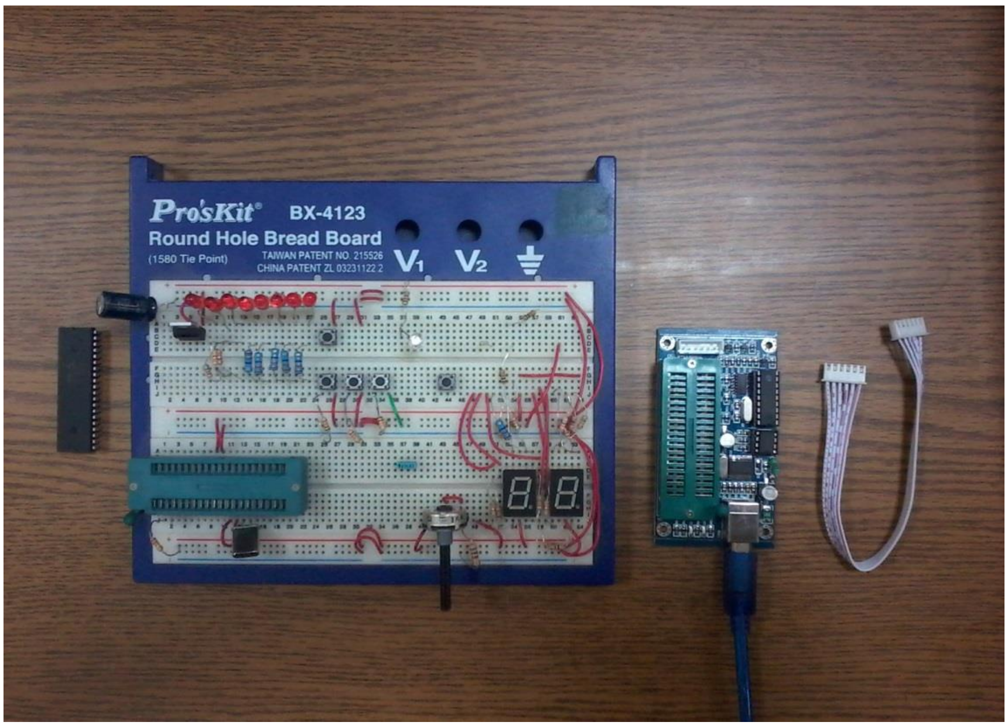
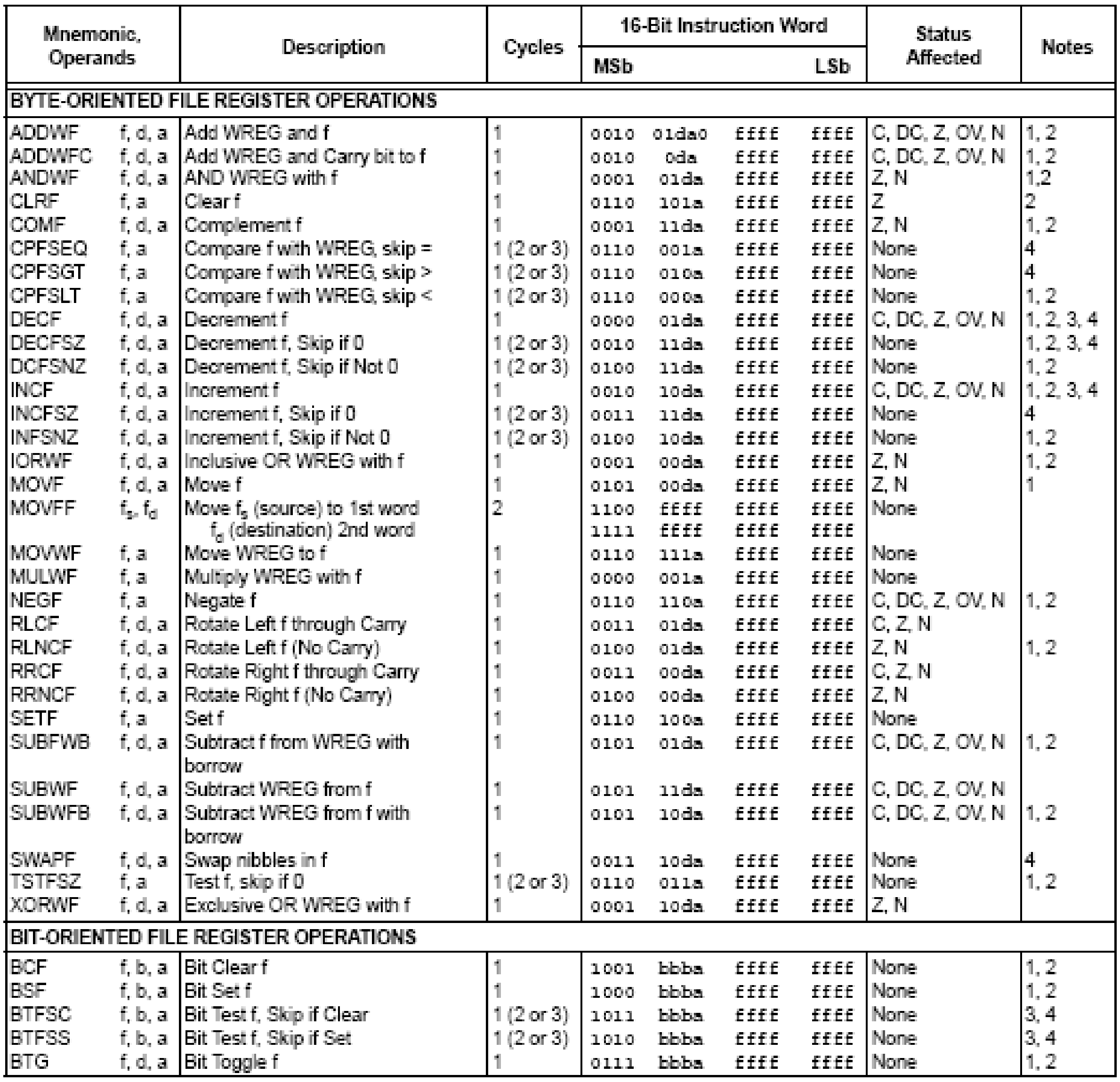


Figure 1.1: Equipment and devices used in the experiment.

Theoretical Background:

Microcontrollers are very important devices in the digital world, so it is necessary to know how to program them, and in order to do this you may use assembly language instructions for PIC18F452 device, and by referring to the device datasheet the following table (Table 1.1) illustrates those instructions.

Table 1.1



Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da0	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	0da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 1111	ffff ffff	ffff ffff	ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	1, 2
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	1, 2
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	1, 2
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2

Mnemonic,	Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	2	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	1 (2)	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	n, s	Call subroutine1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	n	Go to address1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software device RESET	1	0000	0000	1111	1111	All	
RETFIE	s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

Mnemonic,	Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
LITERAL OPERATIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word	2	1110	1110	00ff	kkkk	None	
		to FSRx 1st word		1111	0000	kkkk	kkkk		
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS									
TBLRD*		Table Read	2	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2 (5)	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

Procedure:

- Write a program that adding the lower 4-bits of the input data at portc to the higher 4-bits and moving the result to portb register.
- Build the required circuit.
- Program the device and make sure the circuit is ready to turn the power on.
- Consult your Lab. Supervisor to turn the power on.

Discussion and Analysis:

Observe the execution of the code in the circuit and make your decision about your program if it matches the application requirements or not.

Experiment 2
Input and Output Ports

This experiment focuses on dealing with Input and Output Ports for PIC18F452 device.

Objectives:

1. To make the student familiar with the Input and Output Ports of PIC18F452 device.
2. To know the way of choosing the port to be input or output according to the application requirements

Apparatus:

The devices used in this experiment are:

1. Programmer
2. PIC18F452 IC
3. Breadboard and electronic components as shown in Figure 2.1

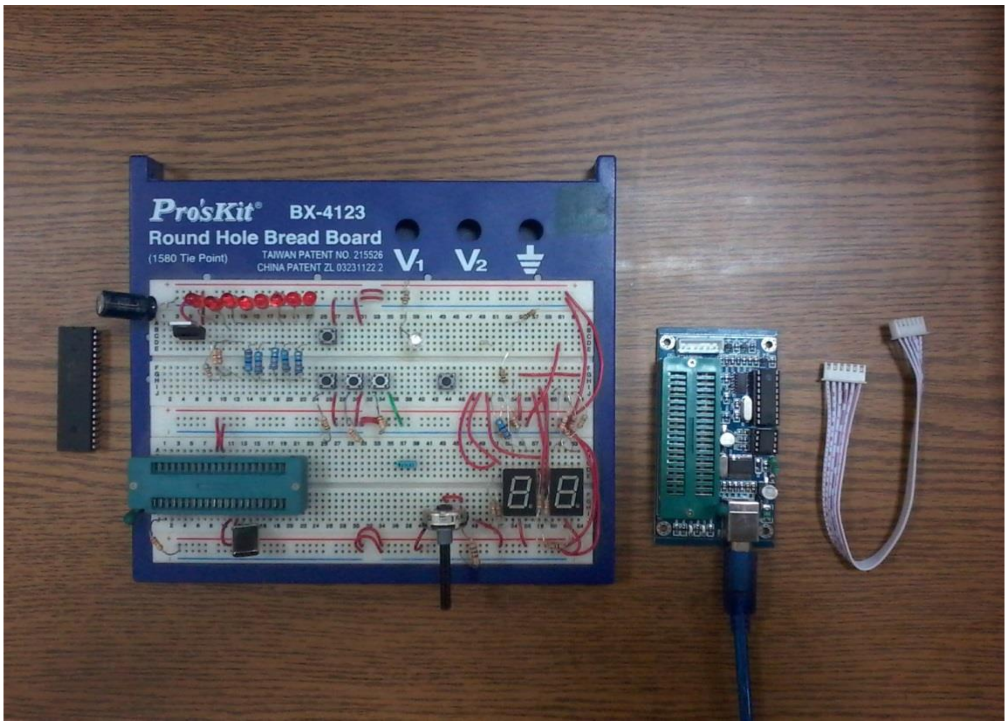


Figure 2.1: Equipment and devices used in the experiment.

Theoretical Background:

PIC18F452 Microcontroller has five I/O ports (PortA, PortB, PortC, PortD, and PortE) that can be defined as input or output ports according to application.

Referring to PIC18F452 datasheet at chapter 9:

9.0 I/O PORTS

Depending on the device selected, there are either five ports or three ports available. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Each port has three registers for its operation. These registers are:

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)

The data latch (LAT register) is useful for read-modify-write operations on the value that the I/O pins are driving.

9.1 PORTA, TRISA and LATA Registers

PORTA is a 7-bit wide, bi-directional port. The corresponding Data Direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch.

The Data Latch register (LATA) is also memory mapped. Read-modify-write operations on the LATA register reads and writes the latched output value for PORTA.

The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers.

The other PORTA pins are multiplexed with analog inputs and the analog VREF+ and VREF- inputs. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

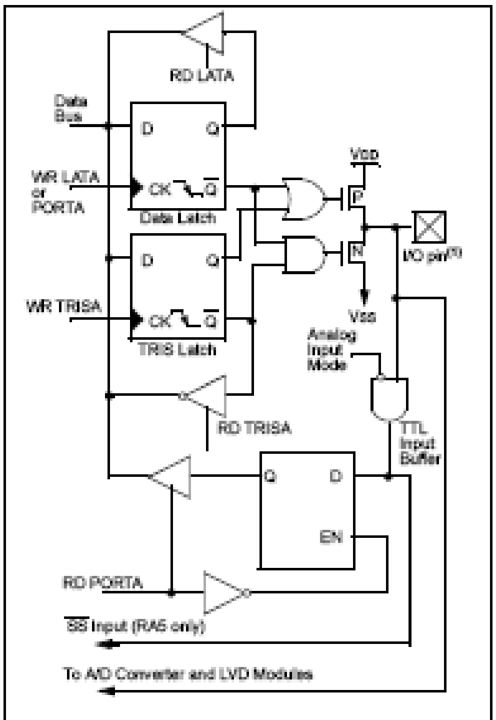
Note:	On a Power-on Reset, RA5 and RA3:RA0 are configured as analog inputs and read as '0'. RA6 and RA4 are configured as digital inputs.

The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 9-1: INITIALIZING PORTA

```
CLRF  PORTA       ; Initialize PORTA by
                  ; clearing output
                  ; data latches
CLRF  LATA        ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW 0X07        ; Configure A/D
MOVWF ADCON1      ; for digital inputs
MOVLW 0XCF        ; Value used to
                  ; initialize data
                  ; direction
MOVWF TRISA       ; Set RA<3:0> as inputs
                  ; RA<5:4> as outputs
```

FIGURE 9-1: BLOCK DIAGRAM OF RA3:RA0 AND RA5 PINS



Note 1: I/O pins have protection diodes to VDD and VSS.

Procedure:

- Write a program that increases the value at portb by 1 if the button at portc-0 was pressed, decreases portb value if the button at portc-1 was pressed and moving the portb data to portd if the button at portc-2 was pressed.
- Build the required circuit.
- Program the device and make sure the circuit is ready to turn the power on.
- Consult your Lab. Supervisor to turn the power on.

Discussion and Analysis:

Observe the execution of the code in the circuit and make your decision about your program if it matches the application requirements or not.

Experiment 3
Wait Code Implementation

This experiment focuses on implementing wait code using PIC18F452 device.

Objectives:

1. To make the student familiar with the wait code implementation in PIC18F452 device.
2. To know how to change the wait time according to the application requirements.

Apparatus:

The devices used in this experiment are:

1. Programmer
2. PIC18F452 IC
3. Breadboard and electronic components as shown in Figure 3.1

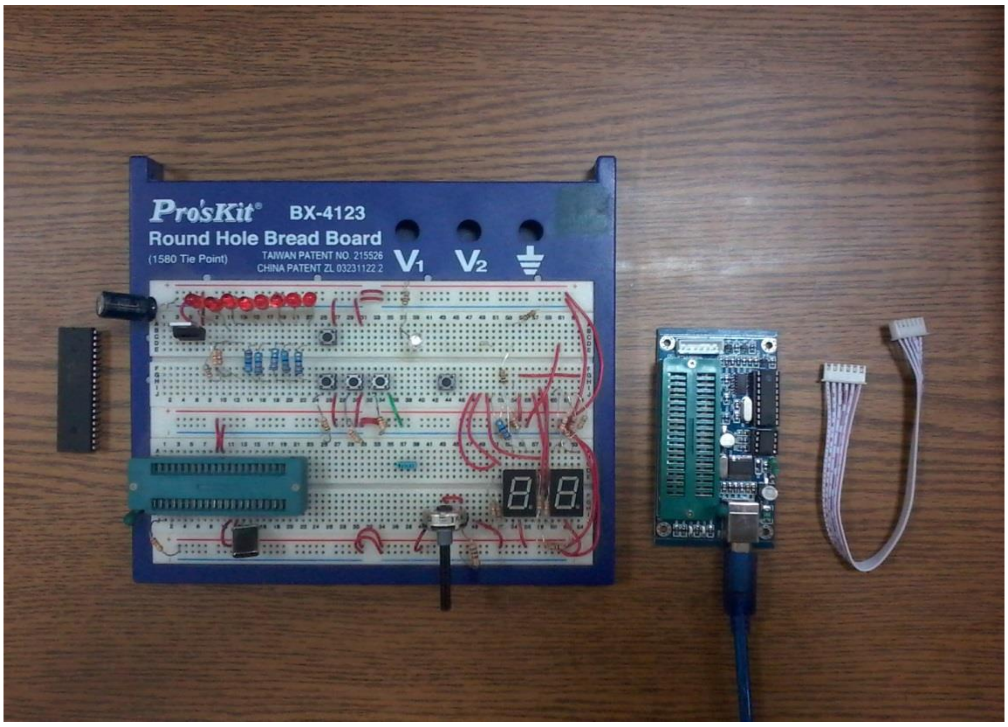


Figure 3.1: Equipment and devices used in the experiment.

Theoretical Background:

One way of make delays between operations is to do a wait or delay code with the required time

Example:

```
WAIT:	MOVLW D’1’
MOVWF TIME3
NEXT3:		MOVLW D'100'
		MOVWF TIME2
NEXT2:		MOVLW D'199'
		MOVWF TIME1
NEXT1:		NOP
		NOP
		DECFSZ TIME1,1
		GOTO NEXT1
		DECFSZ TIME2,1
		GOTO NEXT2
		DECFSZ TIME3,1
		GOTO NEXT3
```

The delay time of the previous code can be calculated to give 100.007 ms.

Procedure:

- Write a code that increasing the data at portb by 1 and if the portb value was odd then it will wait for 1 second and if it is even it will wait 2 seconds to do the next increment.
- Build the required circuit.
- Program the device and make sure the circuit is ready to turn the power on.
- Consult your Lab. Supervisor to turn the power on.

Discussion and Analysis:

Observe the execution of the code in the circuit and make your decision about your program if it matches the application requirements or not.

Experiment 4
Analog to Digital Converter

This experiment focuses on dealing with PIC18F452 device built in analog to digital converter.

Objectives:

1. To make the student familiar with the PIC18F452 device A/D hardware.
2. To know how to change the different A/D settings to match the application requirements.

Apparatus:

The devices used in this experiment are:

1. Programmer
2. PIC18F452 IC
3. Breadboard and electronic components as shown in Figure 4.1

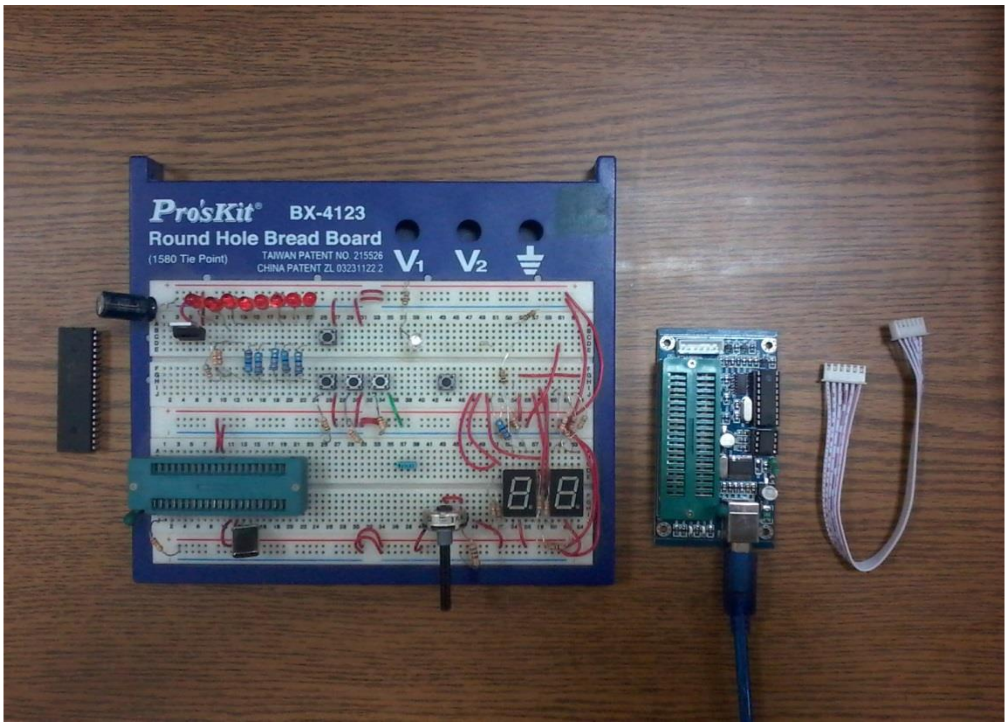
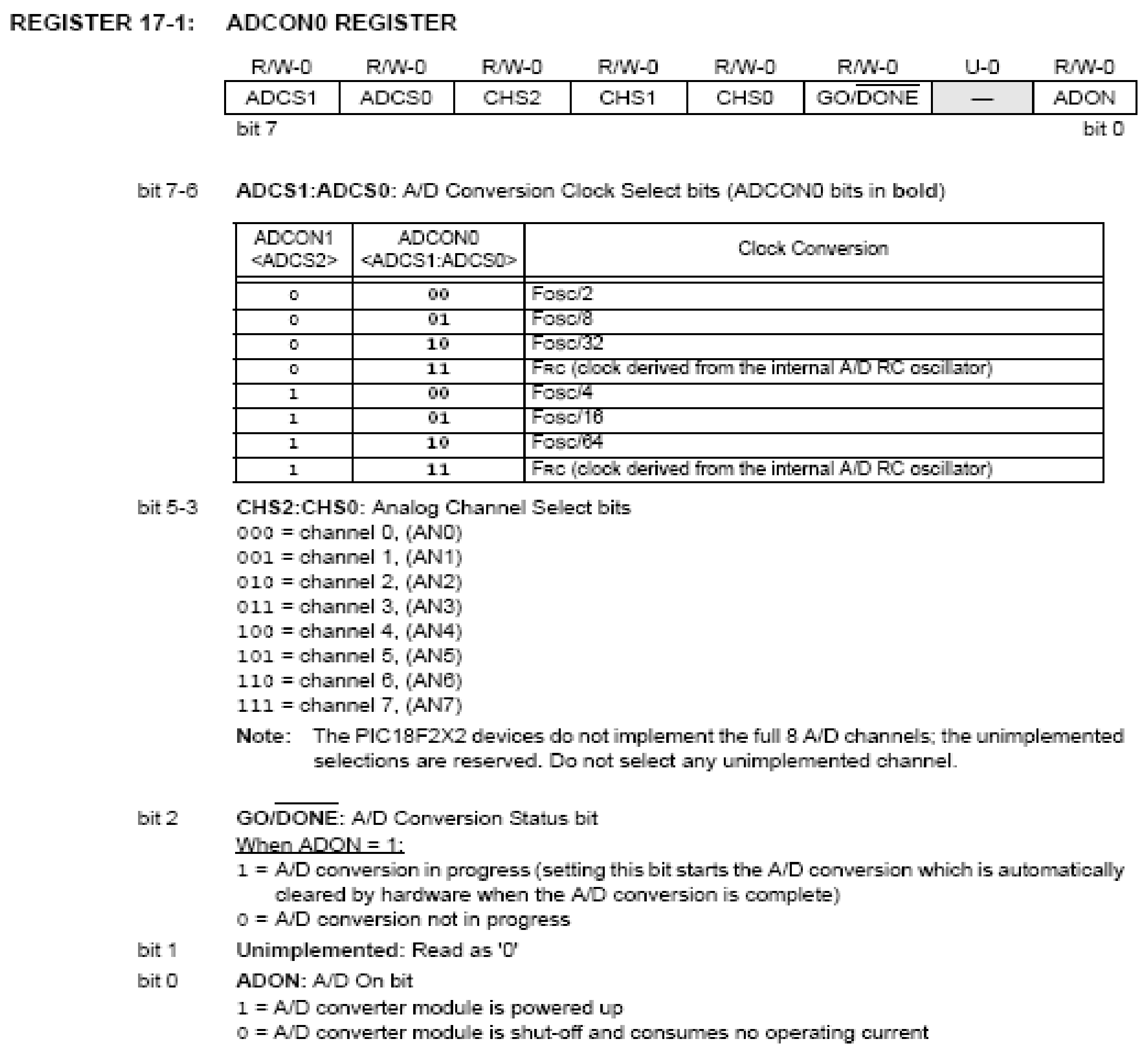


Figure 4.1: Equipment and devices used in the experiment.

Theoretical Background:

A lot of sensors in the real are analog sensors that gives analog voltages, so it is important to know how to read the analog voltages using A/D hardware.

Referring to PIC18F452 datasheet chapter 17:



REGISTER 17-1: ADCON0 REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0**: A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-3 **CHS2:CHS0**: Analog Channel Select bits
000 = channel 0, (AN0)
001 = channel 1, (AN1)
010 = channel 2, (AN2)
011 = channel 3, (AN3)
100 = channel 4, (AN4)
101 = channel 5, (AN5)
110 = channel 6, (AN6)
111 = channel 7, (AN7)

Note: The PIC18F2X2 devices do not implement the full 8 A/D channels; the unimplemented selections are reserved. Do not select any unimplemented channel.

bit 2 **GO/$\overline{\text{DONE}}$**: A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

bit 1 **Unimplemented**: Read as '0'

bit 0 **ADON**: A/D On bit
1 = A/D converter module is powered up
0 = A/D converter module is shut-off and consumes no operating current

REGISTER 17-2: ADCON1 REGISTER

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 6 **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in bold)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	FOSC/2
0	01	FOSC/8
0	10	FOSC/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	FOSC/4
1	01	FOSC/16
1	10	FOSC/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A = Analog input D = Digital I/O
C/R = # of analog input channels / # of A/D voltage references

Procedure:

- Write a code that makes a variable delay time between toggling the value of portb depending on the analog input at AN0
- Program the device and make sure the circuit is ready to turn the power on.
- Consult your Lab. Supervisor to turn the power on.

Discussion and Analysis:

Observe the execution of the code in the circuit and make your decision about your program if it matches the application requirements or not.

Experiment 5
Pulse Width Modulation

This experiment focuses on dealing with PIC18F452 device built in Pulse Width Modulation hardware.

Objectives:

1. To make the student familiar with the PIC18F452 device PWM hardware.
2. To know how to change the different PWM settings to match the application requirements.

Apparatus:

The devices used in this experiment are:

1. Programmer
2. PIC18F452 IC
3. Breadboard and electronic components as shown in Figure 5.1

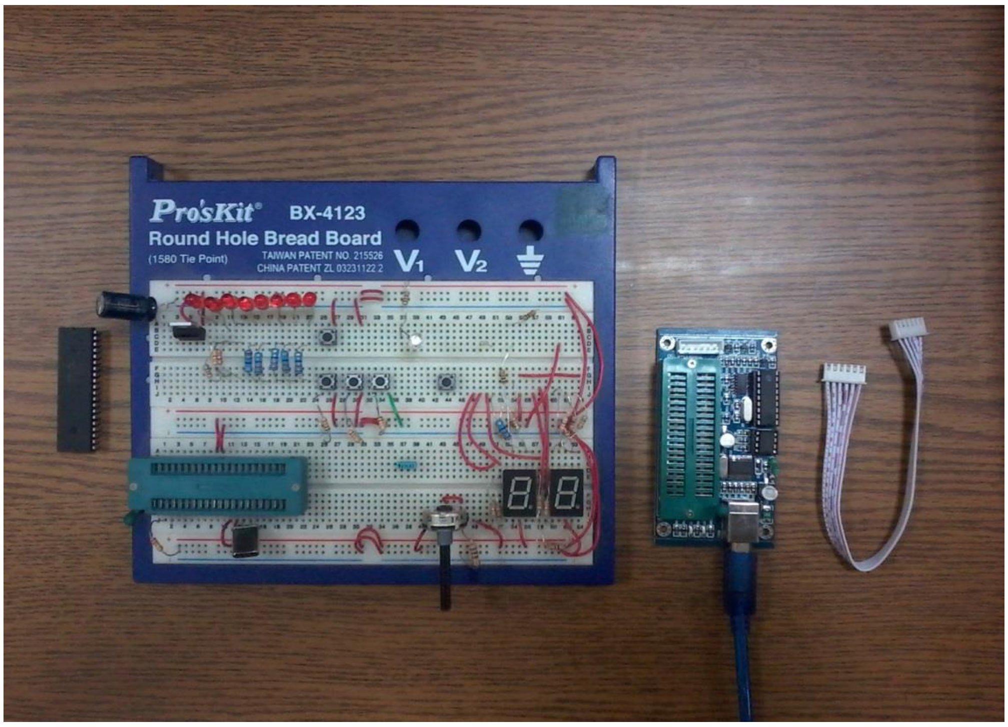
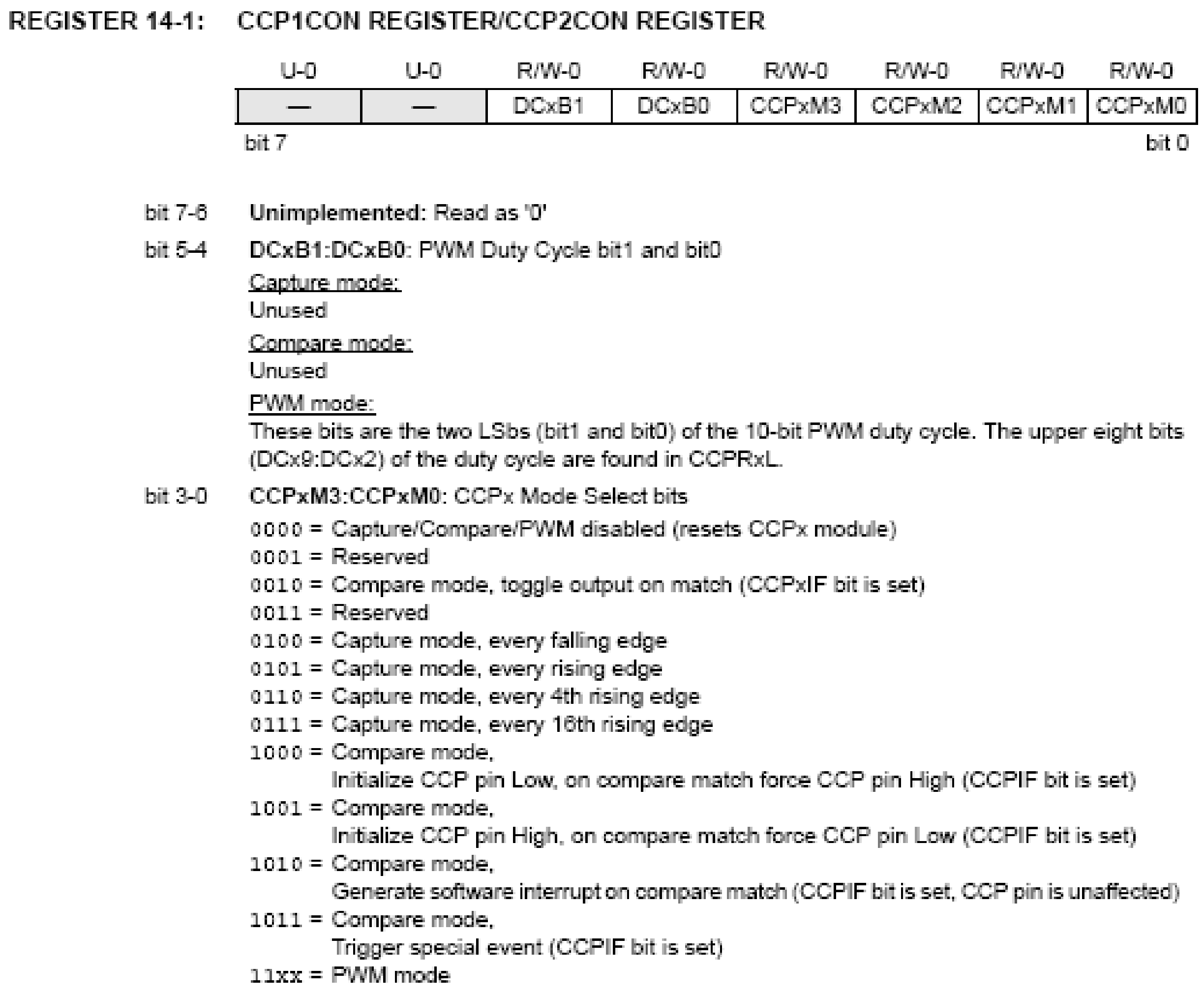


Figure 5.1: Equipment and devices used in the experiment.

Theoretical Background:

PWM is existed in a lot of applications that control some systems through changing the equivalent DC voltage, so it is important to deal with it in microcontrollers.

Referring to PIC18F452 datasheet chapter 14



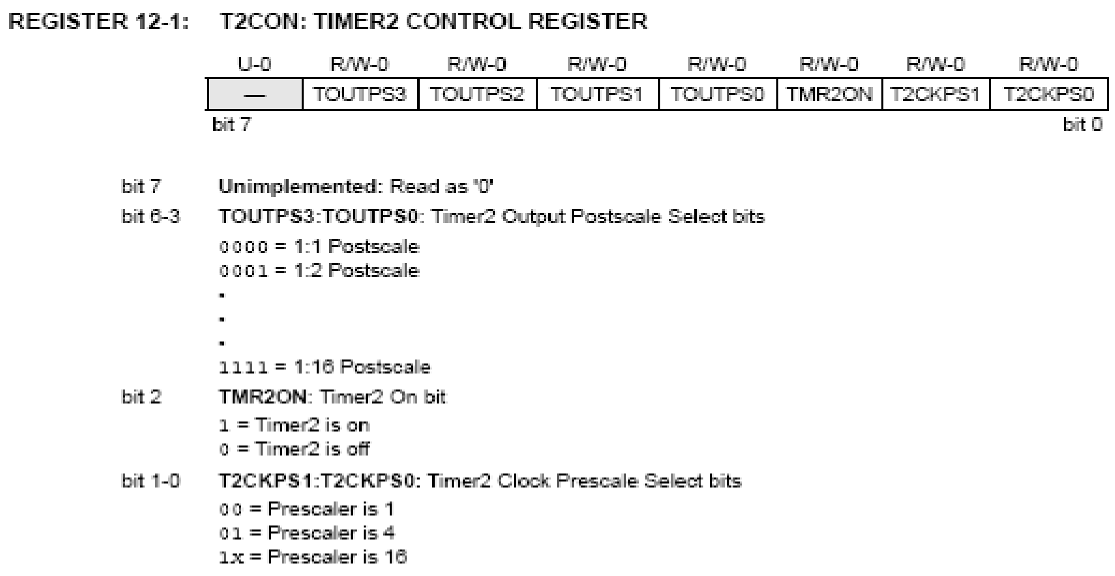
REGISTER 14-1: CCP1CON REGISTER/CCP2CON REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **DCxB1:DCxB0:** PWM Duty Cycle bit1 and bit0
Capture mode:
Unused
Compare mode:
Unused
PWM mode:
These bits are the two LSbs (bit1 and bit0) of the 10-bit PWM duty cycle. The upper eight bits (DCx9:DCx2) of the duty cycle are found in CCPRxL.

bit 3-0 **CCPxM3:CCPxM0:** CCPx Mode Select bits
0000 = Capture/Compare/PWM disabled (resets CCPx module)
0001 = Reserved
0010 = Compare mode, toggle output on match (CCPxIF bit is set)
0011 = Reserved
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode,
Initialize CCP pin Low, on compare match force CCP pin High (CCPIF bit is set)
1001 = Compare mode,
Initialize CCP pin High, on compare match force CCP pin Low (CCPIF bit is set)
1010 = Compare mode,
Generate software interrupt on compare match (CCPIF bit is set, CCP pin is unaffected)
1011 = Compare mode,
Trigger special event (CCPIF bit is set)
11xx = PWM mode



REGISTER 12-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7	**Unimplemented:** Read as '0'
bit 6-3	**TOUTPS3:TOUTPS0:** Timer2 Output Postscale Select bits 0000 = 1:1 Postscale 0001 = 1:2 Postscale • • • 1111 = 1:16 Postscale
bit 2	**TMR2ON:** Timer2 On bit 1 = Timer2 is on 0 = Timer2 is off
bit 1-0	**T2CKPS1:T2CKPS0:** Timer2 Clock Prescale Select bits 00 = Prescaler is 1 01 = Prescaler is 4 1x = Prescaler is 16

Procedure:

- Write a program that changes the duty cycle of the output pulses of RC2 depending on the analog input at AN0.
- Program the device and make sure the circuit is ready to turn the power on.
- Consult your Lab. Supervisor to turn the power on.

Discussion and Analysis:

Observe the execution of the code in the circuit and make your decision about your program if it matches the application requirements or not.

Experiment 6
External Interrupt

This experiment focuses on dealing with PIC18F452 device built in External Interrupt handler.

Objectives:

1. To make the student familiar with the PIC18F452 device Interrupt.
2. To know how to change the different Interrupt settings to match the application requirements.

Apparatus:

The devices used in this experiment are:

1. Programmer
2. PIC18F452 IC
3. Breadboard and electronic components as shown in Figure 6.1

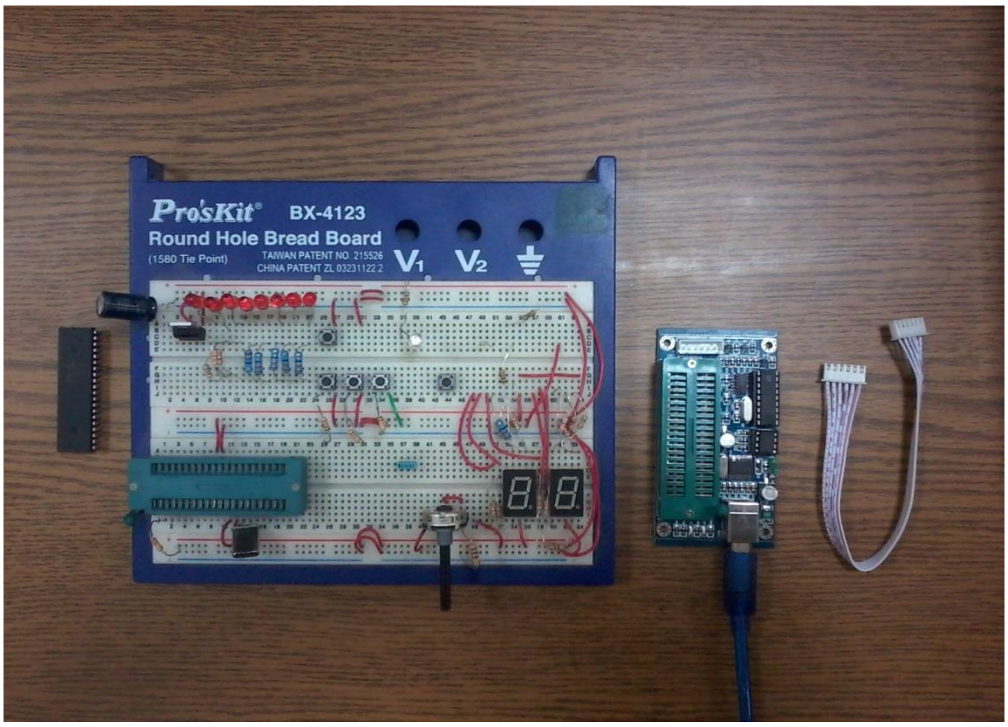
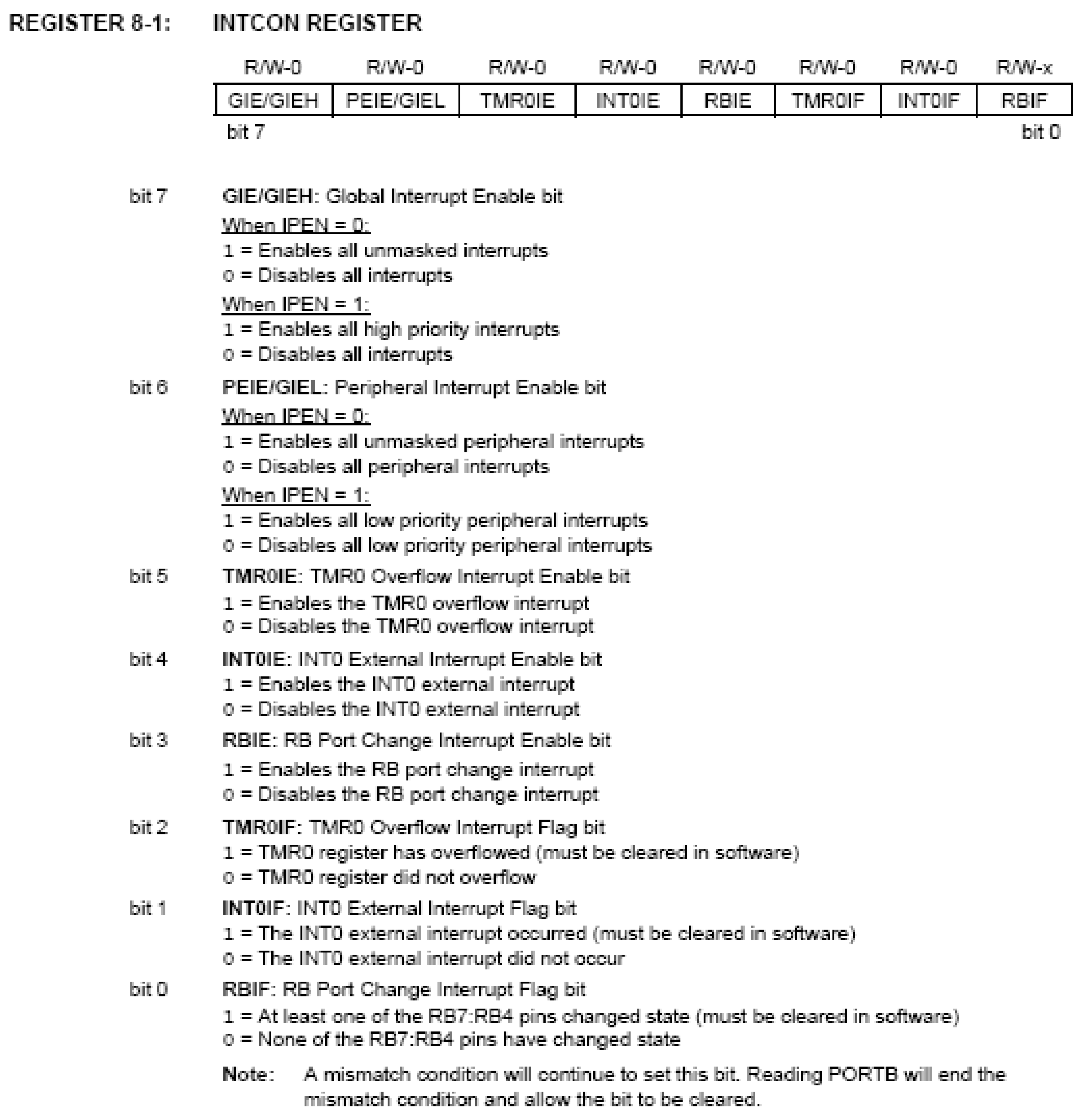


Figure 6.1: Equipment and devices used in the experiment.

Theoretical Background:

Interrupt is very important in microprocessors especially in event driven routines, so it can introduce different methods to deal with hardware.

Referring to PIC18F452 datasheet chapter 8



REGISTER 8-1: INTCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

bit 7 **GIE/GIEH**: Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL**: Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low priority peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE**: INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

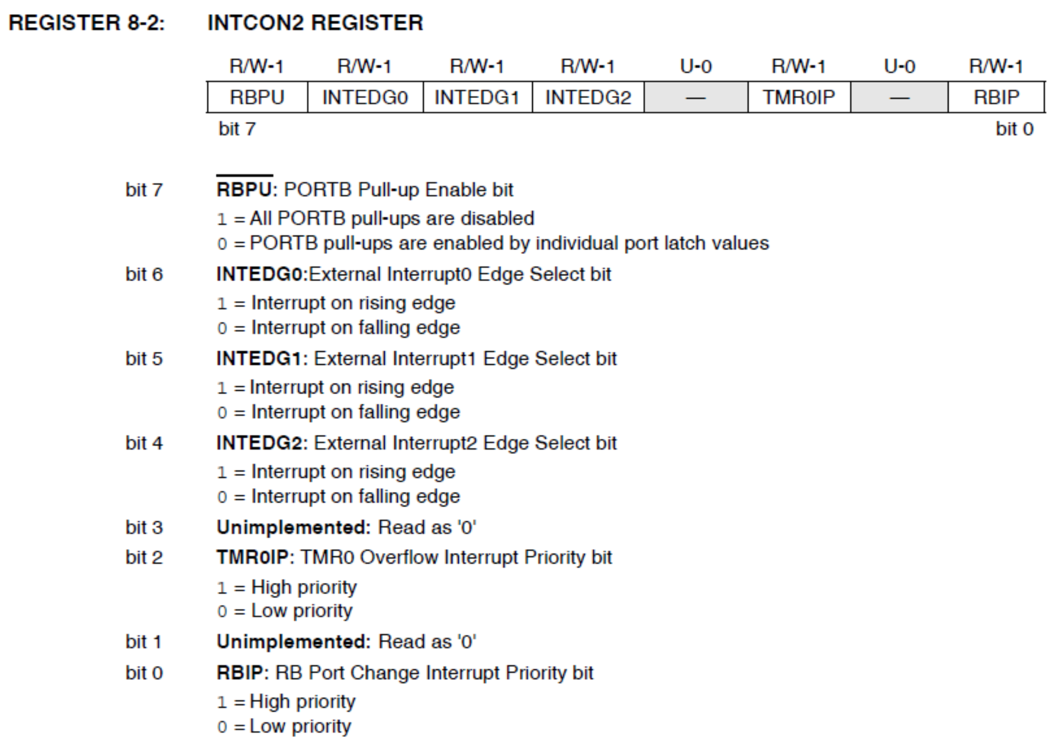
bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF**: INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Note: A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.



REGISTER 8-2: INTCON2 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

bit 7	**$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit 1 = All PORTB pull-ups are disabled 0 = PORTB pull-ups are enabled by individual port latch values
bit 6	**INTEDG0**:External Interrupt0 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 5	**INTEDG1**: External Interrupt1 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 4	**INTEDG2**: External Interrupt2 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 3	**Unimplemented**: Read as '0'
bit 2	**TMR0IP**: TMR0 Overflow Interrupt Priority bit 1 = High priority 0 = Low priority
bit 1	**Unimplemented**: Read as '0'
bit 0	**RBIP**: RB Port Change Interrupt Priority bit 1 = High priority 0 = Low priority

Procedure:

- Write a program that calculates the number of times the button at RB0 was pressed in 3 seconds time and displays it at portc.
- Program the device and make sure the circuit is ready to turn the power on.
- Consult your Lab. Supervisor to turn the power on.

Discussion and Analysis:

Observe the execution of the code in the circuit and make your decision about your program if it matches the application requirements or not.