Palestine Polytechnic University

College of Engineering and Technology
Electrical and Computer Engineering Department

Graduation Project

LabVIEW / PIC Training Environment

Project Team

Ala' Abed Al-Wahid Al-Hashlamoun

Amal Tayseer Wazwaz

Arab Akram Al-Sharif

Project Supervisor
Eng. Elayan Abu Gharbyeh

Hebron – Palestine
May,2007

جامعة بوليتكنك فلسطين

كلية الهندسة والتكنولوجيا

دائرة الهندسة الكهربائية والحاسوب

اسم المشروع

LabVIEW / PIC Training Environment

أسماء الطلبة

آلاء الهشلمون أمل وزوز أرب الشريف

بناءً على نظام كلية الهندسة والتكنولوجيا وإشراف ومتابعة المشرف المباشر على المشروع وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية والحاسوب، وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة الحاسوب.

توقيع المشرف

.....

توقيع اللجنة الممتحنة

....

.....

توقيع رئيس الدائرة

.....

Abstract

Computer-based Learning, sometimes abbreviated CBL, refers to the use of computers as a key component of the educational environment.

This project lies under this huge topic, which is a complete educational environment of PIC (**P**eripheral **I**nterface **C**ontroller) microcontroller that executes several experiments such as traffic light, DC Motor … under the control of a LabVIEW (**L**aboratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) based GUI (**G**raphical **U**ser **I**nterface).

The project's idea aims at developing the currently used system in the Advanced Microprocessor Laboratory in order to make it more efficient to use and understandable by students.

الملخص

يهدف المشروع إلى بناء نظام تعليمي متكامل لدراسة (PIC microcontroller) من خلال تصميم وتنفيذ العديد من التجارب المرفقة بشرح كامل حول آلية عمل كل منها بالإضافة إلى البرنامج المحمل على ال (PIC)، والتوصيلات الكاملة لكل تجربة.

كذلك فقد تم استخدام برنامج (LabVIEW) للتحكم في تنفيذ هذه التجارب من خلال إرسال بيانات معينة خاصة بكل تجربة عن طريق (DAQ) إلى (PIC) حتى تتم الوظيفة المطلوبة.

في النهاية، يستطيع المستخدم رؤية نتائج التجارب على لوحة تحمل ال (PIC) بالإضافة إلى هذه التجارب التي تم بناؤها.

Dedication

In the name of Allah, most Gracious, most Merciful

From the hearts of the children of stones, from the resolution of the martyr who irrigates our chaste land with his blood, our challenge is to do our best to bring out the ripe fruitage, which we water with our efforts, sweat, and sleepless nights.

And now, we reach the time of harvesting to dedicate this progress "LabVIEW/PIC Training Environment" to uprising children, martyrs' mothers, bereaved women, and every martyr who donates his soul for the sake of Allah, and to his land on which we crawled, and on which we were reared and nurtured, to Mother land Palestine.

Anyway, we can't forget our parents and family members who have done their best to make it easy for us to focus on study, and dedication those whom we can't give them their dues in full, whose breath flows with warmth and sympathy, they are the candles that burn to light our life and future.

Ala'

Amal

Arab

Acknowledgement

They said, "Who did not thank people, will not thank God".

We would like to seize this opportunity to extend best wishes and gratitude to well-deserved supervisor Eng. Elayan Abu Gharbyeh, along with respectable Eng. Ala' Jalal, Mr. Rasmi Saed Ahmed, and all who have been so forthcoming and instrumental in helping us reach our goals.

Table of Contents

List of Tables

List of Figures

Chapter One

Introduction

Chapter One
Introduction

1.1 Overview

Innovations in educational technologies have led to various learning environments, and the educational uses of computers are rapidly increasing. Schools and universities are placing more courses on computers to supplement and sometimes replace traditional classroom teaching, in recognition of the potential of computers. They believe that technology can assist in learning, because it provides rich experiences to the learner.

Computer-Based Learning (CBL) is regarded as having great potential for enhancing the quality and effectiveness of education. Cultural differences and attitudes in learning and teaching are an important consideration for tutors engaged choosing and providing of CBL materials.

Computers have been employed in teaching for over thirty years, but their use still plays only a minor role in most undergraduate and postgraduate courses. At present, there are many software packages available for simulation, intelligent tutoring, and mathematical modelling, static and dynamic book emulation and visualization algorithms.

Laboratories, which are found in all engineering and science programs, are an essential part of the education experience. Not only do laboratories demonstrate concepts and ideas, but they also bring the course theory alive so students can see how unexpected events and natural phenomena affect real-world measurements and control algorithms. However, equipping a laboratory is a major expense and its maintenance can be difficult. Teaching assistants are required to set up the laboratory, instruct in the laboratory, and grade laboratory reports. These time-consuming and costly tasks result in relatively low laboratory equipment usage, especially considering that laboratories are available only when equipment and teaching assistants are both available.

Recently many applications are done using different PIC microcontrollers, especially as the process of miniaturization (small size) continued where all of the components needed for a controller were built right onto one chip. Also their importance is increasing by innovation of technologies.

1.2 General Description of the Project

This project creates a complete PIC microcontroller training environment in which the student can choose any implemented experiment to run.

The experiments are viewed in LabVIEW with complete information about each of them such as description, schematic and code also the user can run the required experiment from LabVIEW.

The PIC microcontroller is programmed once to perform all these experiments; so there is no need to program it separately for each of them.

Finally, the user will see the results of his experiments on the implemented hardware for each of them on a single board that includes the PIC microcontroller.

1.3 Project Importance

Support **P**alestine **P**olytechnic **U**niversity (PPU) with a complete system that will be applied in the Advanced Microprocessor Laboratory as a training PIC microcontroller environment.

This system is an alternative training system to the currently used one, which is too complex to be learnt and understood by students.

The previous system based on a low level language (Assembly language), also it uses **C**ommand **L**ine **P**rompt (CMD) as an interface, whereas the new system based on a high level language (C language), also it uses GUI to interact with students using LabVIEW.

1.4 Human Development Resources

The team of this project is composed entirely of three **E**lectrical and **C**omputer **E**ngineer (ECE) undergraduate students who are interested in PIC microcontrollers.

The project team with their supervisor will develop this system

Supervisor: Eng. Elayan Abu-Gharbyeh.
The project Team:

Ala' Al-Hashlamoun.
Amal Wazwaz.
Arab Al-Sharif.

1.5 Literature Review

- Computer Based Learning With LabVIEW/Summer,PPU,2005

  - Overview

    This project aims to be used as an alternative to design and implement experiments in electrical circuit lab in PPU, using new technology to display data and generate signals through DAQ.

- Main differences

    - This project depends on LabVIEW without using any external hardware.
    - It concentrates on circuit lab experiments.
    - All experiments were implemented in LabVIEW.
    - But, our project depends on the PIC microcontroller as a major part, and LabVIEW is used as supporting environment for connectivity between the PIC and user interface.

- Virtual Lab Using LabVIEW/ Summer,PPU,2006

    - Overview

        This project aims to build a remote or virtual laboratory of several experiments that can executed from LabVIEW and monitored from the internet using the web browser.

    - Main differences

        - This project mainly depends on LabVIEW to implement experiments.
        - Simple hardware is implemented.
        - Connecting LabVIEW to the internet is the major purpose.

- But, our project uses LabVIEW as auxiliary software to simplify the use of the PIC microcontroller environment.

- Other projects' ideas were available in **N**ational **I**nstrument (NI) discussion forums, but no one reached to implementation level, so they still under studying.

1.6 Project Estimated Cost

This section lists the overall cost -the rental cost of the components used- that is considered in implementing the system. The costs are divided into: software costs and hardware costs.

Software Estimated Cost

Table 1.1: Software Cost

Component	**Estimated Cost ($)**
Windows XP	400
LabVIEW	1500
	Total: 1900

Hardware Estimated Cost

Table 1.2: Hardware Cost

Component	Estimated Cost ($)
PC (3000GHz CPU, 256MB RAM, 60GB HD)	1500
PIC programmer	160
PIC microcontroller	17
DAQ	550
DAQ connector Block	100
Power supply	100
LCD Display	10
DC Motor	10
H-Bridge	5
Speaker	7
Resistors, LEDs, diodes, transistors	25
Switches	5
Seven Segment	10
Board	10
Wires	15
	Total: 2524

1.7 Time Schedule

Time planning illustrates how the time will be managed to reach the goal of the project. This time management aims to clarify each step of the project and guarantees the full completion of the overall system.

Time schedule includes two time estimation schedule tables, the first one show what is done in the first semester, and the second shows the expected time scheduling of the second semester.

1.7.1 First Semester Time Planning

Week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Data gathering & analyzing																
Requirement analysis																
Studying PIC18F4520																
Studying LABVIEW																
Studying MPLAP																
Project Design																
Documentation																

Figure 1.1: First Semester Time Planning

1.7.2 Second Semester Time Planning

Week / Task	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Implementation																
System Testing																
Documentation																

Figure 1.2: Second Semester Time Planning

1.8 Risk Management

Many types of risks are taken into account in order to be ready to take recovery steps immediately as soon as they appear.

1.8.1 Hardware Risk

The most important hardware part in our project is the microcontroller. The predicted risks are:

- Device failure: the microcontroller may crash because of high voltage supply or other problems.
- The device operates differently than intended.
- Some hardware parts may not be delivered.

1.8.2 Software Risk

There are two important software components to be used in our system. These are C language, and PIC Assembly. There may be some risks with the software such as:

- Incorrect installation of C programming language.
- Trouble in downloading written C programs on the microcontroller.
- Failure in executing C programs on the system.
- Appearance of bugs in the program, which may lead to improper execution.

1.8.3 Group Risk

- Illness of one or more of group members.
- Group meeting difficulties

1.8.4 Requirement Risks

- Several requirements may be affected by other requirements due to the high cost of some components.
- Team fails to understand impact of requirements changes.

Recovery

- Cover budget of extra cost by ignoring unexpected requirements as possible.

1.8.5 Project Risks

- Some requirements and changes may arise lately.
- Schedule not accurate
- Budget not sufficient.
- Latency of devices arrival.

Recovery

- Demand device at earlier time.
- Start working on the implementation earlier.
- Use alternate devices with the same functionality and less cost.
- Try to build our own circuits that can replace unavailable devices.

1.9 Report Contents

The documentation for this project is categorized into seven chapters. Each chapter concerns with a logical or physical part of the system. The followings summarize briefly what each chapter will explain:-

- **Chapter One: Introduction**

This chapter demonstrates on an overview about the system, system design options, a literature review, estimated cost and time planning.

- **Chapter Two: Theoretical Background**

This chapter focuses on theories and materials that are related to our system operation and behaviour.

- **Chapter Three: Design Concepts**

This chapter describes the system in its abstract formula. It describes the project objectives, a general block diagram and how the system works.

- **Chapter Four: Hardware System Design**

This chapter discusses system design options and justifies those that are chosen in the project. A detailed description about the different project parts also is included.

- **Chapter Five: Software System Design**

This chapter handles the software related to our system, depicts flow charts about system operation and illustrates different algorithms and techniques that will be considered in writing the software.

- **Chapter Six: System Testing**

This chapter will manifest the implementation procedures to be acted so as to integrate the project. Then, a sequence of procedural testing will be listed. The testing comprises both software and hardware testing.

- **Chapter Seven: Conclusions and Future Work**

This chapter will list the problems faced us in accomplishing the system and how did they resolved. Notes and Conclusions will help readers are also included. A future work is also proposed.

Chapter Two

Theoretical Background

Chapter Two

Theoretical Background

2.1 Introduction

This chapter contains theoretical information related to the hardware and software components that will be used in the project.

2.2 Software Components

2.2.1 LabVIEW

LabVIEW is a full-featured graphical programming language with all the standard features of the general-purpose programming environment such as data structure, looping structures, and event handling, it is also has a built-in compiler that compiles code at edit time.

However, unlike other general-purpose programming languages, LabVIEW specifically designed for engineers and scientists and has built-in tools to meet their

needs. These high-level functions, assistants, and tools make LabVIEW much more than a programming language.

NI LabVIEW is a graphical development environment for creating flexible and scalable design, control, and test applications rapidly and at minimal cost. With NI LabVIEW, engineers and scientists interface with real-world signals, analyze data for meaningful information, and share results through intuitive displays, reports, and the web. Regardless of experience, LabVIEW makes development fast and easy for all users.

NI LabVIEW delivers a powerful graphical development environment for signal acquisition, measurement analysis, and data presentation, given the flexibility of a programming language without the complexity of traditional development tools, and it also uses dataflow programming, where the flow of data through the nodes on the block diagram determines the execution order of the Virtual Instruments (VI) - VI are LabVIEW programs that imitate physical instruments- and functions. [1]

2.2.1.1 LabVIEW Functions

LabVIEW can perform the following functions:

- **Acquiring**

NI LabVIEW is an open environment designed to make interfacing with any measurement hardware simple. With interactive assistants, code

generation, and connectivity to thousands of devices, LabVIEW makes gathering data as simple as possible. [2]

- **Analyzing:**

LabVIEW has more than 500 built-in functions designed specifically for extracting useful information from any set of acquired data and for analyzing measurements and processing signals.

- **Presenting:**

LabVIEW provides tools for data visualization, user interface design, Web publishing, report generation, data management, and software connectivity.

2.2.1.2 VI

VI is a program in LabVIEW that models the appearance and function of a physical instrument is combined from three major components:

1. Front Panel: acts as a user interface.
2. Block Diagram: contains the graphical source code that defines the functionality of the VI.
3. Icon and Connector pane: identifies the VI.

1 Front Panel

Front Panel is the user interface of the VI, it contains two components s and indicators, which are the interactive input and output terminals of the VI ively.

Controls such as knobs, push buttons, dials, and other input devices, they ate instrument input devices and supply data to the block diagram of the VI.

Indicators such as graphs, LEDs, and other displays, they simulate instrument t devices.

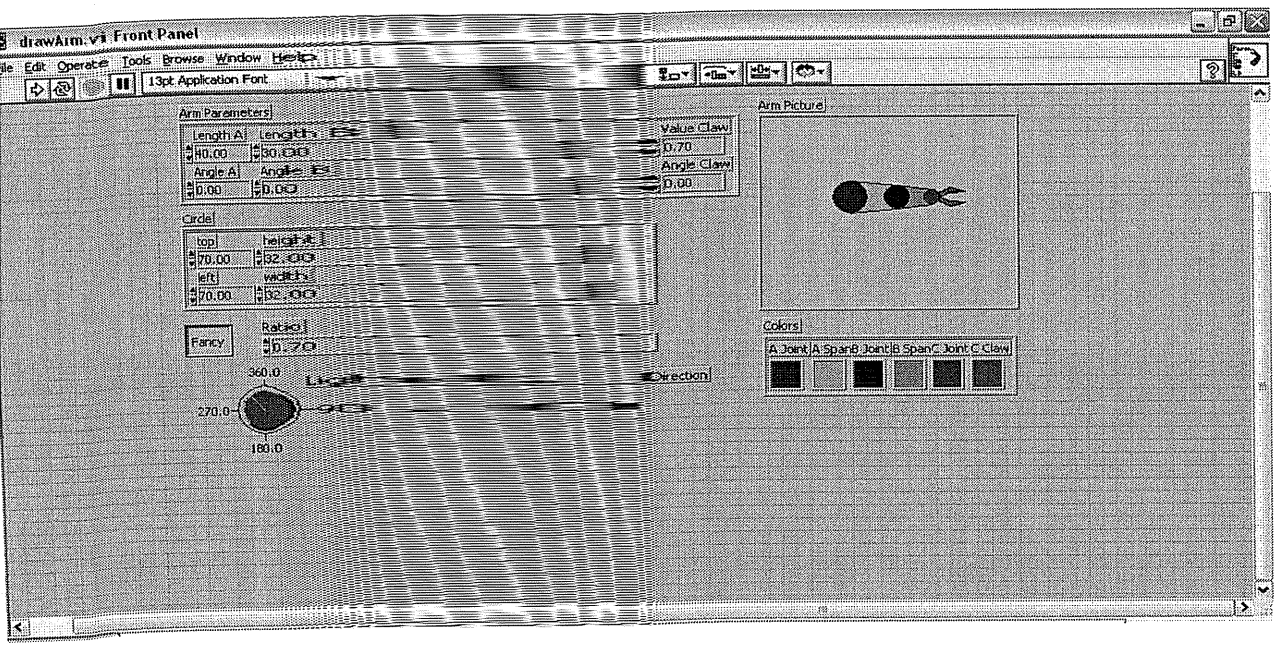


Figure 2.1: Front Panel Window

2.2.1.2.2 Block Diagram

Block Diagram is a window holds the graphical source code of the LabVIEW where Front Panel objects appears as terminals on the block diagram. Also, it contains functions and structures from built-in LabVIEW VI libraries. Wires connect each of the nodes on the block diagram, including control and indicator terminals, functions and structures. [3]

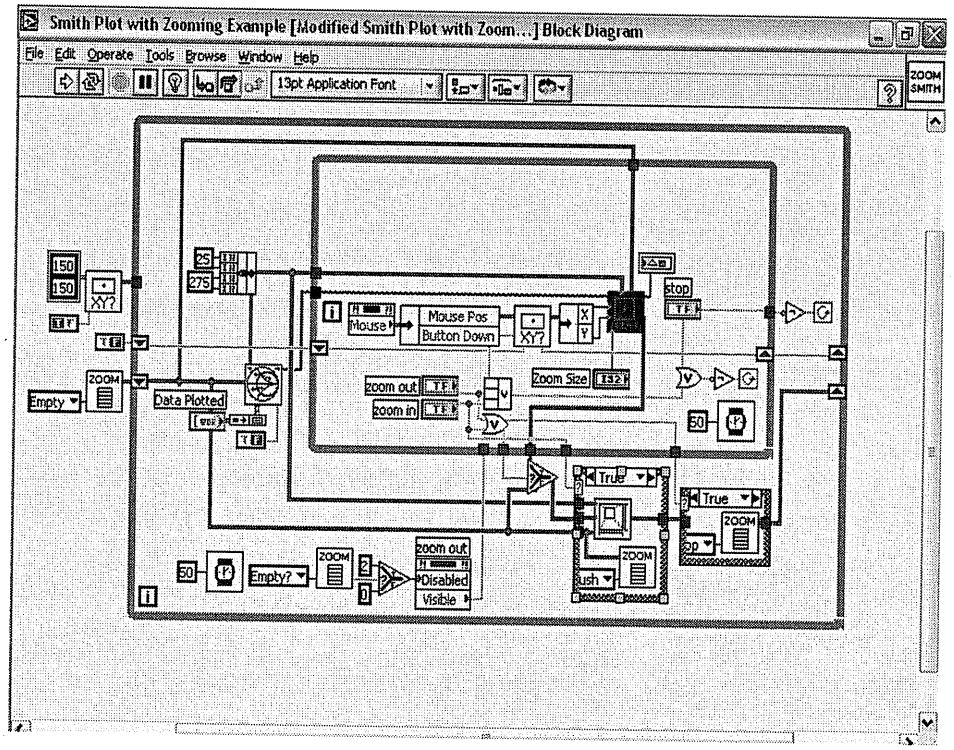


Figure 2.2: Block Diagram Window

2.2.1.3 Why using LabVIEW

LabVIEW is easy to use, interactive, so we can write sophisticated programs and applications in a short time.

In addition, LabVIEW presents the following benefits:

- LabVIEW provides open connectivity with other programs such as MATLAB, Excel …etc.
- LabVIEW is easy to learn, use and gives a great results.
- LabVIEW is widely used in industry, employable skills for students to learn.

2.3 Hardware Components

2.3.1 PIC18F4520 microcontroller

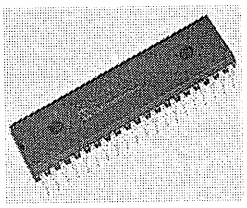


Figure 2.3: PIC18F4520 Microcontroller

2.3.1.1 PIC 18F4520 Features

- **General features**
  - DC - 40 MHz external Operating Frequency.
  - 32 K Program Memories (Bytes).
  - 16384 Program Memories (Instructions).
  - 1536 Data Memory (Bytes).
  - 256 Data EEPROM Memory (Bytes).
  - 20 Interrupt Sources.
  - A, B, C, D, E I/O Ports.
  - 4 Timers.

- 10-bit Analog-to-Digital Module.
- Programmable Low Voltage Detect.
- Programmable Brown-out Reset.
- 75 Instruction Set.
- Internal clock 8 MHz
- 40-pin DIP

- **Pin Layout**

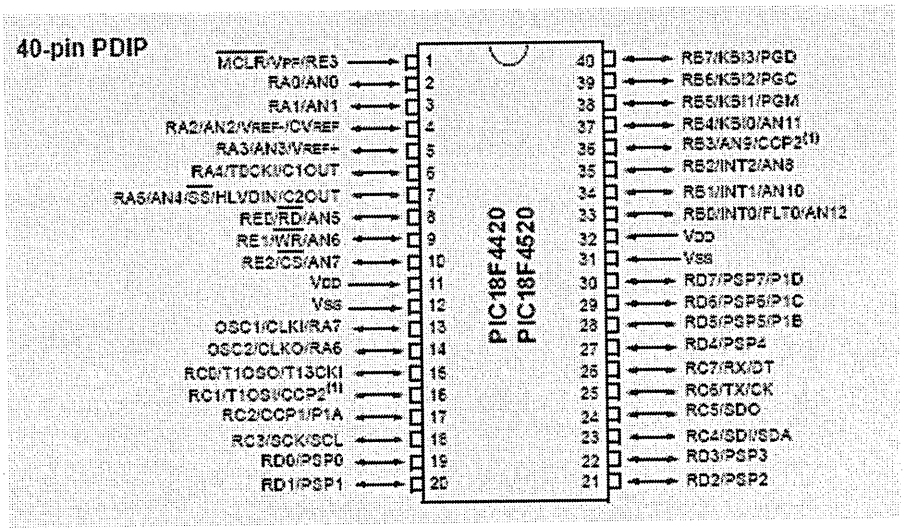


Figure 2.4: PIC18F4520 Pin Layout

- **CPU**

  - Up to 10 MIPS performance
  - C compiler optimized RISC architecture
  - 8 x 8 Single Cycle Hardware Multiply

- **System**

  - Internal oscillator support - 31 kHz to 8 MHz, up to 32 MHz with 4X PLL
  - Fail-Safe Clock Monitor – allows safe shutdown if clock fails
  - Watchdog Timer with separate RC oscillator
  - Wide operating Voltage range; 2.0V to 5.5V

- **Analog Features**

  - 10-bit Analog to Digital Converter(ADC), 13 channels, 100K samples per second
  - Programmable Low Voltage Detection Module
  - Programmable Brown-out Reset module
  - Two Analog comparators with input multiplexing

- **Special Microcontroller Features**

  - C compiler optimized architecture:
    - Optional extended instruction set designed to optimize re-entrant code
  - 100,000 erase/write cycle Enhanced Flash program memory typical
  - 1,000,000 erase/write cycle Data EEPROM memory typical
  - Flash/Data EEPROM Retention: 100 years typical
  - Self-programmable under software control

- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
    - Programmable period from 4 ms to 131s
- Single-supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Wide operating voltage range: 2.0V to 5.5V
- Programmable 16-level High/Low-Voltage Detection (HLVD) module:
    - Supports interrupt on High/Low-Voltage Detection
- Programmable Brown-out Reset (BOR
- With software enable option

2.3.1.2 PORTA

PORTA is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISA.

The RA4 pin is multiplexed with the Timer0 module clock input and one of the comparator outputs to become the RA4/T0CKI/C1OUT pin. Pins RA6 and RA7 are multiplexed with the main oscillator pins; they are enabled as oscillator or I/O pins by the selection of the main oscillator in the configuration register. When they are not used as port pins, RA6 and RA7 and their associated TRIS and LAT bits are read as '0'.

The other PORTA pins are multiplexed with analog inputs, the analog VREF+ and VREF- inputs and the comparator voltage reference output. The operation of pins RA3:RA0 and RA5 as A/D converter inputs is selected by clearing or setting the control bits in the ADCON1 register.

2.3.1.3 PORTB

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB.

Four of the PORTB pins (RB7:RB4) have an interruption-change feature. Only pins configured as inputs can cause this interrupt to occur. The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are ORed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>), this interrupt can wake the device from the Sleep mode, or any of the Idle modes.

2.3.1.4 PORTC

PORTC is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISC.

RC1 is normally configured by configuration bit, CCP2MX, as the default peripheral pin of the CCP2 module (default/erased state, CCP2MX = 1).

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for additional information.

The contents of the TRISC register are affected by peripheral overrides. Reading TRISC always returns the current contents, even though a peripheral device may be overriding one or more of the pins.

2.3.1.5 PORTD

PORTD is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISD.

All pins on PORTD are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. Three of the PORTD pins are multiplexed with outputs P1B, P1C and P1D of the enhanced CCP module.

2.3.1.6 PORTE

PORTE is a 4-bit wide port. Three pins (RE0/RD/AN5, RE1/WR/AN6 and RE2/CS/AN7) are individually configurable as inputs or outputs.

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

The upper four bits of the TRISE register also control the operation of the Parallel Slave Port.

The fourth pin of PORTE (MCLR/VPP/RE3) is an input only pin. Its operation is controlled by the MCLRE configuration bit. When selected as a port pin (MCLRE = 0), it functions as a digital input only pin; as such, it does not have TRIS or LAT bits associated with its operation.

Otherwise, it functions as the device's Master Clear input. In either configuration, RE3 also functions as the programming voltage input during programming.

2.3.1.7 Interrupts

The PIC18F4520 has multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high priority level or a low priority level. The high priority interrupt vector is at 0008h and the low priority interrupt vector is at 0018h. High priority interrupt events will interrupt any low priority interrupts that may be in progress.

There are ten registers which are used to control interrupt operation. These registers are:

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

2.3.1.8 Timers

2.3.1.8.1 Timer0

- **Module**

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

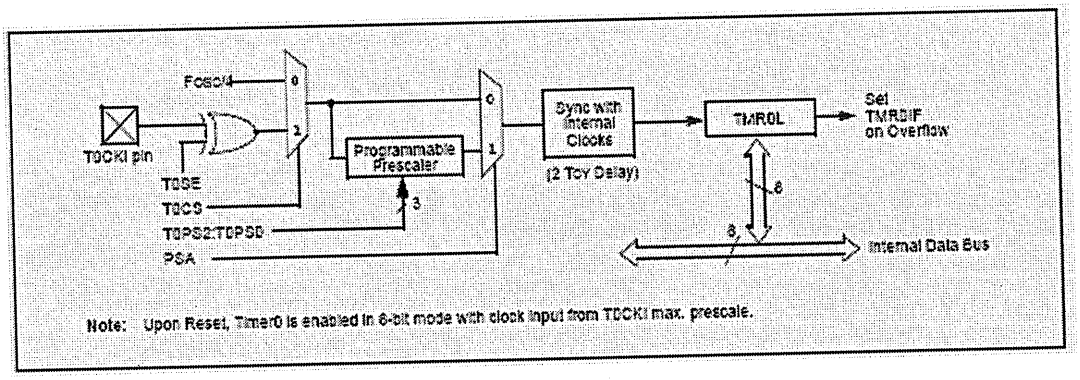


Figure 2.5 : Timer0 Block Diagram (8-bit Mode)

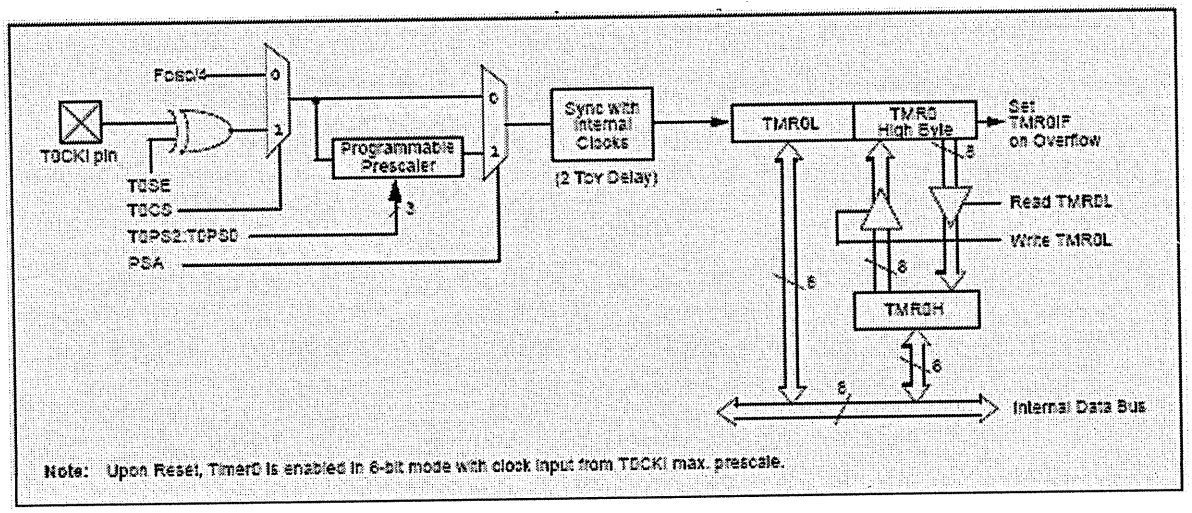


Figure 2.6: Timer0 Block Diagram (16-bit Mode)

- **Operation**

Timer0 can operate as either a timer or a counter; the mode is selected with the T0CS bit (T0CON<5>). In Timer mode (T0CS = 0), the module increments on every clock by default unless a different prescaler value is selected. If the TMR0 register is written to, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

The Counter mode is selected by setting the T0CS bit (= 1). In this mode, Timer0 increments either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (T0CON<4>); clearing this bit selects the rising edge. Restrictions on the external clock input are discussed below.

An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the internal phase clock (TOSC). There is a delay between synchronization and the onset of incrementing the timer/counter.

2.3.1.8.2 Timer1

- **Module**

The Timer1 timer/counter module incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR1H and TMR1L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Reset on CCP Special Event Trigger
- Device clock status flag (T1RUN)

- **Operation**

Timer1 can operate in one of these modes:

- Timer
- Synchronous Counter
- Asynchronous Counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>). When TMR3CS is cleared, Timer1 increments on every internal instruction cycle (Fosc/4), when the bit is set, Timer1 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

When Timer1 is enabled, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

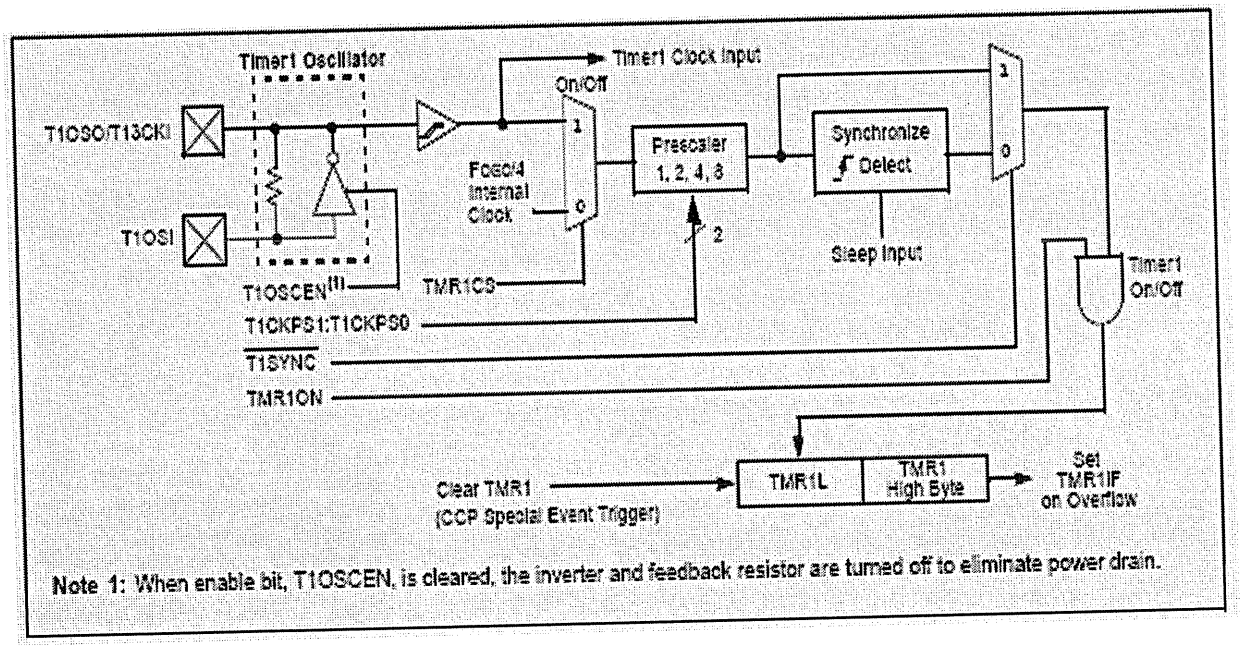


Figure 2.7: Timer1 Block diagram

2.3.1.8.3 Timer2

- **Module**

The Timer2 module timer incorporates the following features:

- 8-bit timer and period registers (TMR2 and PR2,respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4 and 1:16)
- Software programmable postscaler (1:1 through 1:16)
- Interrupt on TMR2-to-PR2 match
- Optional use as the shift clock for the MSSP module the module is controlled through the T2CON register, which enables or disables the timer and configures the prescaler and postscaler.

Timer2 can be shut off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

A simplified block diagram of the module is shown in following figure.

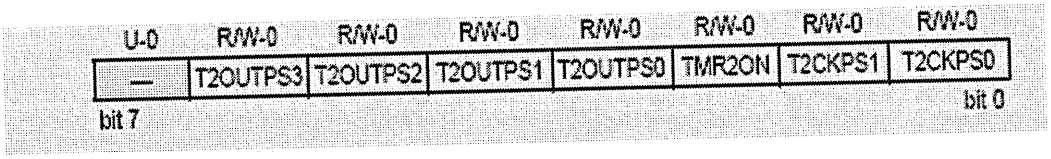


Figure 2.8: T2CON: Timer2 Control Register

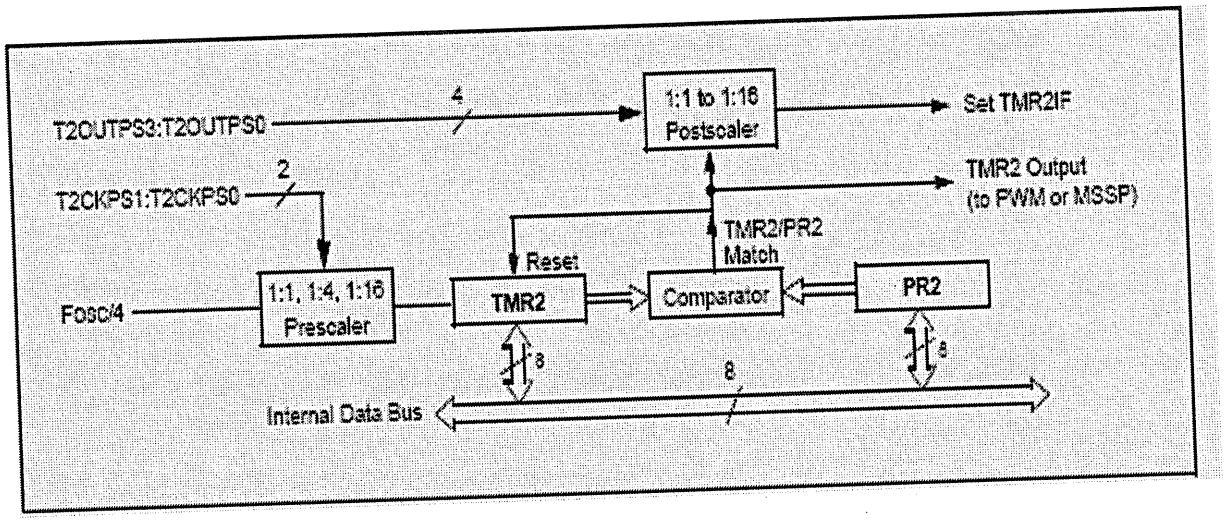


Figure 2.9: Timer2 Block Diagram

- **Operation**

In normal operation, TMR2 is incremented from 00h on each clock (FOSC/4). A 4-bit counter/prescaler on the clock input gives direct input, divide-by-4 and divide-by-16 prescale options; these are selected by the prescaler control bits, T2CKPS1:T2CKPS0 (T2CON<1:0>).

The value of TMR2 is compared to that of the period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/ postscaler.

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, while the PR2 register initializes at FFh.

Both the prescaler and postscaler counters are cleared on the following events:

- a write to the TMR2 register
- a write to the T2CON register
- any device Reset (Power-on Reset, MCLR Reset,

2.3.1.8.4 Timer3

- **Module**

The Timer3 module timer/counter incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR3H and TMR3L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Module Reset on CCP Special Event Trigger

The Timer3 module is controlled through the T3CON register. It also selects the clock source options for the CCP modules.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

Figure 2.10: T3CON: Timer3 Control Register

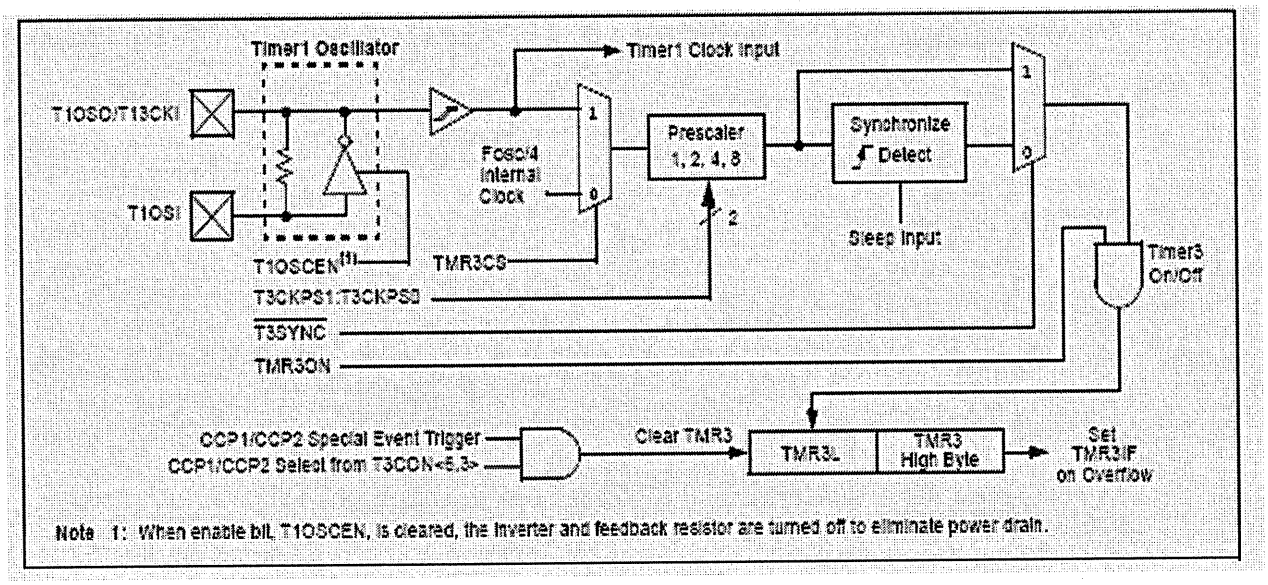


Figure 2.11: Timer3 Block Diagram

- **Operation**

Timer3 can operate in one of three modes:

- Timer
- Synchronous Counter
- Asynchronous Counter

The operating mode is determined by the clock select bit, TMR3CS (T3CON<1>). When TMR3CS is cleared (= 0), Timer3 increments on every internal instruction cycle (FOSC/4). When the bit is set, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

As with Timer1, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs when the Timer1 oscillator is enabled. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

2.3.1.9 PWM Mode

In Pulse Width Modulation (PWM) mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP2 pin is multiplexed with a PORTB or PORTC data latch, the appropriate TRIS bit must be cleared to make the CCP2 pin an output.

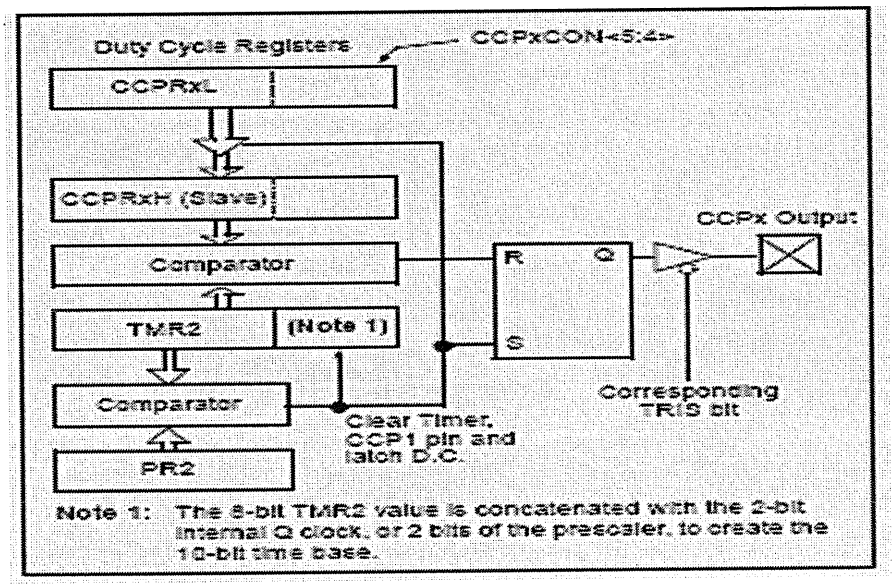


Figure 2.12: Simplified PWM Block Diagram

2.3.1.10 10-Bit Analog-To-Digital Converter (A/D) Module

The Analog-to-Digital (A/D) converter module has 10 inputs for the 28-pin devices and 13 for the 40/44-pin devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

The module has five registers:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

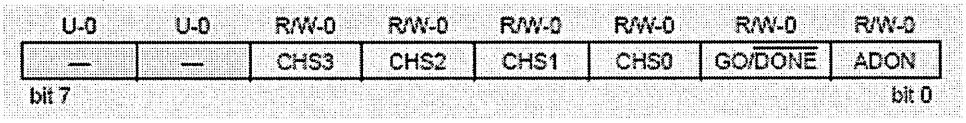


Figure 2.13: ADCON0 Register

The ADCON0 register controls the operation of the A/D module. The ADCON1 register configures the functions of the port pins. The ADCON2 register configures the A/D clock source, programmed acquisition time and justification.

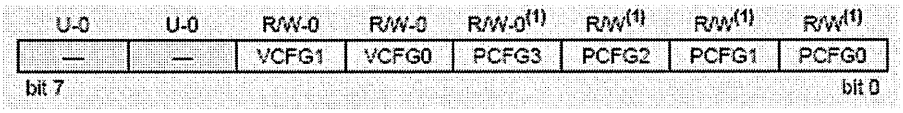


Figure 2.14: ADCON1 Register

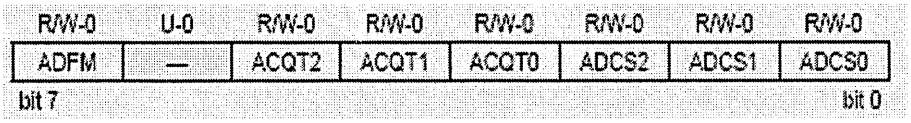


Figure 2.15:ADCON2 Register

The analog reference voltage is software selectable to either the device's positive or negative supply voltage (VDD and VSS), or the voltage level on the RA3/AN3/ VREF+ and RA2/AN2/VREF-/CVREF pins.

The A/D converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

A device Reset forces all registers to their Reset state; this forces the A/D module to be turned off and any conversion in progress is aborted.

Each port pin associated with the A/D converter can be configured as an analog input, or as a digital I/O.

2.3.2 Data Acquisition Card (DAQ NI 6024E)

NI Company produces NI 6024E devices that use Electric (E) Series technology to deliver high performance, reliable data acquisition capabilities.

Figure 2.16: DAQ NI 6024E

This project will use low-cost DAQ Card-6024E.

2.3.2.1 NI 6024E Features

- **General features**
  1- Bus type is PCI.
  2- Supported in some operating systems as Windows, Linux, Real-Time.
  3- Recommended software is LabVIEW.
  4- Driver software is NI-DAQ.

- **Analog input**

  1- 16 channels of 12-bit resolution.
  2- Number of samples is 200 kSample/s.
  3- Maximum voltage range is (-10V to 10V).
  4- Minimum voltage range is (-50mV to 50mV).

- **Analog output**

  1- 2 channels of 12-bit resolution.
  2- Number of samples is 10 kSample/s.
  3- Maximum voltage range is (-10V to 10V).
  4- Minimum voltage range is (-10V to 10V).

2.3.2.1 NI 6024E Features

- **General features**
  1- Bus type is PCI.
  2- Supported in some operating systems as Windows, Linux, Real-Time.
  3- Recommended software is LabVIEW.
  4- Driver software is NI-DAQ.

- **Analog input**

  1- 16 channels of 12-bit resolution.
  2- Number of samples is 200 kSample/s.
  3- Maximum voltage range is (-10V to 10V).
  4- Minimum voltage range is (-50mV to 50mV).

- **Analog output**

  1- 2 channels of 12-bit resolution.
  2- Number of samples is 10 kSample/s.
  3- Maximum voltage range is (-10V to 10V).
  4- Minimum voltage range is (-10V to 10V).

- **Digital input/output**

1- 8 digital I/O lines.
2- Compatible with both TTL and COMS logic levels.
3- Digital trigging.
4- Maximum input range (0V to 5V)
5- Maximum output range (0V to 5V)
6- Input current flow are sinking and sourcing.
7- Output current flow are sinking and sourcing.

- **Counter/Timer**

1- 2 Counter/Timer of 24-bits resolution.
2- Maximum source frequency is 20MHz.
3- Minimum input pulse width is 10ns.
4- Maximum range is (0 to 5V).
5- TTL logic level.

- **Physical specification**

1- Length is 17.5 cm.
2- Width is 10.7 cm.
3- I/O connector is 68 pins male.

- **Driver Software**

NI-DAQ is the robust driver software included with all NI data acquisition and signal conditioning products. This easy-to-use software tightly integrates the full functionality the DAQ hardware to LabVIEW.

2.3.2.2 DAQ NI 6024E Pinout

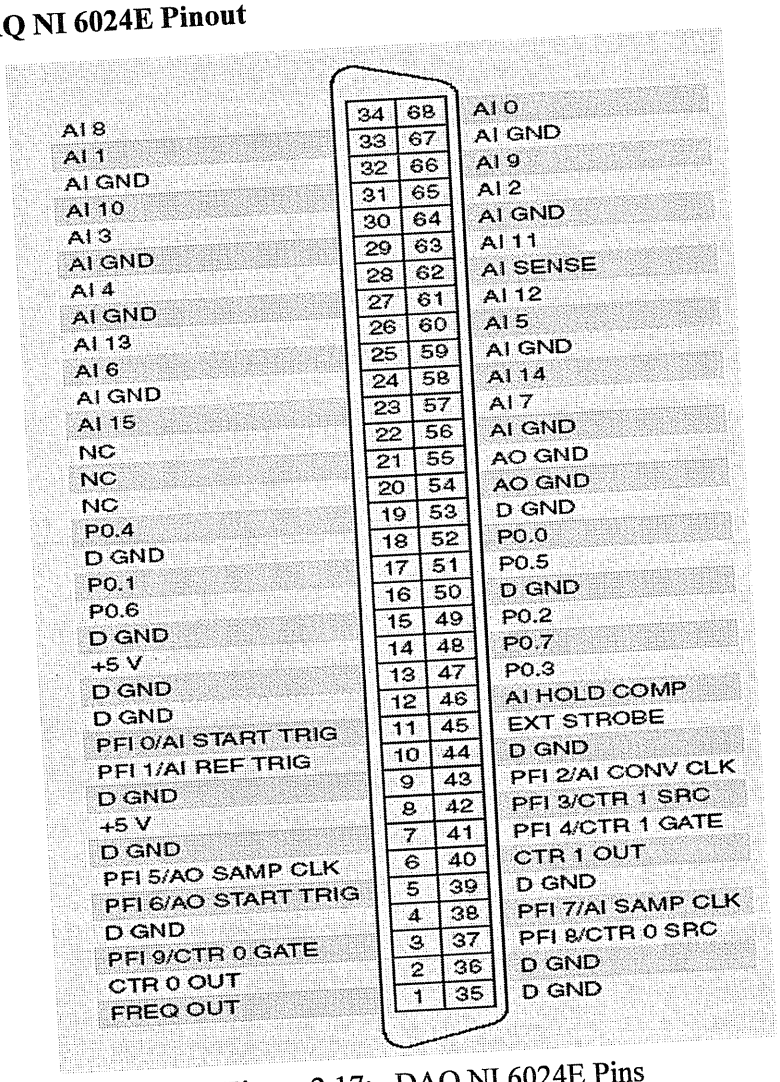


Figure 2.17: DAQ NI 6024E Pins

Each NI DAQCard-6024E requires 1 Cable, 1 Connector Block.

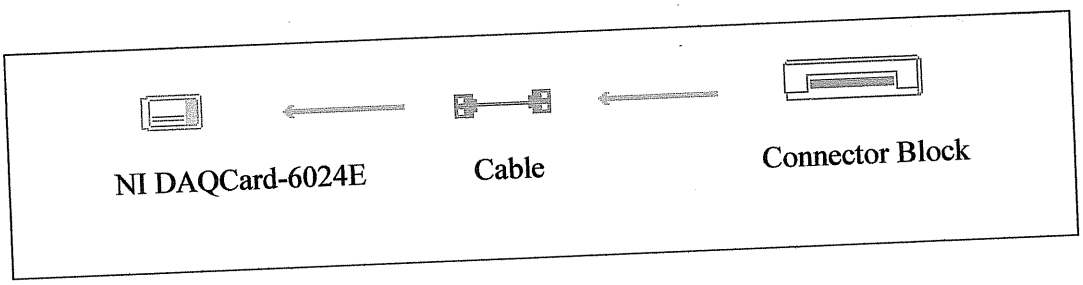


Figure 2.18: DAQ Connection

2.3.3 HD44780 Liquid Crystal Display (LCD) Display

2.3.3.1 HD44780 LCD General Description

One of the most popular components people want to interface with their systems is an LCD display, both to help them debugging their programs and also provide a way to display the results to the outside world.

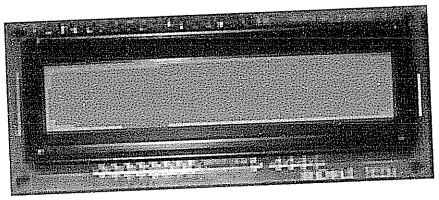


Figure2.19: LCD Display

HD44780 dot-matrix liquid crystal display controller and driver that displays alphanumerics, kana characters and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8-bit microcontroller. Since all the functions such as display **R**andom **A**ccess **M**emory (RAM), character generator, and liquid crystal driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver.

Most text displaying LCD's are based on the LCD controller which takes care of all the multiplexing and the peculiarities required by the LCD display and provides a low level command interface for certain actions like screen clearing, cursor shape and size, character displaying .

The LCD module contains three different functional blocks.

- **C**haracter **G**enerator **R**ead **O**nly **M**emory (CG-ROM).
  The ROM character generator is factory programmed and contains the 208 characters to be displayed.
- **C**haracter **G**enerator **R**andom **A**ccess **M**emory (CG-RAM).
  The CG-RAM allows the user to define up to 8 special characters that are not included in the CG-ROM.
- **D**isplay **D**ata **R**andom **A**ccess **M**emory (DD-RAM).
  The DD-RAM holds the characters that are actually displayed in the HD44780 LCD

The most important of the function blocks is the DD-RAM. This has to do with the way that is implemented and sometimes is different from one LCD to

another depending on how many lines by how many rows the LCD can display the DD-RAM address changes.

2.3.3.2 HD44780 LCD Features

- 5 x7 and 5 x10 dot matrix liquid crystal display controller driver.
- Capable of interfacing to 4-bit or 8-bit MPU.
- DD-RAM: 80 x8 bits (80 characters max).
- Character generator **R**ead **O**nly **M**emory (ROM).
    - Character font 5 x7 dots: 160 characters.
    - Character font 5 x10 dots: 32 characters.
- Both display data and character generator RAM can be read from the MPU using special commands.
- Duty factor selection (selected by program).
    - 1/8 duty: 1 line of 5 x 7 dots + cursor.
    - 1/11 duty: 1 line of 5 x 10 dots + cursor.
    - 1/16 duty: 2 lines of 5 x 7 dots + cursor.
- Maximum number of display characters.
    - 1/8 or 1/11 duty provide 8 characters x 1 line.
    - 1/16 duty provide 8 characters x2 lines.
- Wide range of instruction functions
    - Display clear.
    - Cursor home.
    - Display ON/OF.
    - Cursor ON/OFF.
    - Display character blink.
    - Cursor shift.

- Internal automatic reset circuit at power ON.
- Internal oscillation circuit, also external clock is operation possible.
- Complementary Metal Oxide Semiconductor (CMOS) process
  - Logic power supply: A single + 5V (excluding power for liquid crystal display drive)
  - Operation temperature range: -20 to +75 °C.

2.3.3.3 HD44780 LCD Pins Assignment.

The microcontroller interface to HD44780 LCD modules is through 14 pins as shown in Figure2.21.

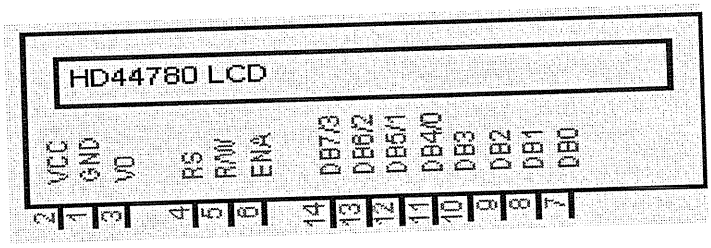


Figure2.20: LCD Pins Assignment

The first three pins provide power to the HD44780 LCD module. Pin 1 is GND and should be grounded to the power supply. Pin 2 is VCC and should be connected to +5V power. Pin 3 is the LCD display bias, By adjusting the voltage or duty cycle of pin 3, the contrast of the display can be adjusted, Most character LCDs can achieve good display contrast with a voltage between 5V and 0V on pin 3.

Greater contrast comes with lower voltage and you should never apply a VLCD higher than VCC.

The second three pins 4, 5, and 6 are the control lines for the HD44780 LCD. These lines indicate what kind of transaction you are proposing to do over the data lines DB0-DB7. The state of Pin4 which is RS indicates whether you wish to transfer commands or display data. Pin5 which is R/W line indicates whether you intend to read or write. Finally, Pin 6 which is E line tells the display when you are actually ready to perform the transaction. The control lines RS, R/W, and E, along with the data lines DB0-DB7 are standard digital logic inputs or outputs.

The remaining pins form 7 to 14 are eight bit bi-directional lines (DB0-DB7) which used to send data.

Table 2.1: HD44780 LCD Pins Assignment

Pin#	Name	Description	In/Out/Power
1	GND	Ground	Power
2	VCC	LCD Controller Power (+3 to +5V)	Power
3	VLCD	LCD Display Bias (+5 to -5V)	Analog
4	RS	Register Select: H: Data L: Command	Input
5	R/W	H: Read L: Write	Input
6	E	Enable (Data strobe, active high)	Input
7	DB0	Data : Least Significant Bit (LSB)	I/O
8	DB1	Data	I/O
9	DB2	Data	I/O
10	DB3	Data	I/O
11	DB4	Data	I/O
12	DB5	Data	I/O
13	DB6	Data	I/O
14	DB7	Data : Most Significant Bit(MSB)	I/O

2.3.3.4 Interfacing HD44780 LCD with Microcontroller.

The interface between microcontroller and HD44780 LCD is either a 4-bit or 8-bit parallel bus that allows fast reading/writing of data to and from the LCD. The waveform will write an American National Standard Code for Information

Interchange (ASCII) Byte out to the LCD's screen. The ASCII code to be displayed is eight bits long and is sent to the LCD either four or eight bits at a time. If 4-bit mode is used, two nibbles of data (First high four bits and then low four bits with an E clock pulse with each nibble) are sent to complete a full eight-bit transfer. The E clock is used to initiate the data transfer within the LCD. 8-bit mode is best used when speed is required in an application and at least ten I/O pins are available. 4-bit mode requires a minimum of six bits. In 4-bit mode, only the top 4 data bits (DB4-7) are used.

LCD's are slow devices when compared to microcontrollers. Special attention must be paid so the commands and data are not send too quickly from the Micro Processing Unit (MPU), and the designer must control the communication speed and timing to ensure that the slow LCD and the fast MPU stay synchronized, for more details see Appendix D.

2.3.4 H-bridge component

An H-bridge is an electronic circuit which enables DC electric motors to be run forwards or backwards, which are available as integrated circuits, the term "H-bridge" is derived from the typical graphical representation of such a circuit as shown in Figure 2.22.

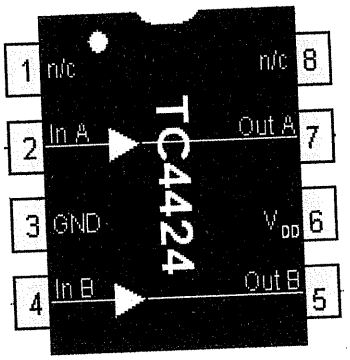


Figure 2.21: H-bridge Chip

An H-bridge is built with four switches (PNP BJTs or P-channel MOSFETs connected to the high voltage bus and NPN BJTs or N-channel MOSFETs connected to the low voltage bus), as shown in Figure 2.23.

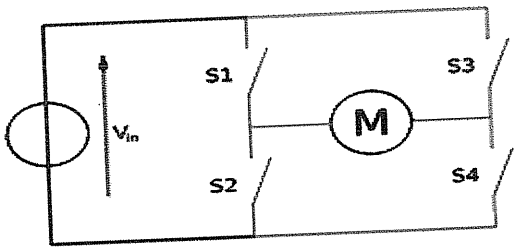


Figure 2.22: Structure of H-bridge

When the switches S1 and S4 (according to the first half in Figure 2.24) are closed (and S2 and S3 are open) a positive voltage will be applied across the motor and move in forward.

By opening S1 and S4 switches and closing S2 and S3 switches (as shown in second half of Figure 2.24), this voltage is reversed, allowing reverse operation of the motor. [7]

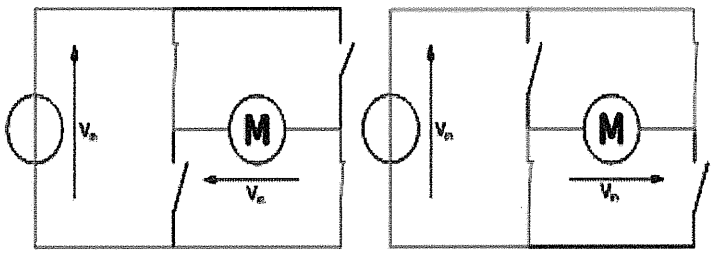


Figure 2.23: Two Basic Operation of H-bridge

There are two logic level compatible inputs in H-bridge chip, A and B, and two outputs, A and B. If input A is brought high, output A goes high and output B goes low. The motor goes in one direction. If input B is driven, the opposite happens and the motor runs in the opposite direction. If both inputs are low, the motor is not driven and can freely, and the circuit consumes no power. If both inputs are brought high, the motor is shorted and braking occurs, as shown in the following Table 2.2.

Table 2.2: Truth Table of H-bridge Input and Output Characteristic

Input		Output	
A	**B**	**A**	**B**
0	0	Float	
1	0	1	0
0	1	0	1
1	1	1	1

Chapter Three

Design Concepts

Chapter Three

Design Concepts

3.1 Introduction

This chapter will describe the objectives of the project, also contains the general block diagram and show in details how the system works.

3.2 Project Objectives

The system is considered to have many objectives, these objectives are:-

1. Create a complete laboratory environment to teach the PIC18F4520 microcontroller features.

2. Simplify using the laboratory environment by implementing a graphical user interface instead of a command line prompt.

3. Give students the ability to choose the required application from the specified list, to run it on the PIC18F4520.

4. Provide students with a complete description about the applications they will run.

3.3 General Block Diagram

The following block diagram defines the general system components:

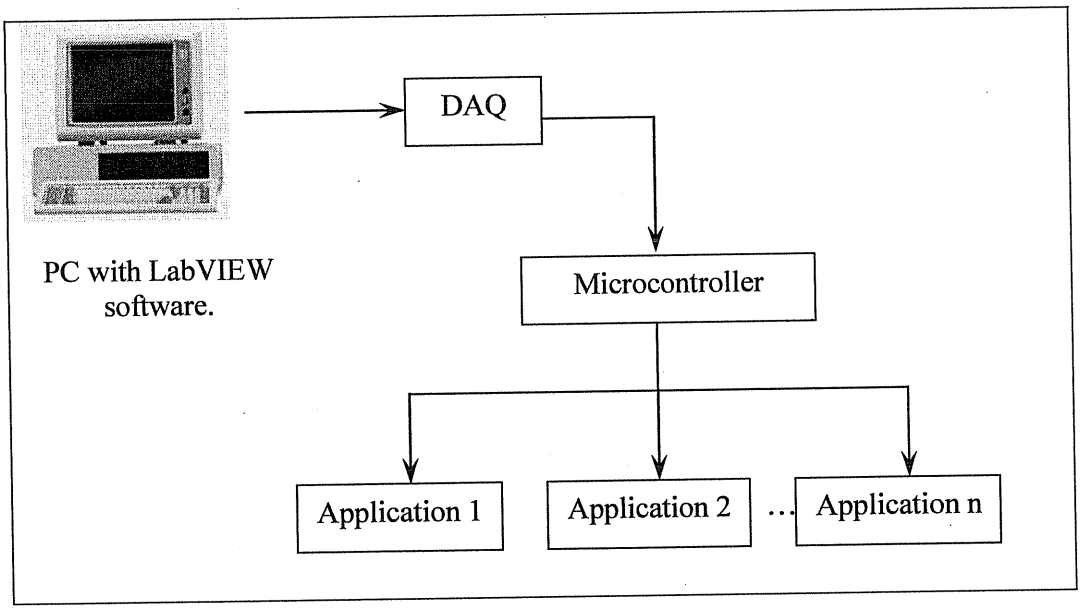


Figure 3.1: General Block Diagram

3.4 How the System Works

This project is a complete educational system that allows the user to choose an application from several choices to run.

These applications are implemented on board to generate an embedded system to the PIC18F4520 microcontroller which is clearly visible to students to increase the interaction with the system.

The user interface is designed in LabVIEW to help the user to use this system in the best way. This is done by providing an efficient introduction about each application and using the user interface components the user will act with the system properly.

Also, the main page in the interface will introduce the project in general, and the other pages will do the same with the applications.

The user will choose a specific application to be executed; internally in the hardware this is done by sending specific values through the data acquisition card (DAQ) to the PIC18F4520 microcontroller that will activate the pins related to that application in order to perform it properly.

These applications are chosen carefully to interact with students requirements to satisfy the objectives of the project, and here we decided to implement the following applications:

- Traffic Light
- Music Tones Generation
- Two-digit Decimal Timer
- LCD Voltmeter
- DC Motor

Chapter Four

Hardware System Design

Chapter Four

Hardware System Design

4.1 Introduction

This chapter describes the main hardware components that are used during the building of the system, and represents block diagrams and general schematic diagrams which represent how these components are integrated to each other.

4.2 Design Options

4.2.1 PIC Microcontroller

Because of the wide using of the PIC microcontroller in these days, this project aims to establish training environment for the PIC microcontroller, and so helps the students to learn its features and how to configure and program it in easier way than the one used now in the Advanced Microprocessor Laboratory of the PPU.

Choosing a specific version of the PIC microcontroller was one of the problems in this project.

There are two possible choices for PIC microcontroller version, the first one is to use the same PIC microcontroller version that currently used in the laboratory, and the second one is to use PIC microcontroller version with advanced features than the currently used one, the two choices discussed below.

- **PIC16F84 Microcontroller**

PIC16F84 which is currently used in the laboratory is a available in the market, and PPU has a programmer that support it, but it has limited features such as number of input and output ports, program memory size, number of timers that can't support the completion of the project. Features of PIC16F84 are listed below.

- **Features**

1- 13 input/output bidirectional pins, only 2 ports A with 5 pins and B with 8 pins.

2- 8-bits timer (only one timer).

3- Data memory consists of EEPROM and RAM memories, 64 bytes of EEPROM memory, and 68 locations in RAM memory.

4- Program memory is 1024 locations with 14 bits width stored in flash memory.

5- There are 4 interrupt sources.

6- 37 instruction set.

7- 28-pin PIDP.

- **PIC18F4520 Microcontroller**

PIC18F4520 is a available in the market, and PPU has a programmer that support it, also since this PIC provide features which are good for our applications and since this project aims to develop the currently used system, the option of choosing the PIC18F4520 is applied, Features of PIC18F4520 are listed below.

- **Features**

1- 36 input/output bidirectional pins, 5 ports (A, B, C, D of 8-bits) but port E of 4-bits.

2- 4 timers, timer0 & timer1 & timer3 of 16-bits, but timer2 of 8-bits.

3- Program memory is 32k stored in flash memory.

4- Data memory consists of EEPROM and SRAM memories,1536 bytes of SRAM memory, 256 bytes of EEPROM memory.

5- 10-bit Analog to digital converter (13-channels).

6- There are 20 interrupt sources.

7- 75 instructions, 83 with extended instruction set enabled.

8- 40-pin PDIP.

9- Capture/Compare/PWM Modules.

10- Enhanced Capture/Compare/PWM Modules.

4.2.2 Interfacing PIC Microcontroller with PC

In the project PIC microcontroller is connected to PC, and to run a specific application on the PIC microcontroller, three bits must be sent from PC to the PIC microcontroller and must exist on three pins of the microcontroller until the PIC check the status of these three pins to determine which application will be executed.

This interface can be done by serial port, parallel port, or DAQ. Each one of theses possibilities is discussed below.

1- Serial Port

Features:

- Slow because it sends data one bit at a time.
- Send data for long distance.
- Need less number of wires.

Drawbacks:

- There is no driver for serial port in LabVIEW.
- Since serial port sends one bit at a time, it is impossible to send three bits to three pins of PIC simultaneously.
- Slow.

2- Parallel Port

Features:

- Sends more than one bit at a time.
- More wires are needed.
- Sends data for short distance, reliable for distances less than 20 feet.
- There isn't driver for parallel port to use them easily in LabVIEW.

Drawbacks:

- There is no driver for parallel port in LabVIEW

3- DAQ

NI produces NI DAQ (Driver software) that helps the user to use DAQ in LabVIEW easily

- **DAQ advantages**

  - Used in high-speed data converter systems
  - High-speed digitizer cards.
  - Are ideal for high-precision, high-speed, power-sensitive applications.
  - Provide sampling rates from 100 MS/s to 8 GS/s
  - Provide feature wide bandwidths and acquisition memories up to 1 Gpoints per channel.

According to the advantages of DAQ the interface will be implemented through it.

4.3 System Block Diagrams

4.3.1 General System Block Diagram

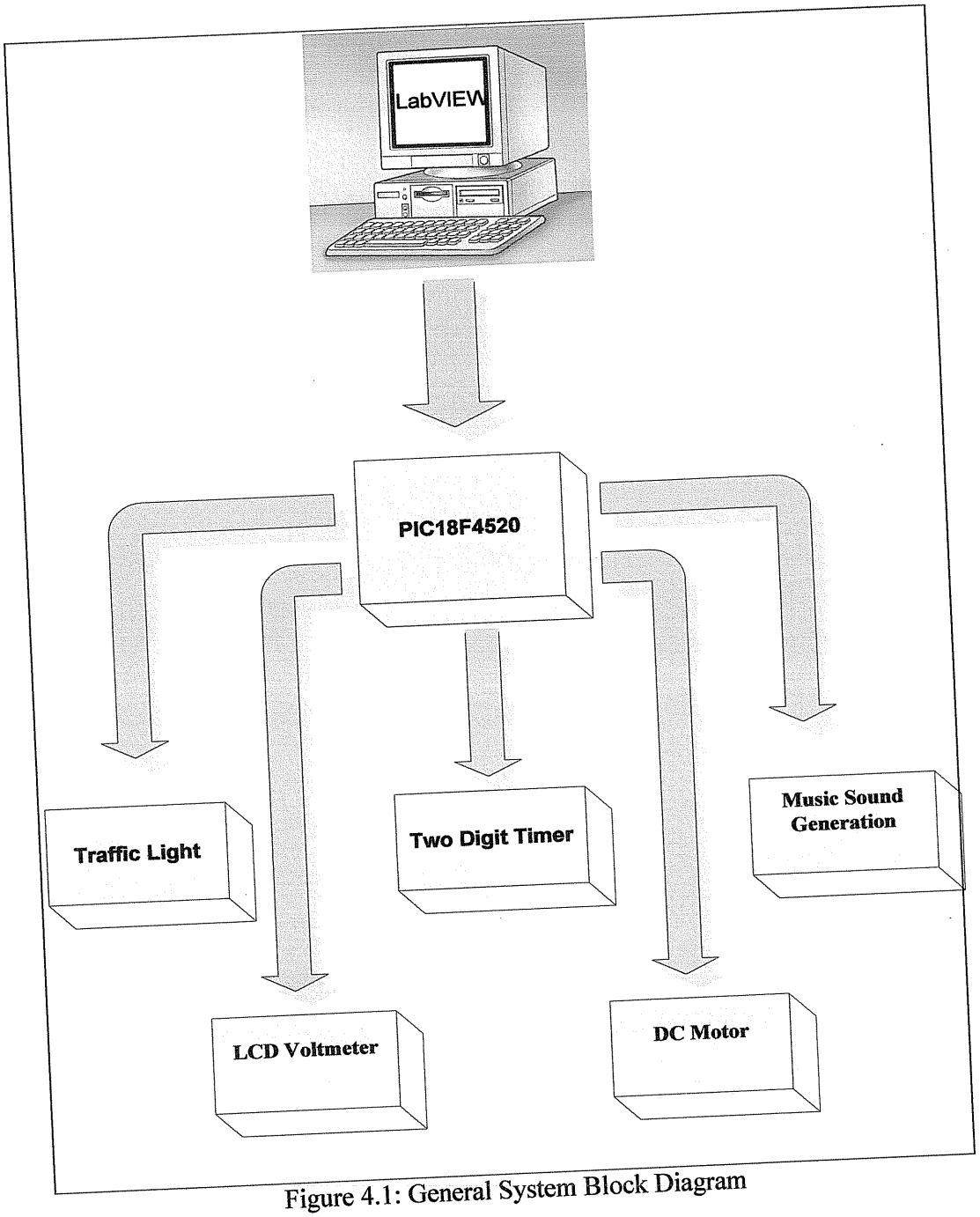


Figure 4.1: General System Block Diagram

4.3.2 Traffic Light Experiment Block Diagram

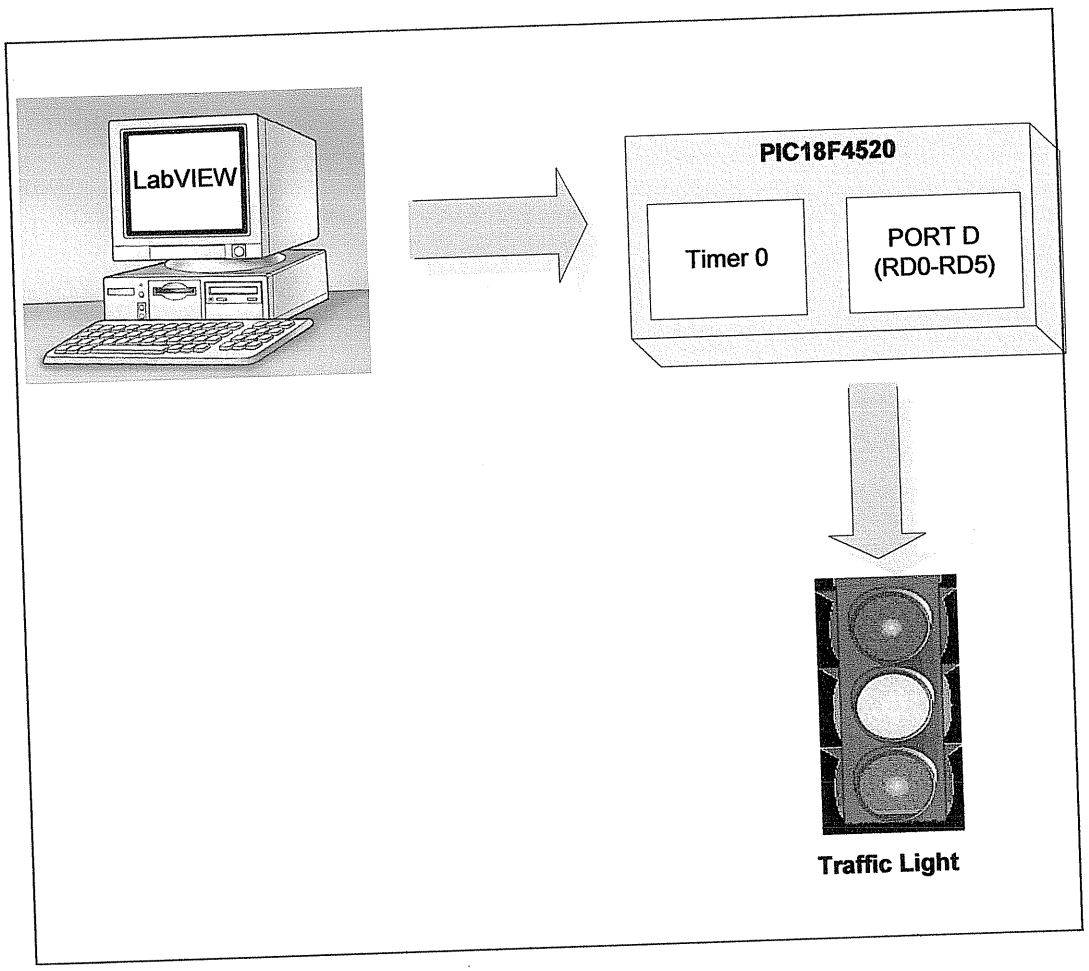


Figure 4.2: Traffic Light Experiment Block Diagram

4.3.3 Music Tones Generation Experiment Block Diagram

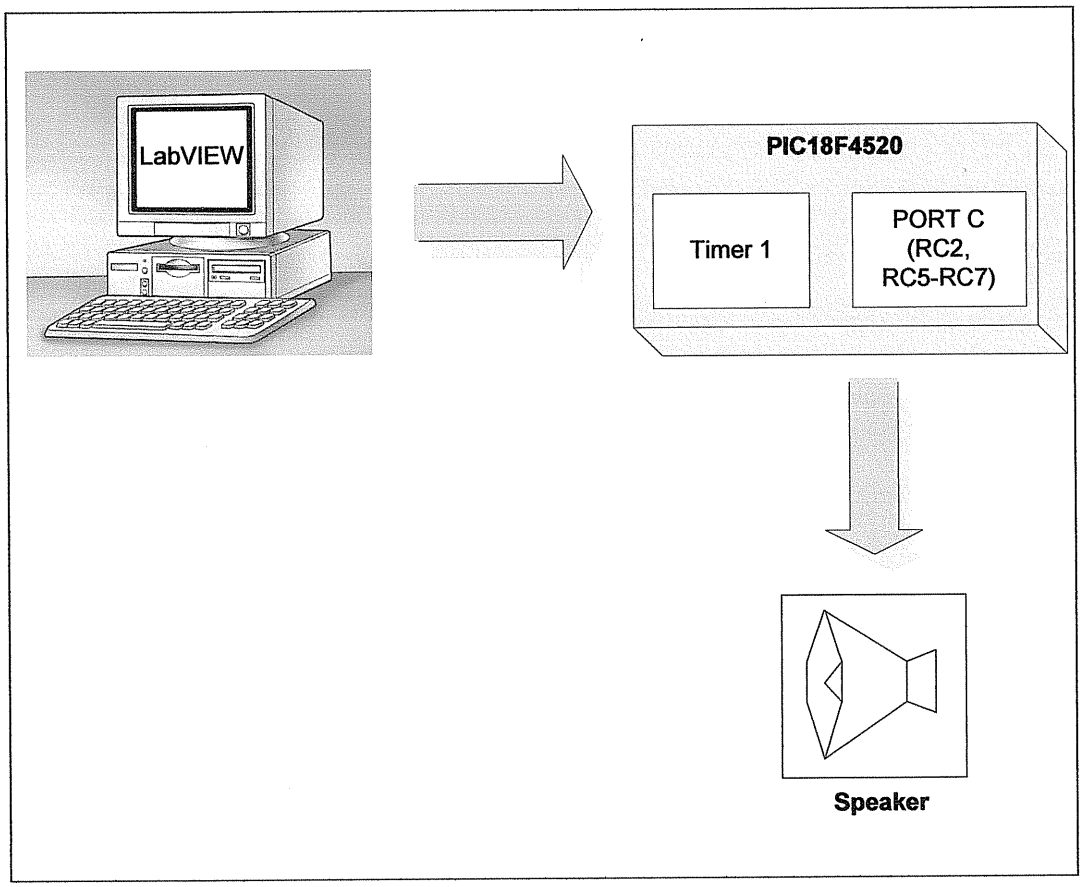


Figure 4.3: Music Tones Generation Experiment Block Diagram

4.3.4 Two Digit Timer Experiment Block Diagram

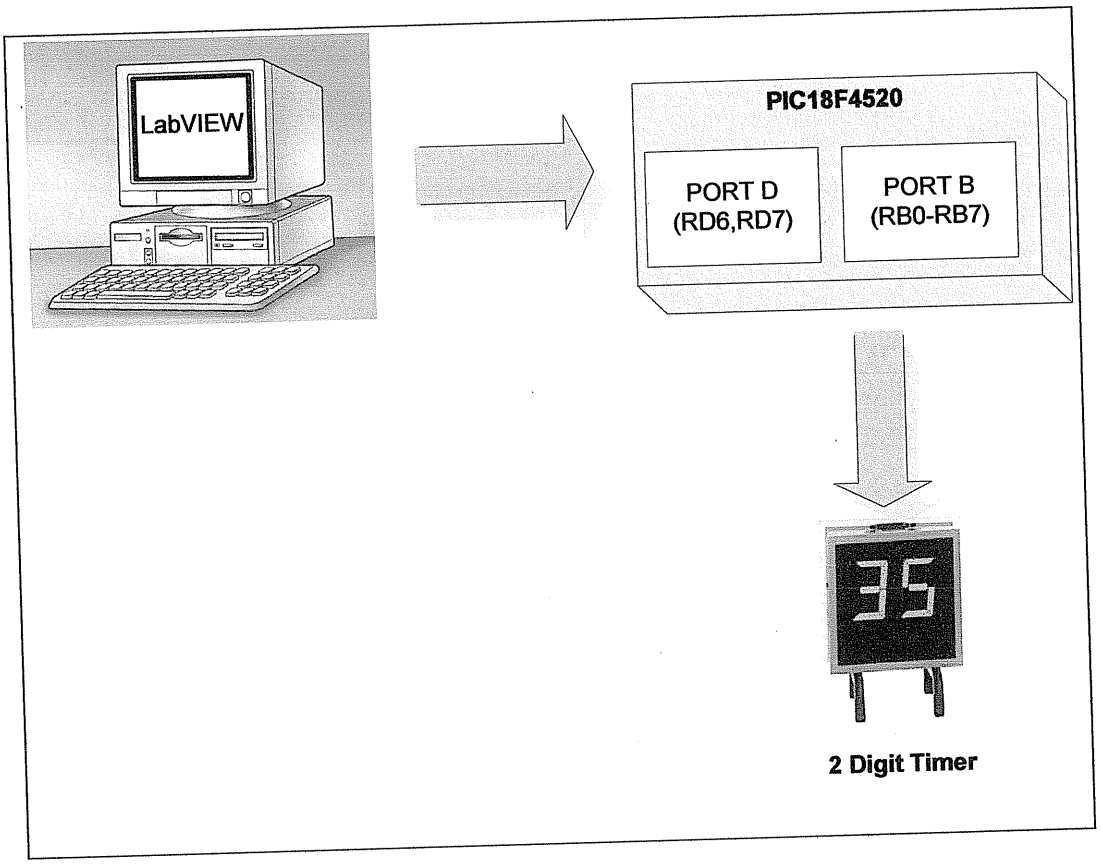


Figure 4.4: Two Digit Timer Experiment Block Diagram

4.3.5 DC Motor Experiments Block Diagram

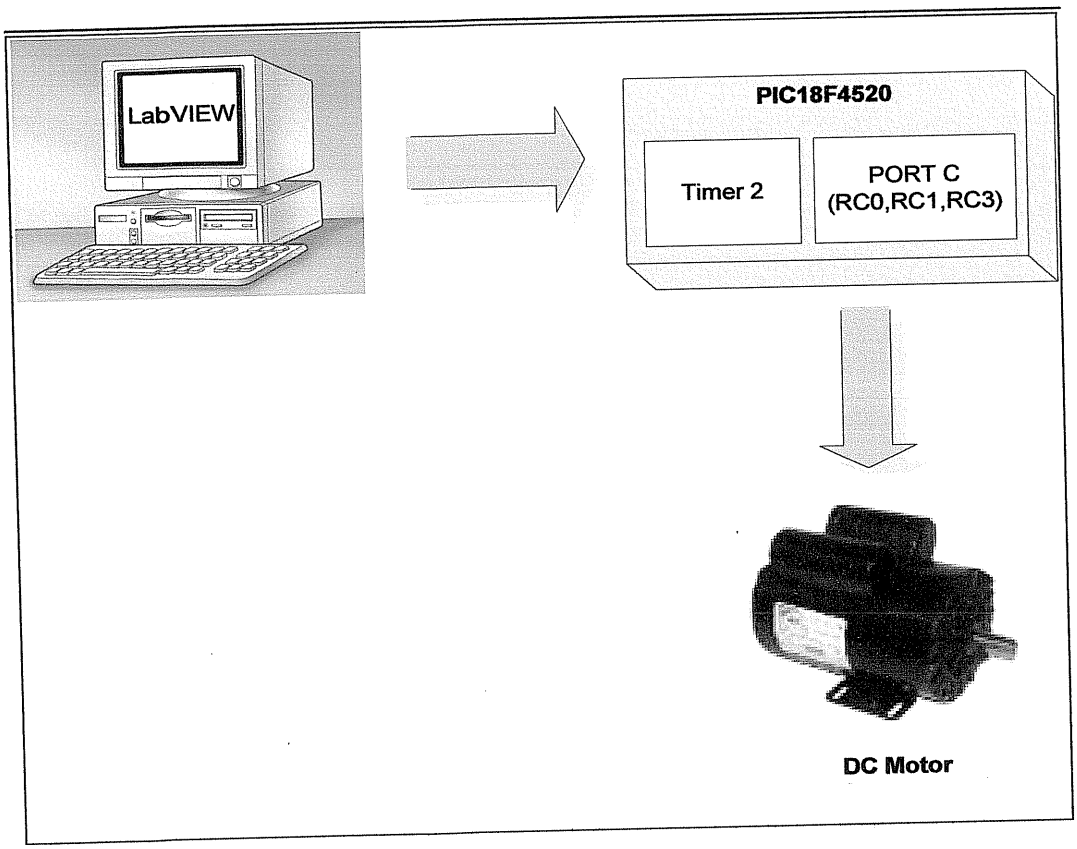


Figure 4.5: DC Motor Experiment Block Diagram

4.3.6 LCD Voltmeter Experiment Block Diagram

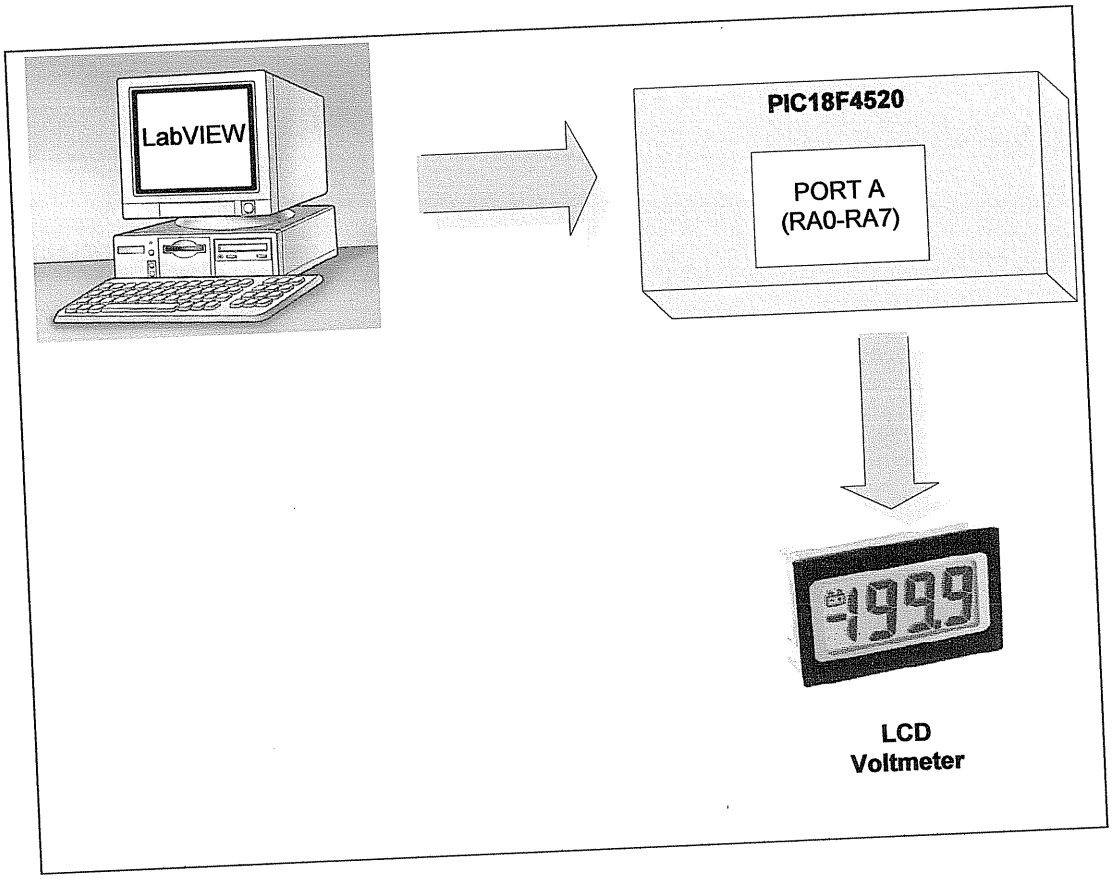


Figure 4.6: LCD Voltmeter Experiment Block Diagram

4.4 System Circuits

4.4.1 Interfacing Circuit of PIC18F4520 and DAQ

Project aims to establish a GUI environment for PIC microcontroller, that helps the student to determine the application he wants to run from specified menu in LabVIEW, and then executes it on the PIC18F4520 which is already programmed using MP-LAB and fixed in the system.

As shown in Figure 4.7, PIC18F4520 is connected to the PC using DAQ, which is connected to PIC using three output digital lines to give eight different patterns, one of these patterns used to reset the system which is (000), and the other seven pattern are enough to run seven different applications, only five applications have been built in the project and have the patterns (001, 010, 011,100,101), and the patterns 110 and 111 are ignored.

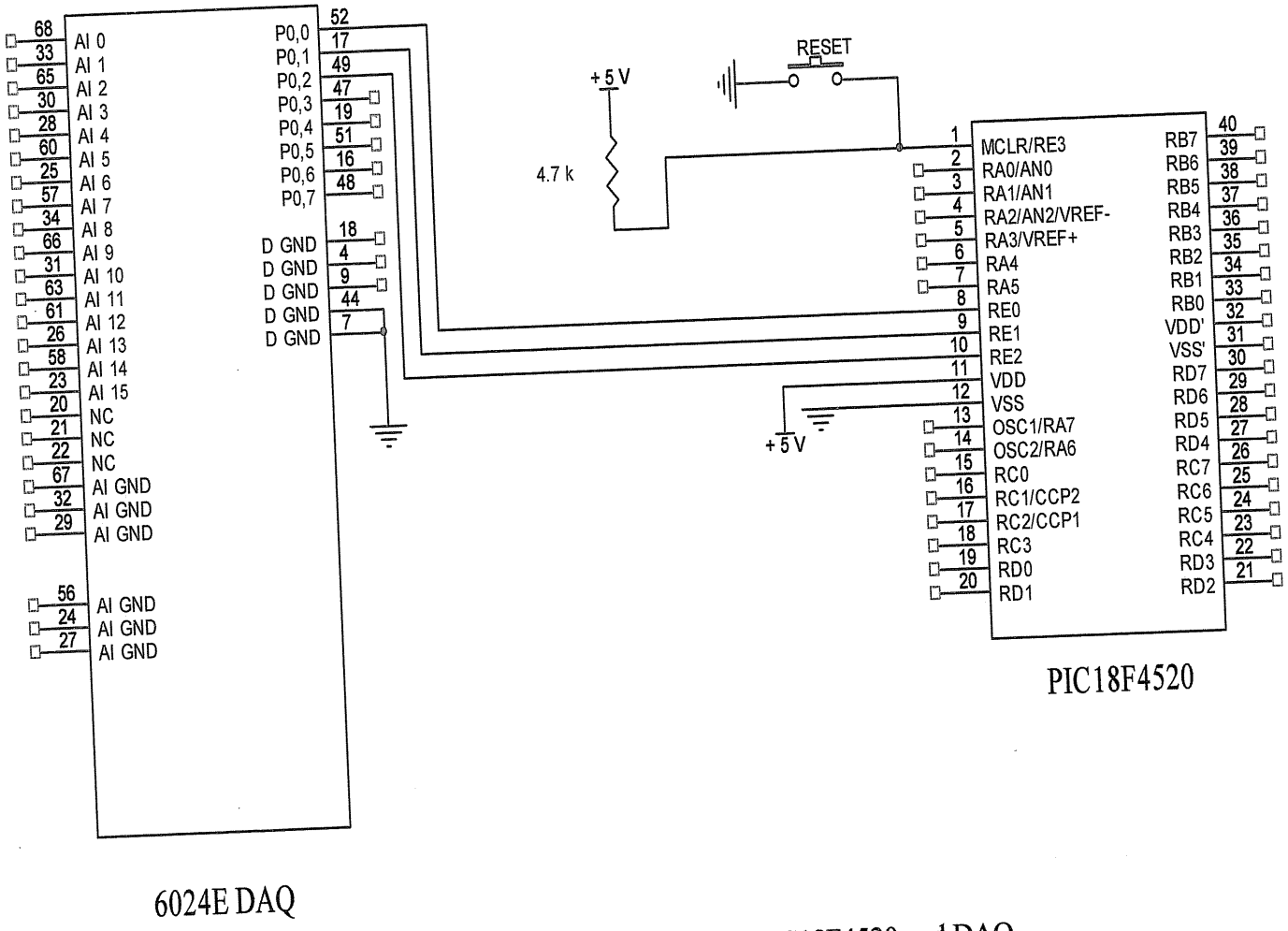


Figure 4.7: Interfacing Circuit of PIC18F4520 and DAQ

4.4.2 Traffic Light Experiment Circuit

Traffic light experiment is for two junction roads, and is implemented in the system using two sets of LEDs (Red, Yellow, and Green), each set is used to represent one junction of the road.

Student can run this experiment by choosing it from GUI menu implemented in the LabVIEW and then press the run button, pressing the run button involves that the required pattern which is (001) is sent from PC to PIC18F4520 through the DAQ.

PIC18F4520 checks continuously the status of the three pins which are used to specify the experiment that will be executed, and if it finds that RE0=1 and the other two pins equal to zero, then the traffic light experiment will go on.

Traffic light circuit components are connected to PIC18F4520 using the low six pins of PORTD which are RD0 to RD5 as shown in Figure 4.8 LEDs are common cathode and so to turn on the LED one must be sent from PIC18F4520 to it.

The code downloaded on PIC18F4520 to execute this experiment informs PIC18F4520 to send six different values for each case as shown in Table 4.1, each of these values must remain stable on PORTD for specific period of time this is implemented using delays which generated using timer0.

If the student wants to reset the experiment he must press reset button, and so the PIC18F4520 will send 0b xx000000 to PORTD, and so LEDs will turned off until he chooses the application again and runs it.

Table 4.1: Traffic Light Sequence

SET1	SET2	DURATION	PORTD
Red	Green	15 seconds	0b xx100001
Red	Yellow	2 seconds	0b xx010001
Red & Yellow	Red	2 seconds	0b xx001011
Green	Red	15 second	0b xx001100
Yellow	Red	2 seconds	0b xx001010
Red	Red & Yellow	2 seconds	0b xx011001

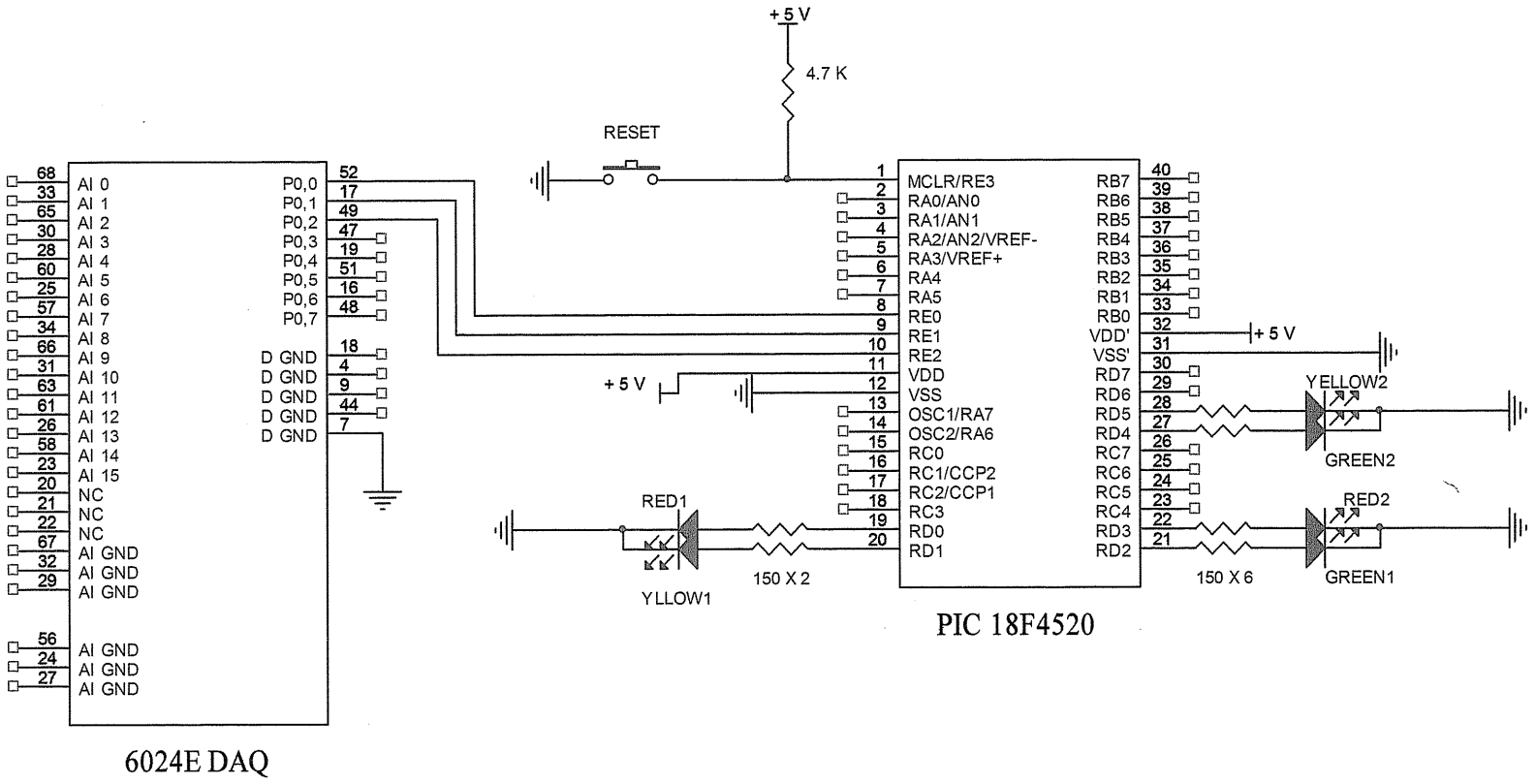


Figure 4.9: Traffic Light Experiment Circuit

4.4.3 Music Tones Generation Experiment Circuit

Music Tones experiment aims to generate seven different music tones, and is implemented in the system by connecting speaker circuit - consists of the speaker, other basic electronic devices, and seven switches for the tones - to PIC18F4520.

Student can run this experiment by choosing it from GUI menu implemented in LabVIEW and then press the run button, pressing the run button involves that the required pattern which is (010) is sent from PC to PIC18F4520 through the DAQ.

PIC18F4520 checks continuously the status of the three pins which are used to specify the experiment that will be executed, and if it finds that RE1=1 and the other two pins equal to zero, then the music tones generation experiment will go on.

Music tones generation experiment components are connected to PIC18F4520 using five pins of PORTC which are RC2, RC4, and RC5 - RC7 as shown in Figure 4.9, since there are seven music tones so in the ordinary way seven pins must be used with each tone's switch to generate it, and to reduce number of pins necessary for this experiment 74LS148 encoder is used, and so five pins are used to determine which tone will be generated instead of seven pins.

Pins (RC5-RC7) are configured as input pins and connected with the output of the encoder and pin (RC4) is configured as output and is connected to the enable

pin of the encoder. RC2 is used as output pin to send square wave from PIC18F4520 to the speaker circuit, a square wave with specific frequency must be produced as to generate a tone as shown in Table 4.2, and this is done using timer1.

If the student wants to reset the experiment he must press reset button, and so the PIC18F4520 will send 0 to RC2, and so the speaker will turned off until he chooses the application again and runs it.

Table 4.2: Tones frequencies and corresponding pins values

NOTE	FREQUENCY(Hz)	KEYSWITCH(RC7 RC6 RC5)
DO	261.78	000
RAY	294.12	001
ME	328.95	010
FAH	349.65	011
SOH	390.63	100
LAH	438.60	101
TE	495.05	111

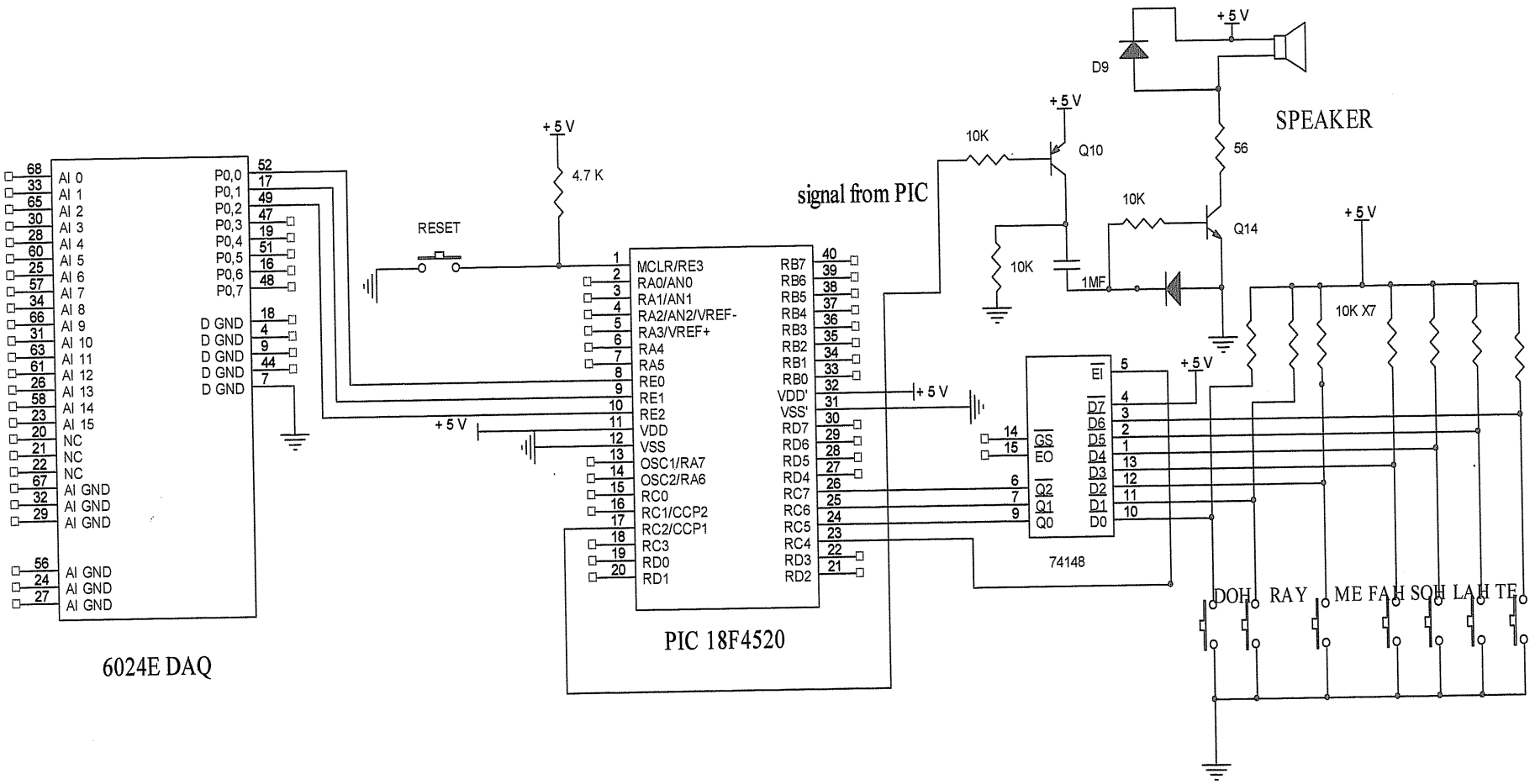


Figure 4.9: Music Tones Generation Experiment Circuit

4.4.4 Two Digit Timer Experiment Circuit

Two Digit Timer experiment purpose is to count from 0.0s to 9.9s, the timer increments it's count each time the student presses the increment switch on the system board, and clears it's count if clear button is pressed, this experiment is implemented using two seven segments to display the count of the timer with additional electronic components as shown in Figure 4.10.

Student can run this experiment by choosing it from GUI menu implemented in the LabVIEW and then press the run button, pressing the run button involves that the required pattern which is (011) is sent from PC to PIC18F4520 through the DAQ.

PIC18F4520 checks continuously the status of the three pins which are used to specify the experiment that will be executed, and if it finds that RE2=0 and the other two pins equal to one the timer will go on.

Two digit timer circuit components are connected to PIC18F4520 using ten pins, all the pins of PORTB from RB0 to RB7 are connected with the two seven segments pins at the same time, the reason for this connection is to use less number of I/O ports, since if we connect each seven segment with different I/O lines sixteen lines are needed for the two seven segment.

The remaining two pins are used in this experiment are RD6 and RD7 these pins acts as input and output lines, when they are configured as output they used to enable one of the seven segment at a time and send the ASCII code of the number to

be displayed on it while the other sev[redacted]en segment is disabled, and then after small delay enables the other seven segment [redacted]and send the ASCII code of the number to be displayed on it while the first seven [redacted]egments is disabled, and because of the fast execution of the PIC18F4520 the stud[redacted]nt can't notice that the seven segments go on and off.

When RD6 is input, PIC18F45[redacted]0 check if the clear switch is pressed, and if it is pressed then the timer is cleared, [redacted]When RD7 is input, PIC18F4520 check if the increment count switch button is pres[redacted]ed, and if it is pressed then the timer's count is incremented.

If the student wants to reset th[redacted]e experiment he must press reset button, and so the PIC18F4520 will disable the tw[redacted] seven segments by sending one on RD6 and RD7, and so seven segments turned [redacted]off until he chooses the application again and runs it.

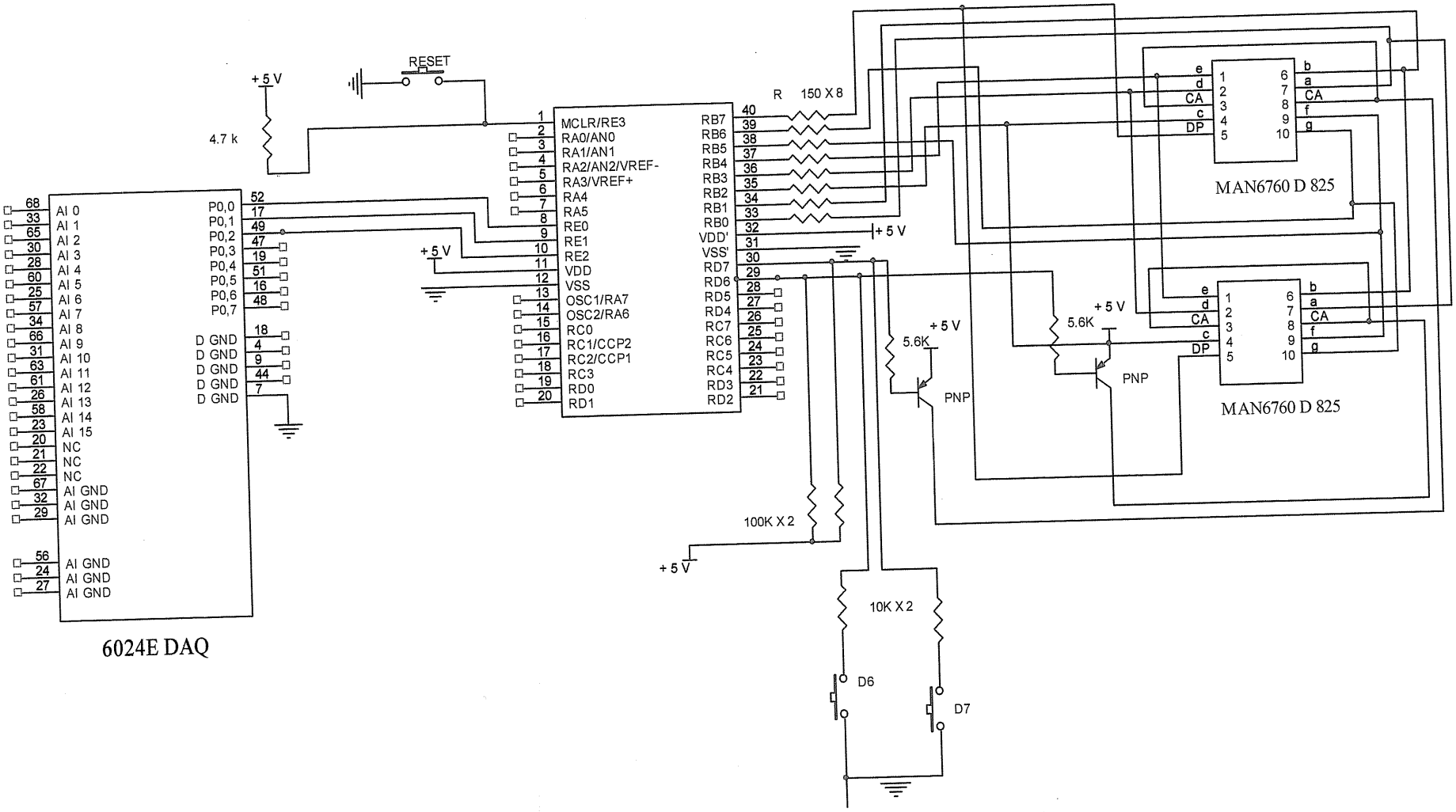


Figure 4.10: Two Digit Timer Experiment Circuit

4.4.5 DC Motor Experiment Circuit

DC Motor experiment is used to run DC Motor and control the direction of the rotation, this experiment is implemented using DC Motor and H-Bridge circuit.

Student can run this experiment by choosing it from GUI menu implemented in the LabVIEW and then press the run button, pressing the run button involves that the required pattern which is (101) is sent from PC to PIC18F4520 through the DAQ.

PIC18F4520 checks continuously the status of the three pins which are used to specify the experiment that will be executed, and if it finds that RE1=0 and the other two pins equal to one DC Motor will go on.

DC Motor circuit components are connected to PIC18F4520 using three pins from PORTC which are RC0, RC1 and RC3 as shown in Figure 4.11, RC3 is configured as input and used to read the value sent from the DAQ to determine the direction of rotation, RC1 and RC0 are used to send the speed and the direction of rotation read from DAQ respectively.

The code downloaded on PIC18F4520 to execute this experiment informs PIC18F4520 to send fix speed to DC Motor which is 20 kHz, this frequency is generated using PWM and sent to H-Bridge from CPP2 (RC1) pin.

If the student wants to reset the experiment he must press reset button, and so the PIC18F4520 will send DC signal on CCP2 and so the motor will turned off until he chooses the application again and runs it.

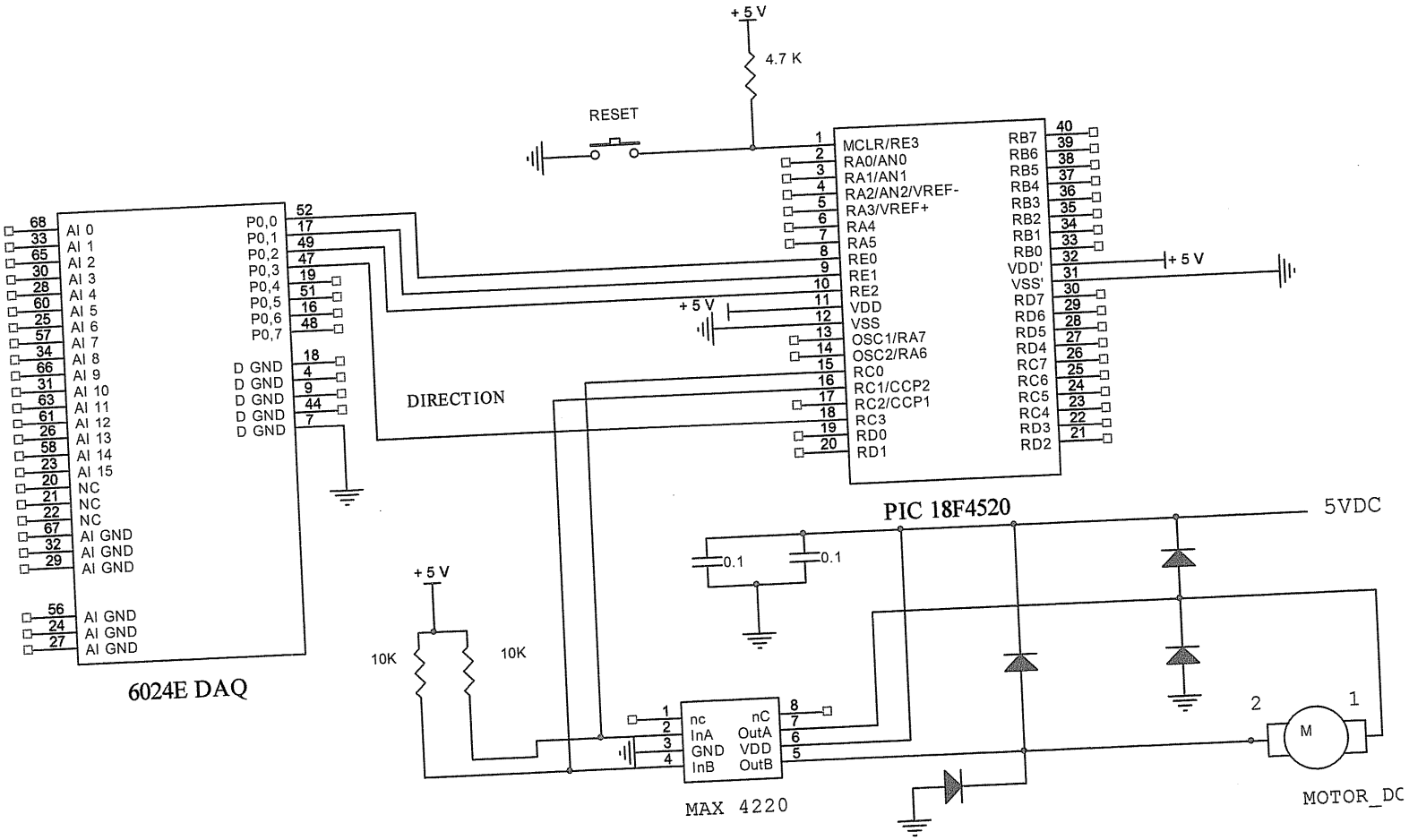


Figure 4.11: DC Motor Experiment Circuit

4.4.6 LCD Voltmeter Experiment Circuit

LCD Voltmeter experiment aims to measure the value of the voltage from 0V to 5V and display its value on LCD screen, this experiment is implemented using potentiometer and LCD screen as shown in Figure 4.12.

Student can run this experiment by choosing it from GUI menu implemented in the LabVIEW and then press the run button, pressing the run button involves that the required pattern which is (100) is sent from PC to PIC18F4520 through the DAQ.

PIC18F4520 checks continuously the status of the three pins which are used to specify the experiment that will be executed, and if it finds that RE2=1 and the other two pins equal to zero, then this experiment will go on.

LCD voltmeter circuit components are connected to PIC18F4520 using PORTA, RA0 is configured as analog input channel and used to read the voltage signal, RA1 to RA2 are used to send control signals that determine the operation of the LCD (RS, R/W, and E), RA4 to RA7 are used to send data to LCD since it is configured to operate in four bit modes, and so reduce number of pins needed.

PIC18F4520 read the analog signal from RA0 and convert it to digital using A/D converter, which is configured to read from 0V to 5V by setting Vref- to 0V and Vref+ to 5V

If the student wants to reset the experiment he must press reset button, and so the PIC18F4520 will clear command to the LCD and so the LCD will turned off until he chooses the application again and runs it.

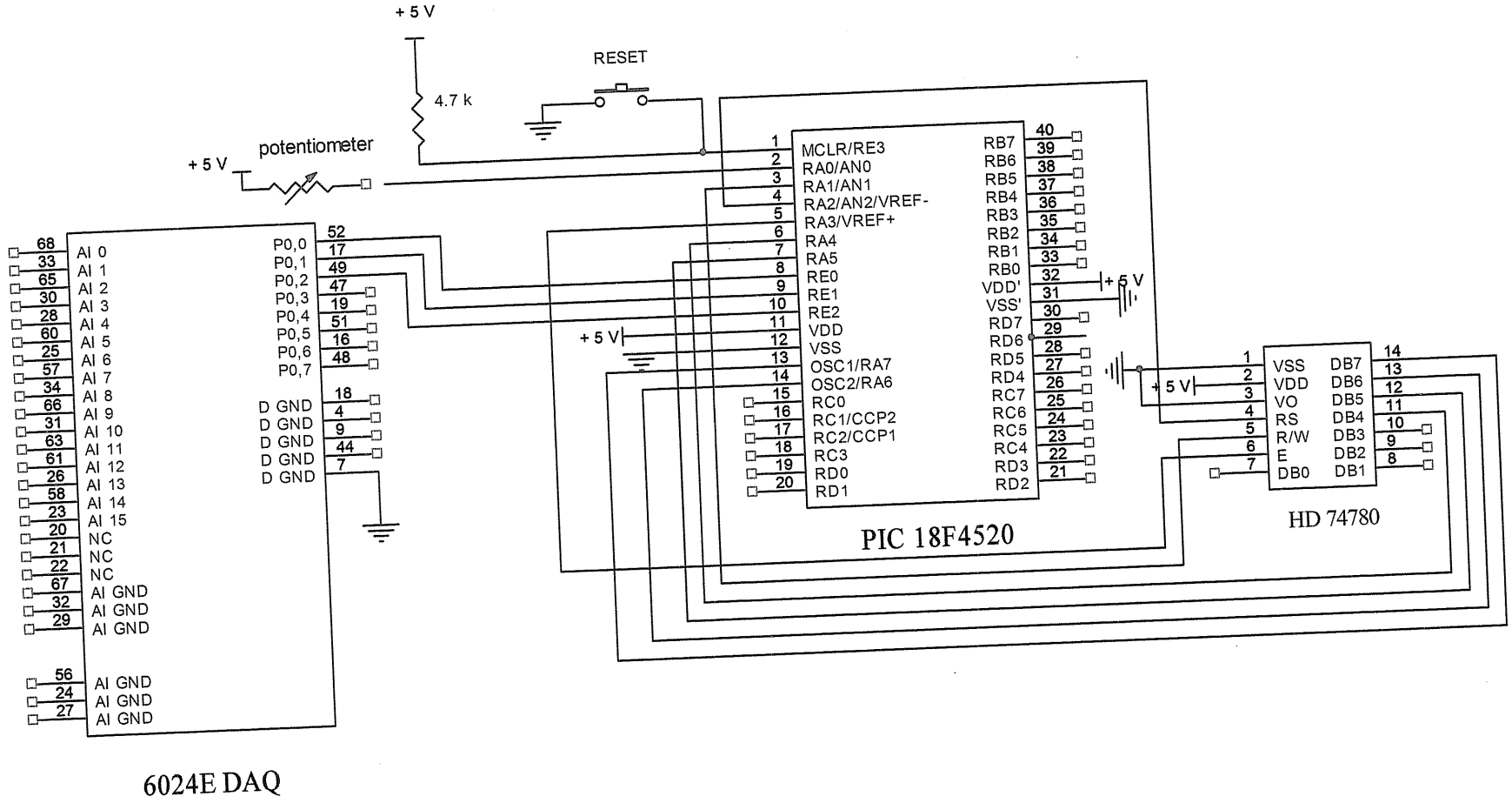


Figure 4.12: LCD Voltmeter Circuit

4.4.7 Parallel Programmer Circuit

One of the problems of this project is that MPLAB ICD2 programmer is damaged, so we design a programmer to complete this project, also this system can use this programmer as a part of it, to help student to change in code of some experiments and then load it to PIC18F4520 directly using Win PIC 800 (programmer software).

Figure 4.13 describes how the circuit works, by using parallel port with a driving circuit (inverters and resistors and capacitors, and transducer of 12 volt), that connect to some MCLR, V_{DD}, RB6, RB7.

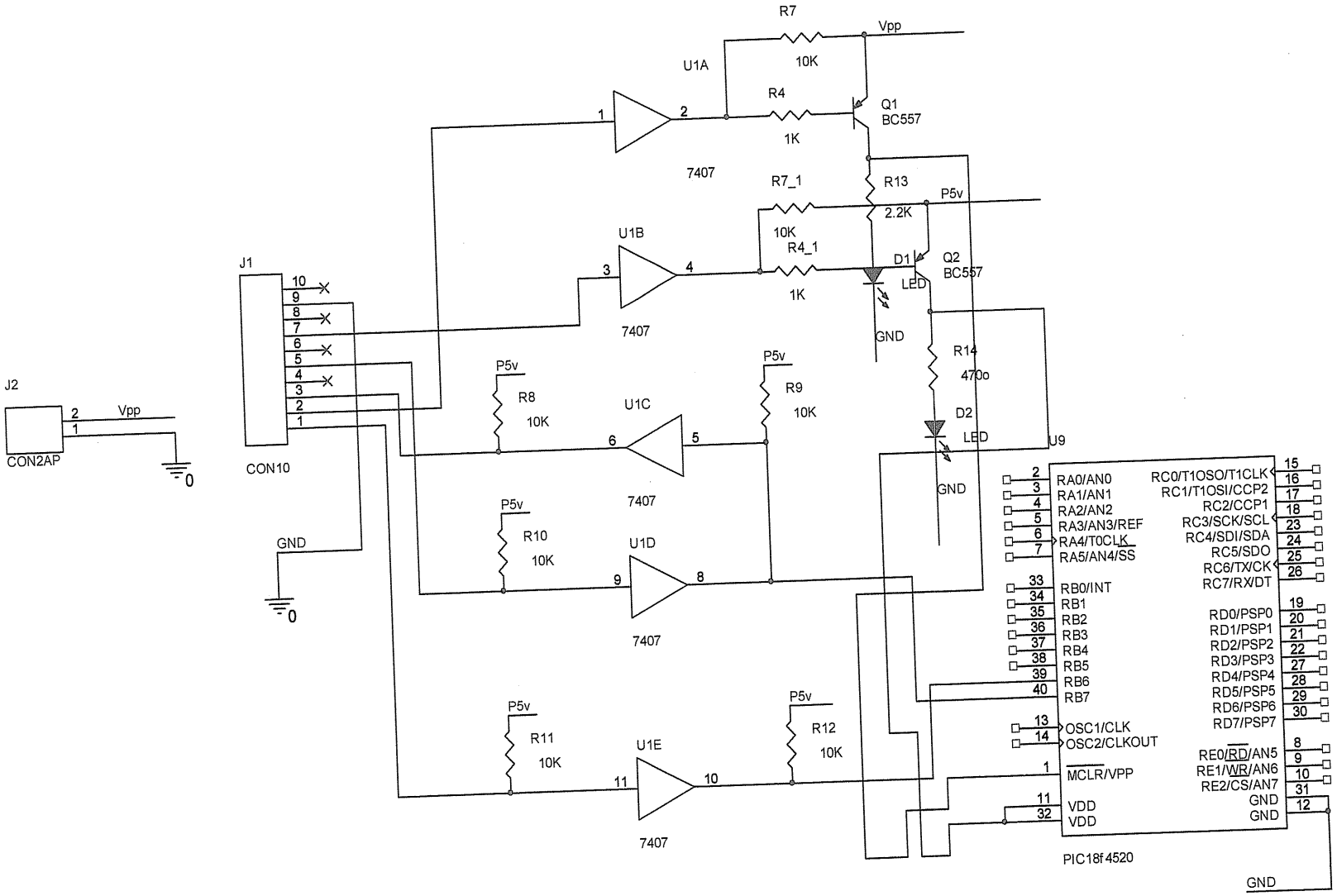


Figure 4.13: Parallel Programmer Circuit

Chapter Five

Software System Design

Chapter Five

Software System Design

5.1 Introduction

This chapter includes software system implementation depends on PIC programming and software tools, and describe the Flowcharts and algorithms of all the system.

5.2 Software Tools

5.2.1 MPLAB IDE Software

MPLAB IDE is a software program that runs on a PC to develop applications for Microchip microcontrollers. It is called an Integrated Development Environment, or IDE, because it provides a single integrated "environment" to develop code for embedded microcontrollers, for more details see Appendix C. [4]

The code of each experiment is written in C language, then use MPLAB C18 compiler, which generate an executable code in machine language, and hex code, also using MPLAB ICD2 programmer in first steps of project, after that this programmer was damaged.

5.2.2 Win PIC 800 Software

This software is supported by parallel programmer, which is used to download hex code to PIC microcontroller; it supports PIC18F4520 when the user chooses setting up hardware as ProPIC2, for more details see Appendix C.

5.3 System Implementation

This section discusses the system software functionality in all of the experiments.

5.3.1 Main Interface

This is the main interface that is implemented in LabVIEW, it gives the user a general description about the project.

Each tab in this interface represents the name of the experiment the user can choose in order to execute it, the last tab shows the parallel programmer description and schematic.

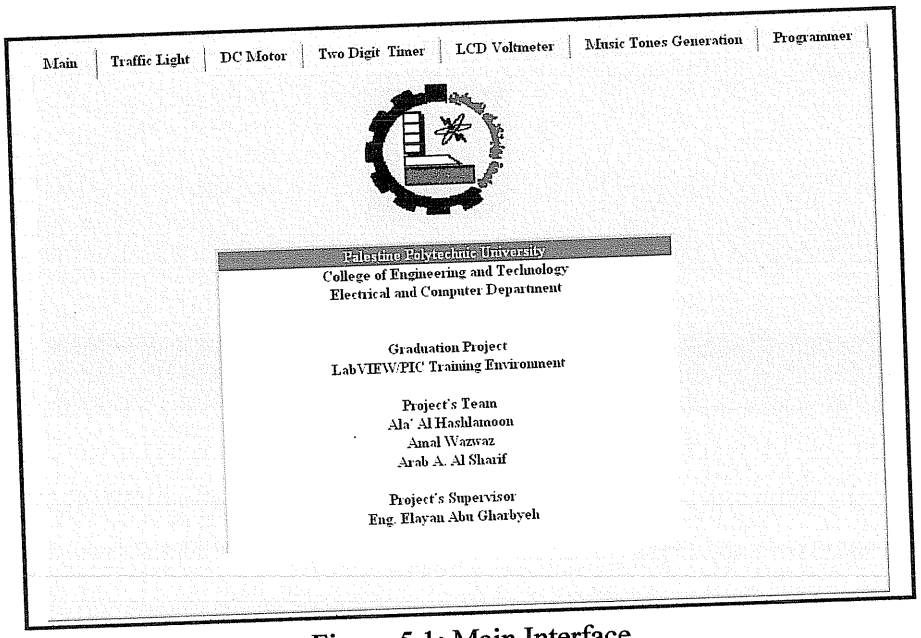


Figure 5.1: Main Interface

Each experiment includes three major tabs, which are:

- Front panel with three tabs, description, schematic and source code.
- Block diagram.

5.3.2 Experiments' Interfaces

- **Front Panel**

  - **Description tab**

The following figures show the description front panel of each experiment; it gives the user a brief description about how it works and which parts of the PIC18F4520 are used to satisfy its purposes also it contains the button which is used to run or reset this application.

- **Traffic Light experiment**

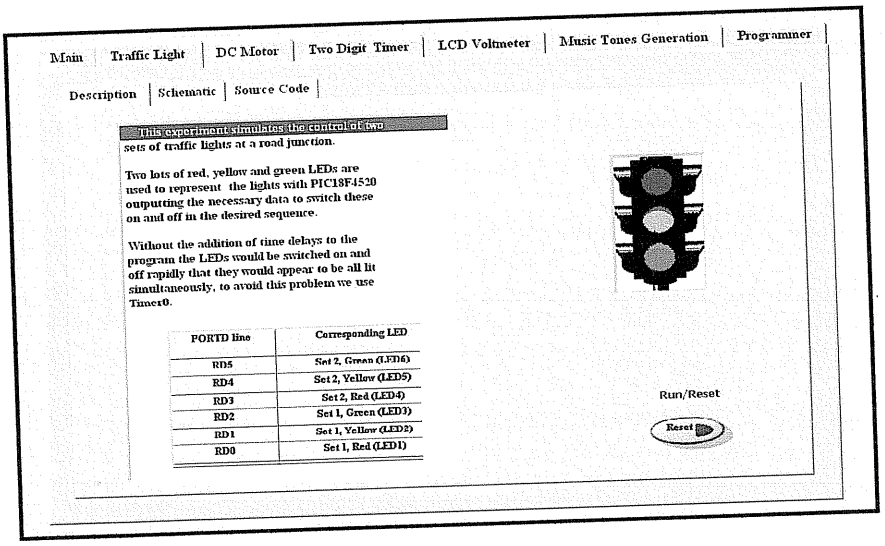


Figure 5.2: Traffic Light Description Front Panel

- **DC Motor experiment**

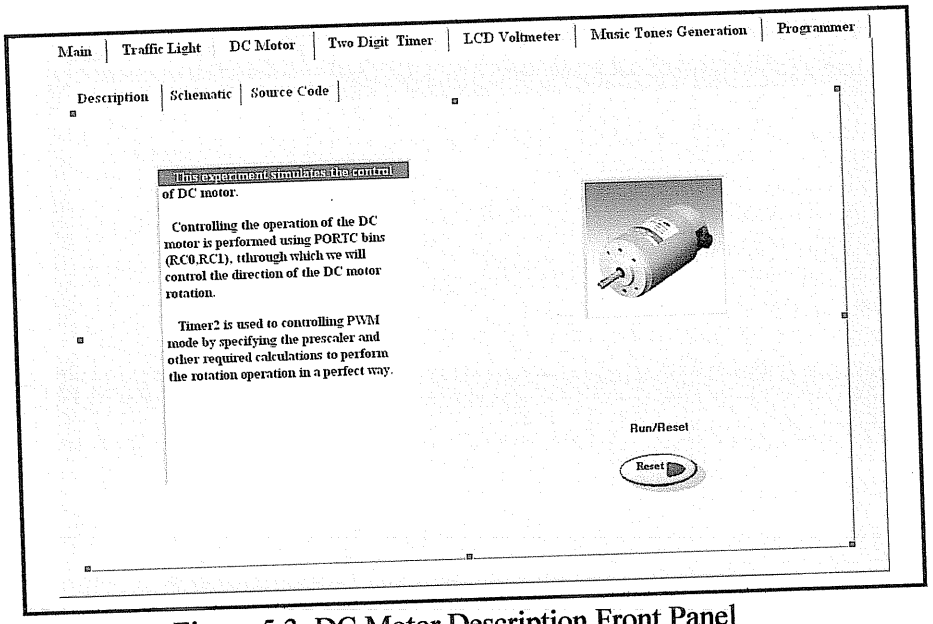


Figure 5.3: DC Motor Description Front Panel

- **Two Digit Timer experiment**

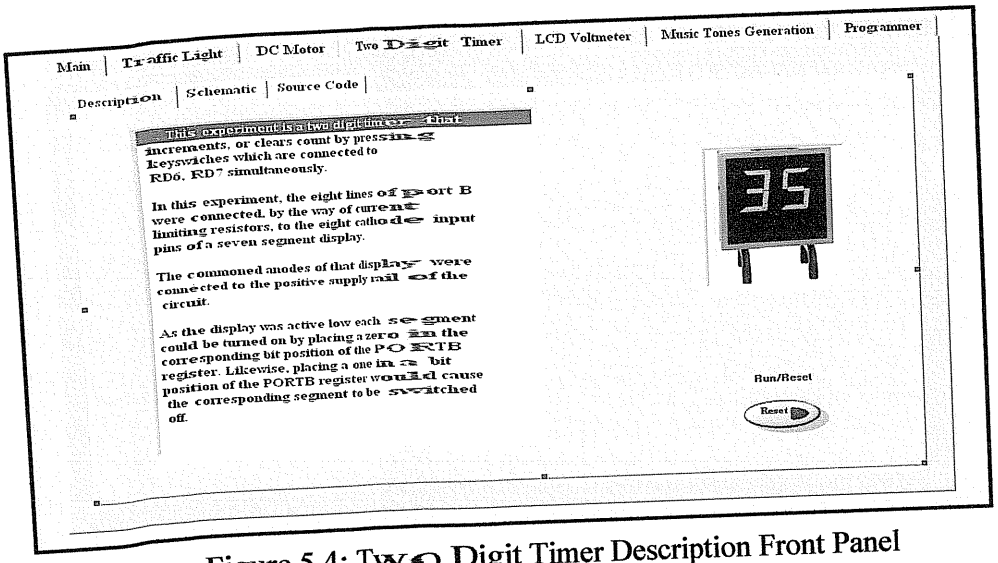


Figure 5.4: Two Digit Timer Description Front Panel

- **LCD Voltmeter experiment**

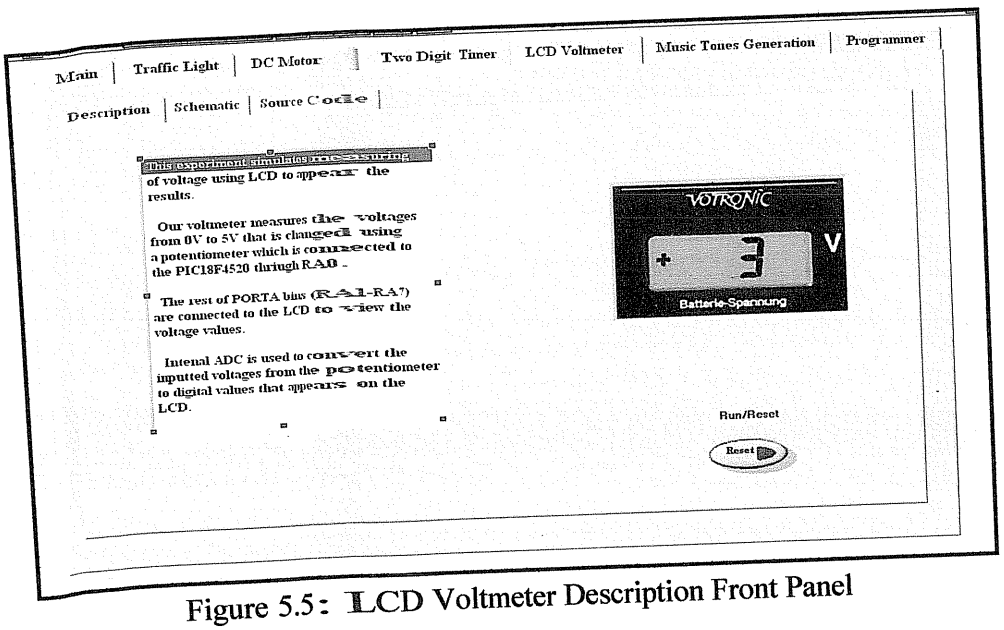


Figure 5.5: LCD Voltmeter Description Front Panel

- **Music Tones Generation experiment**

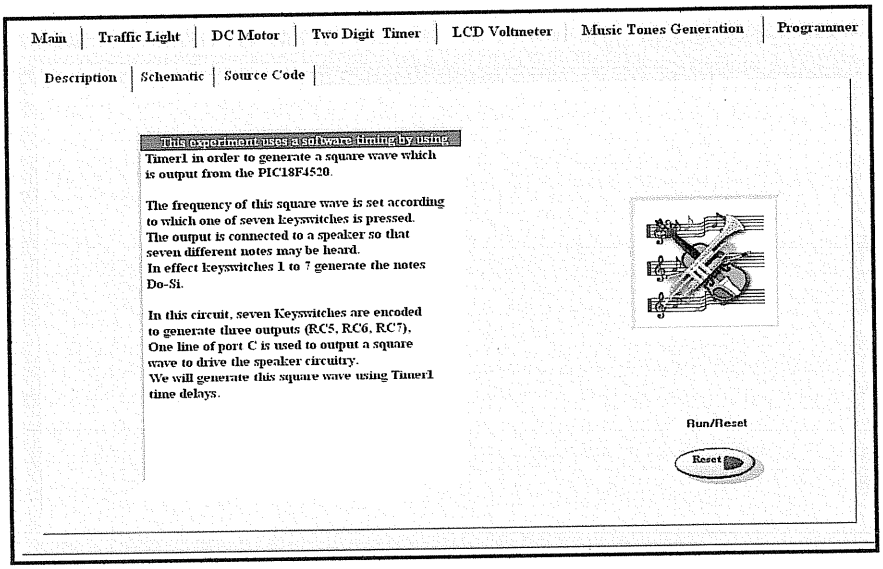


Figure 5.6: Music Tones Generation Description Front Panel

- **Schematic tab**

The following figures, show the schematic of the experiments which are drawn in Orcad to help the user to understand the internal connections and interfaces of each experiment.

- **Traffic light experiment**

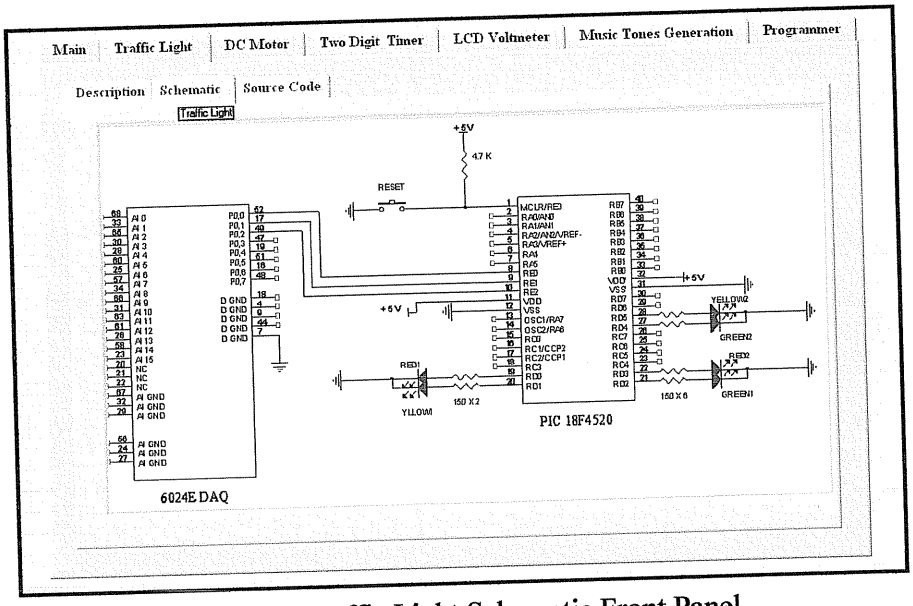


Figure 5.7: Traffic Light Schematic Front Panel

- **DC Motor experiment**

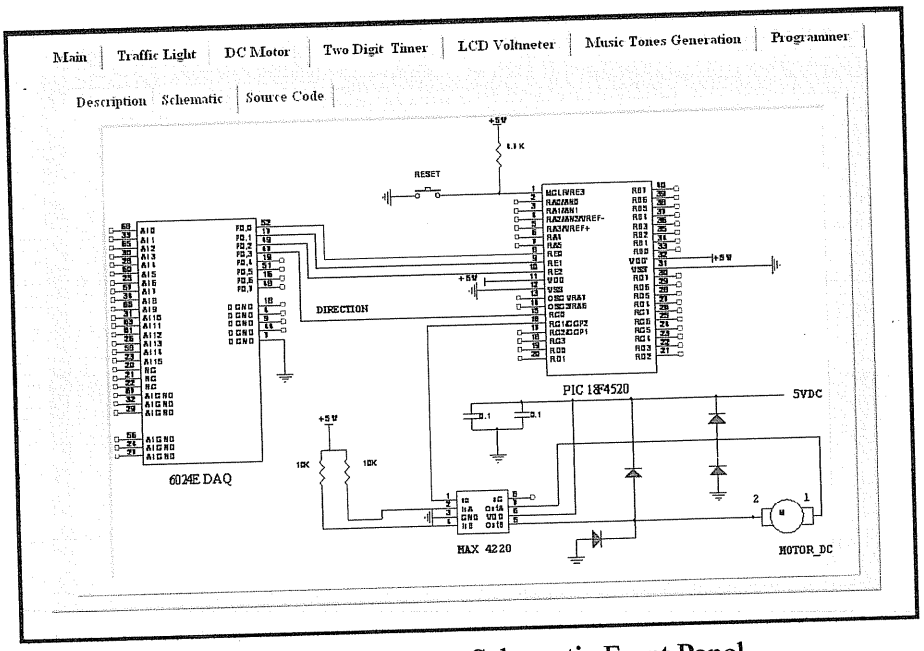


Figure 5.8: DC Motor Schematic Front Panel

- **LCD Voltmeter experiment**

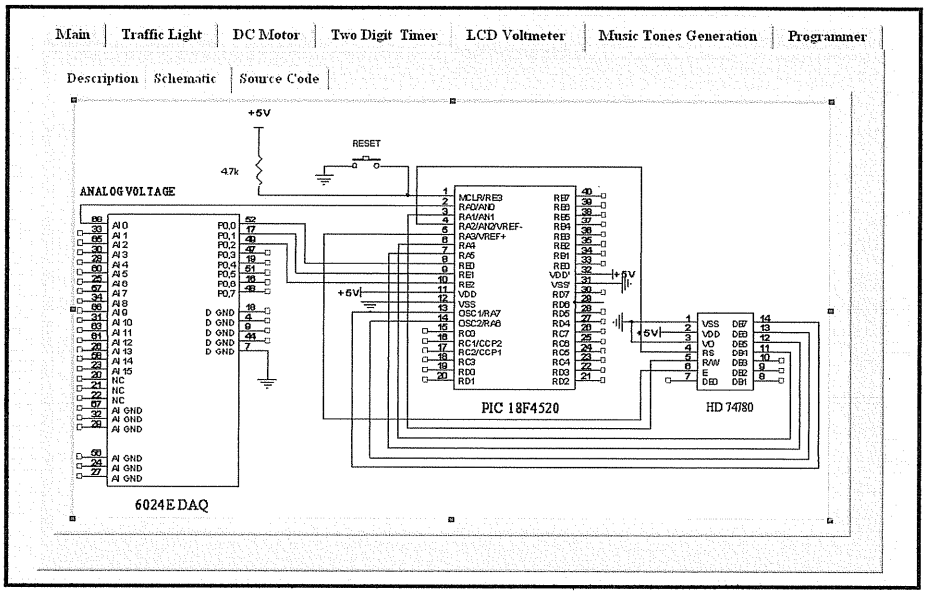


Figure 5.9: LCD Voltmeter Schematic Front Panel

- **Two Digit Timer experiment**

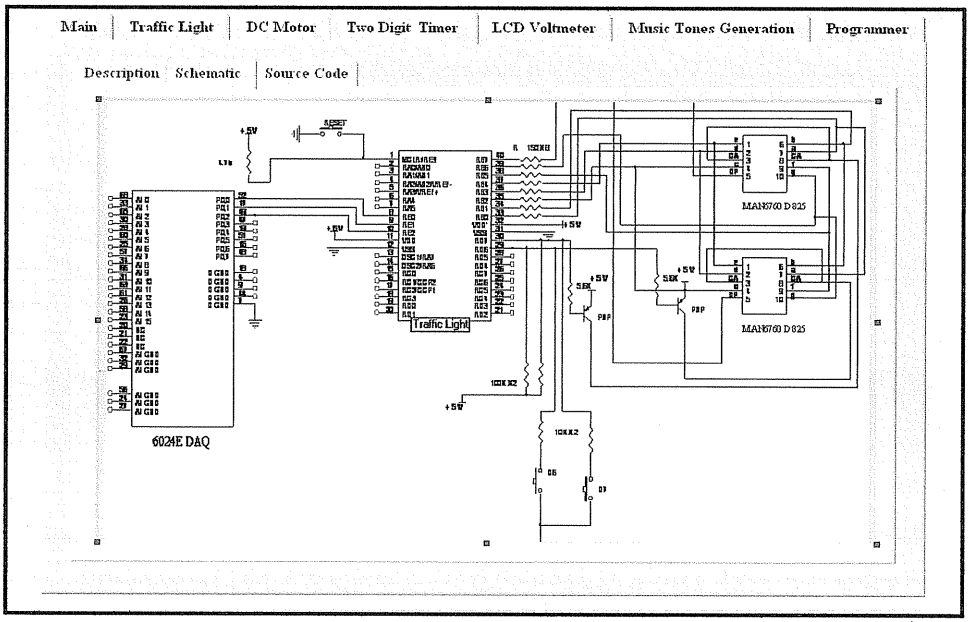


Figure 5.10: Two Digit Timer Schematic Front Panel

- **Music Tones Generation experiment**

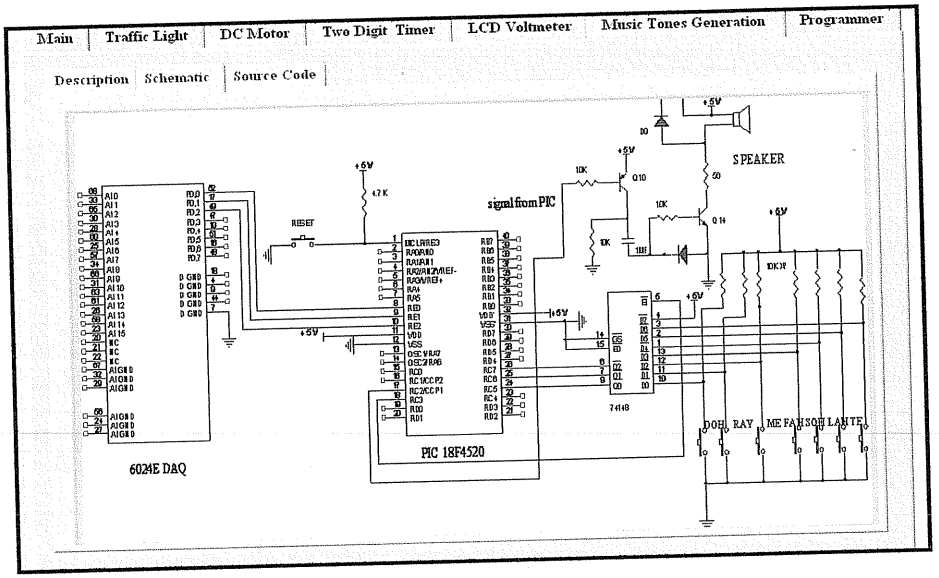


Figure 5.11: Music Tones Generation Schematic Front Panel

- **Source Code tab**

The following figures, show the code of the experiments which is written in C language to help the user to understand how this experiment works relating to software issues.

- **Traffic light experiment**

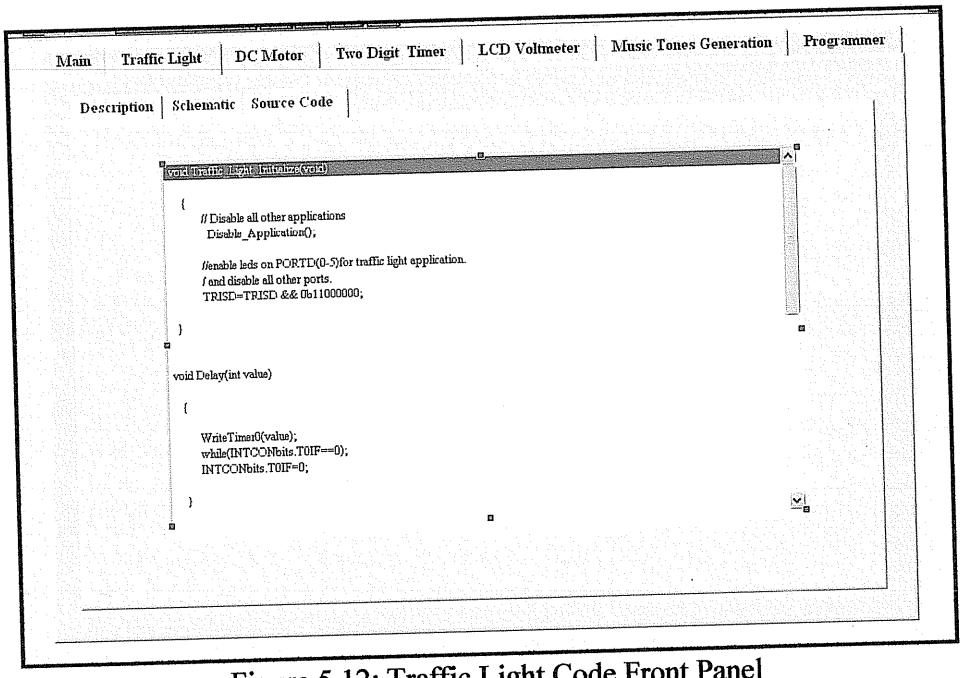


Figure 5.12: Traffic Light Code Front Panel

- **DC Motor experiment**

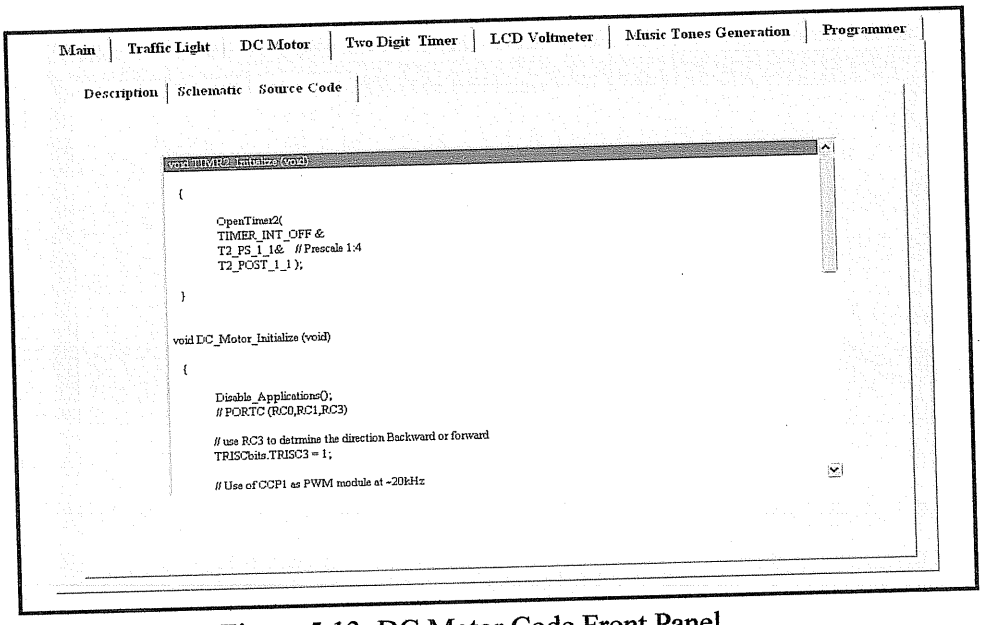


Figure 5.13: DC Motor Code Front Panel

- **Two Digit Timer experiment**

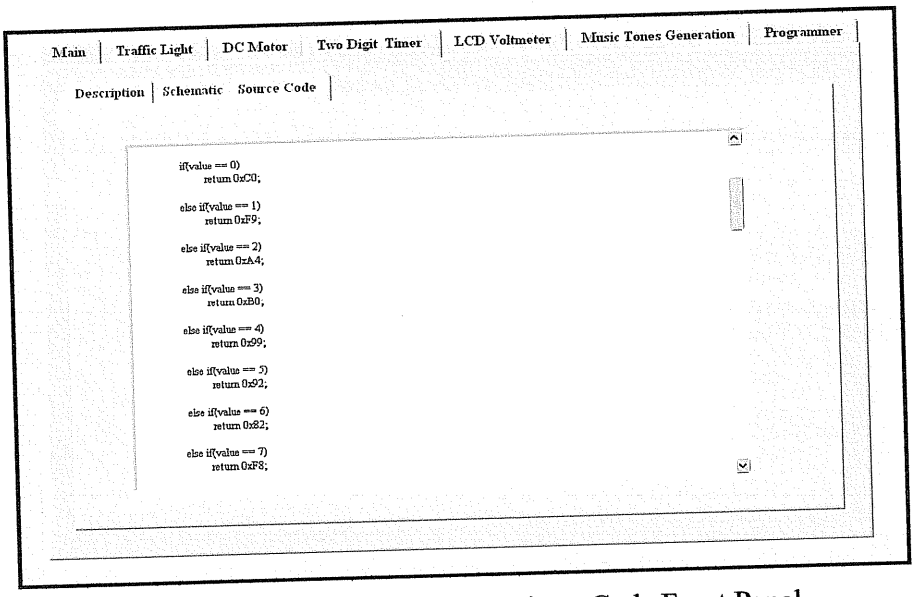


Figure 5.14: Two Digit Timer Code Front Panel

- **LCD Voltmeter experiment**

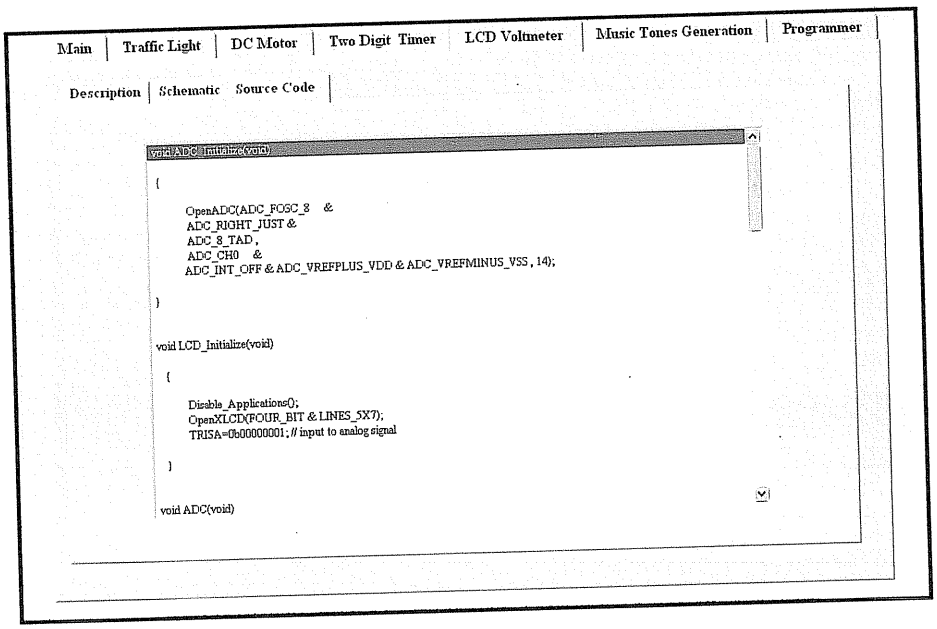


Figure 5.15: LCD Voltmeter Code Front Panel

- **Music Tones Generation experiment**

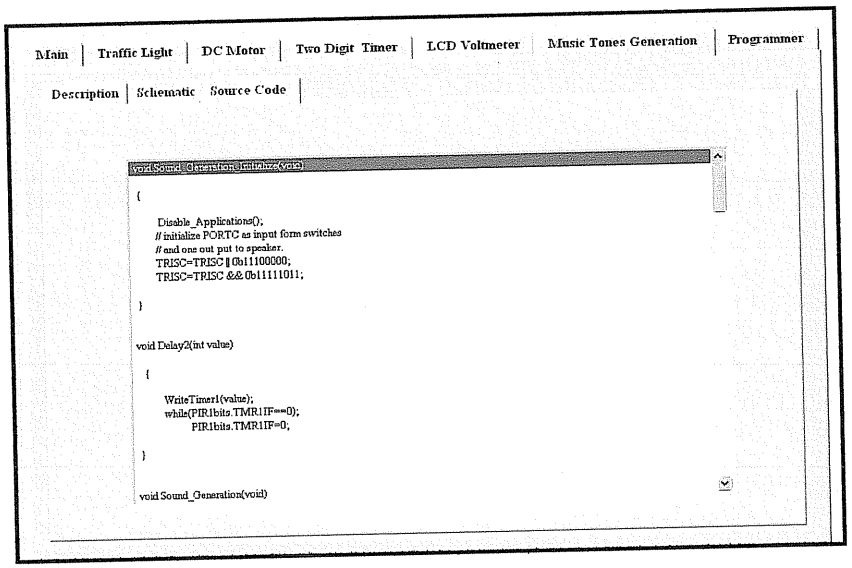


Figure 5.16: Music Tones Generation Code Front Panel

- **Block Diagram**

The following figures show the block diagram of the experiments which shows the main components in LabVIEW that we use in order to make this experiment works properly when interact with LabVIEW.

As shown, the main tab is joined with a case structure to select the active tab, when an experiment tab is active in the front panel, the Boolean button will send

different values to the DAQ according to it's status in order to run or reset this experiment respectively.

When the button is pressed (Run case) the internal case structure will send 1 to the DAQ which will cause to execute traffic light experiment, on the other hand, when the button is not pressed (Reset case) the internal case structure will send 0 to the DAQ which will cause to reset this experiment.

- **Traffic light experiment**

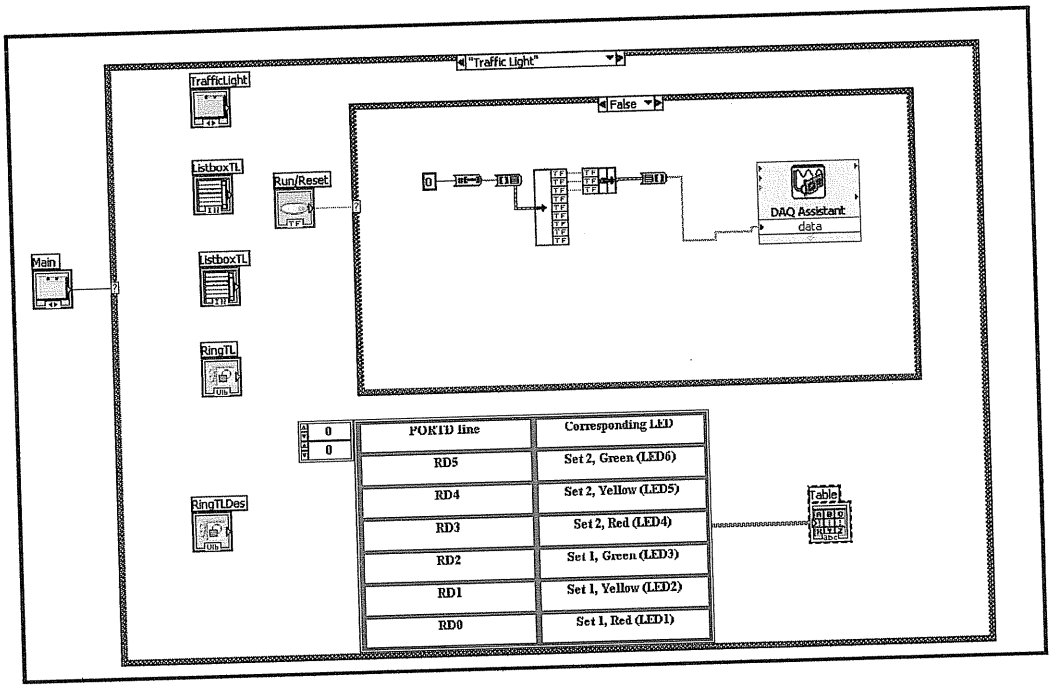


Figure 5.17: Traffic Light Block Diagram

- **DC Motor experiment**

When the button is pressed (Run case) the internal case structure will send 5 to the DAQ which will cause to execute DC Motor experiment, on the other hand, when the button is not pressed (Reset case) the internal case structure will send 0 to the DAQ which will cause to reset this experiment.

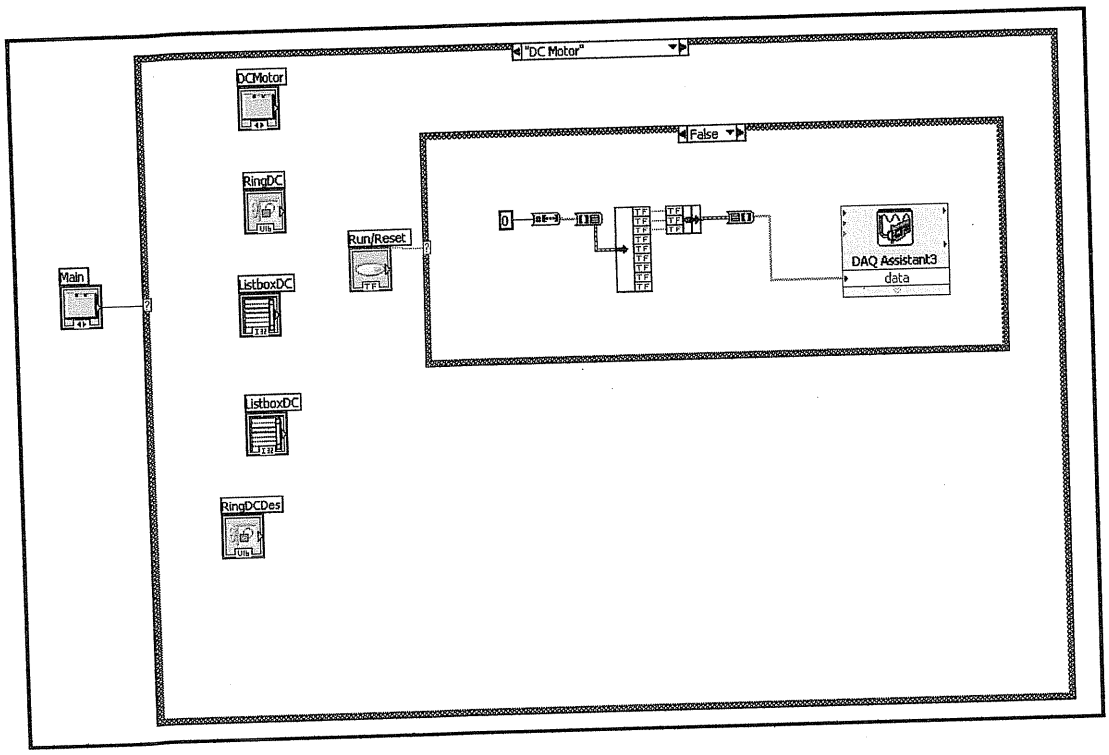


Figure 5.18: DC Motor Block Diagram

- **Two Digit Timer experiment**

When the button is pressed (Run case) the internal case structure will send 3 to the DAQ which will cause to execute Two Digit Timer experiment, on the other hand, when the button is not pressed (Reset case) the internal case structure will send 0 to the DAQ which will cause to reset this experiment.

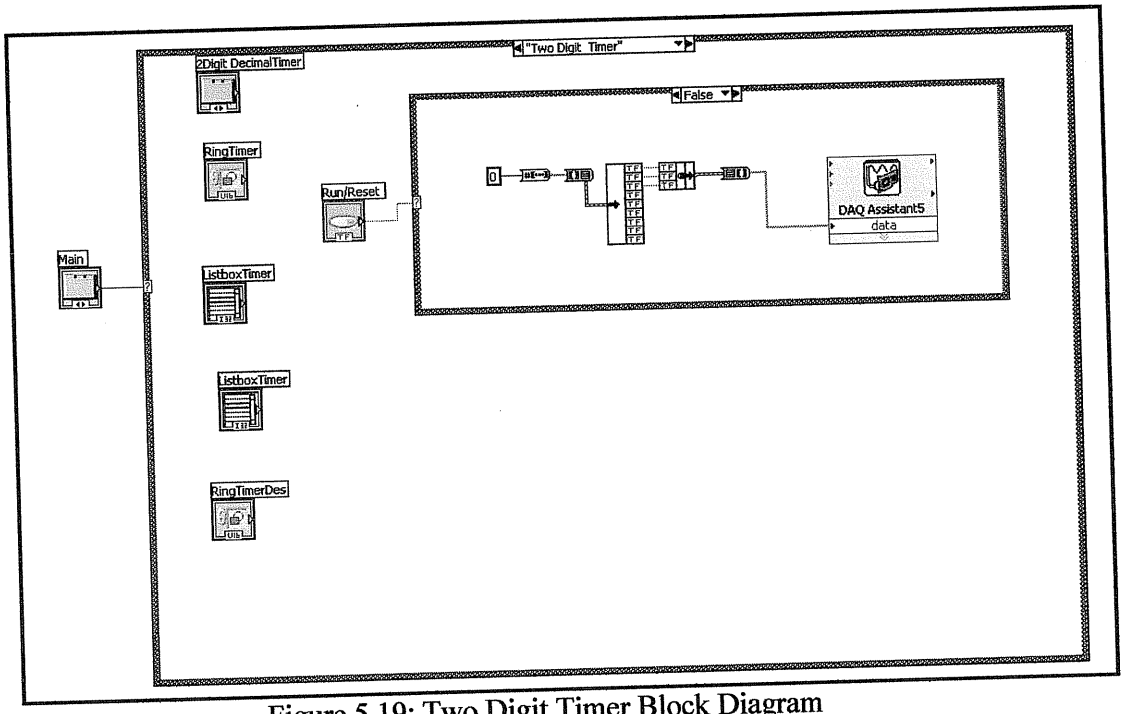


Figure 5.19: Two Digit Timer Block Diagram

- **LCD Voltmeter experiment**

When the button is pressed (Run case) the internal case structure will send 4 to the DAQ which will cause to execute LCD Voltmeter experiment, on the other

hand, when the button is not pressed (Reset case) the internal case structure will send 0 to the DAQ which will cause to reset this experiment.

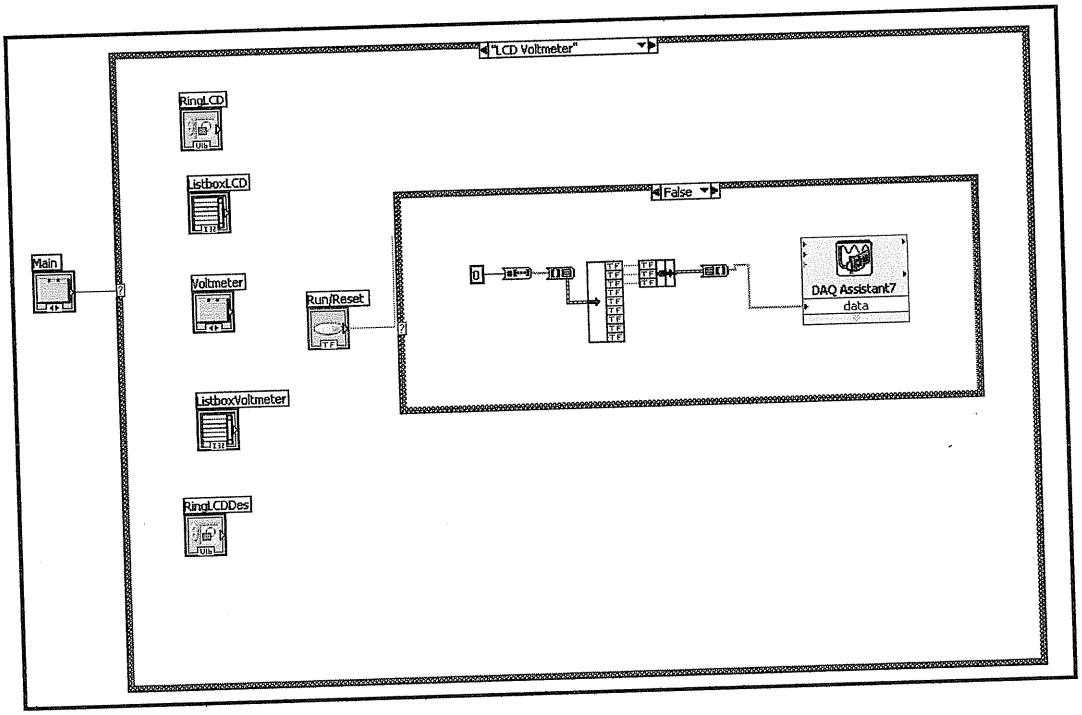


Figure 5.20: LCD Voltmeter Block Diagram

- **Music Tones Generation experiment**

When the button is pressed (Run case) the internal case structure will send 2 to the DAQ which will cause to execute Music Tones Generation experiment, on the other hand, when the button is not pressed (Reset case) the internal case structure will send 0 to the DAQ which will cause to reset this experiment.

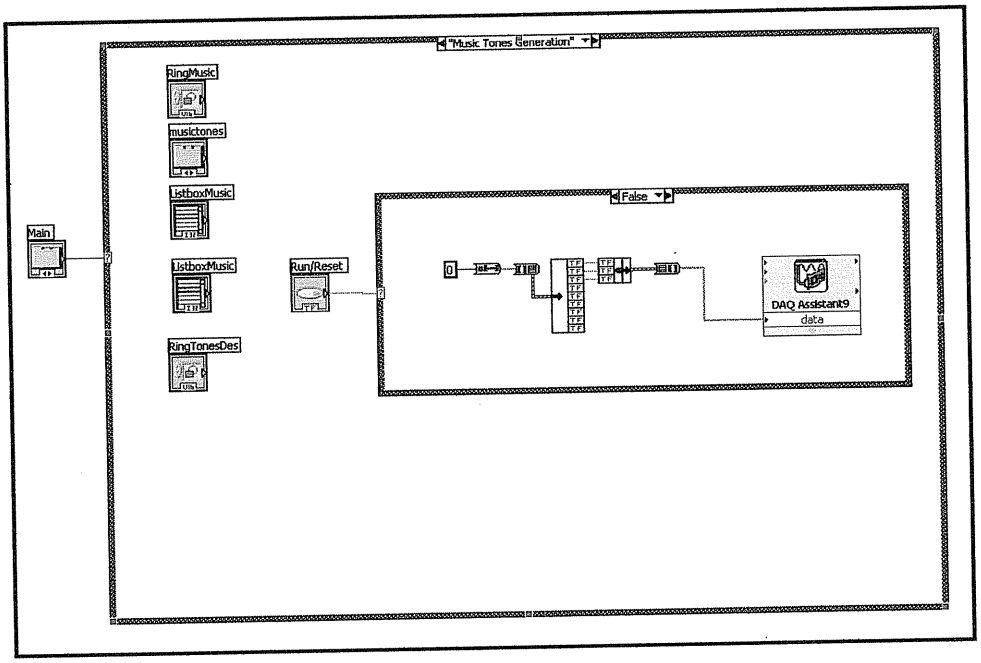


Figure 5.21: Music Tones Generation Block Diagram

5.3.3 Programmer Interfaces

This is the description front panel of the parallel Programmer; it gives the user a brief description about how it works and which software products it supports.

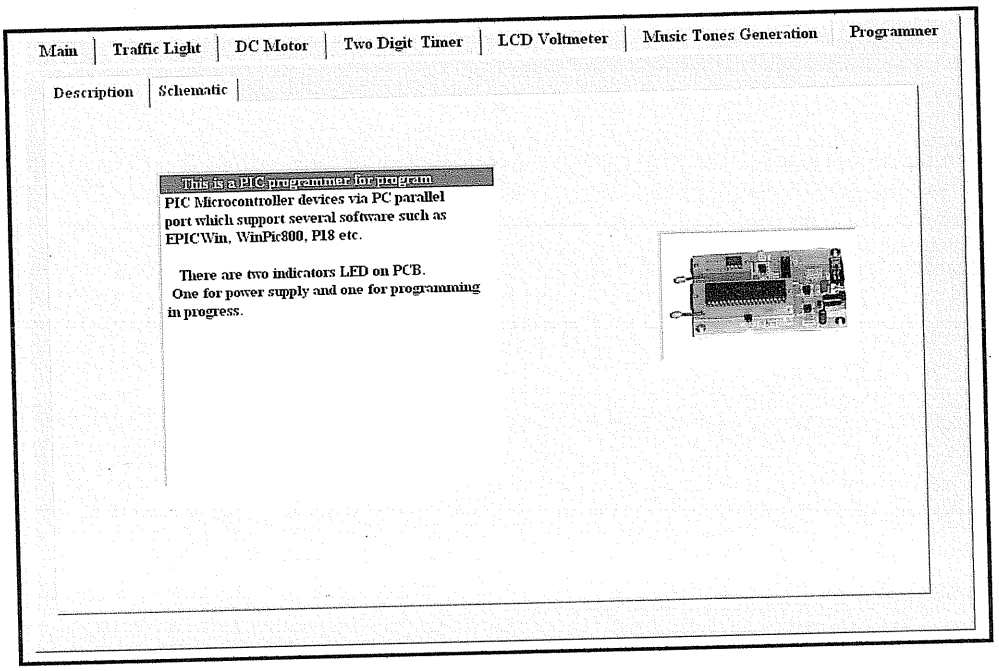


Figure 5.22:Parallel Programmer Description Front Panel

The following figure shows the schematic of the Programmer which is drawn in Orcad to help the user to understand the internal connections and interfaces of this programmer.

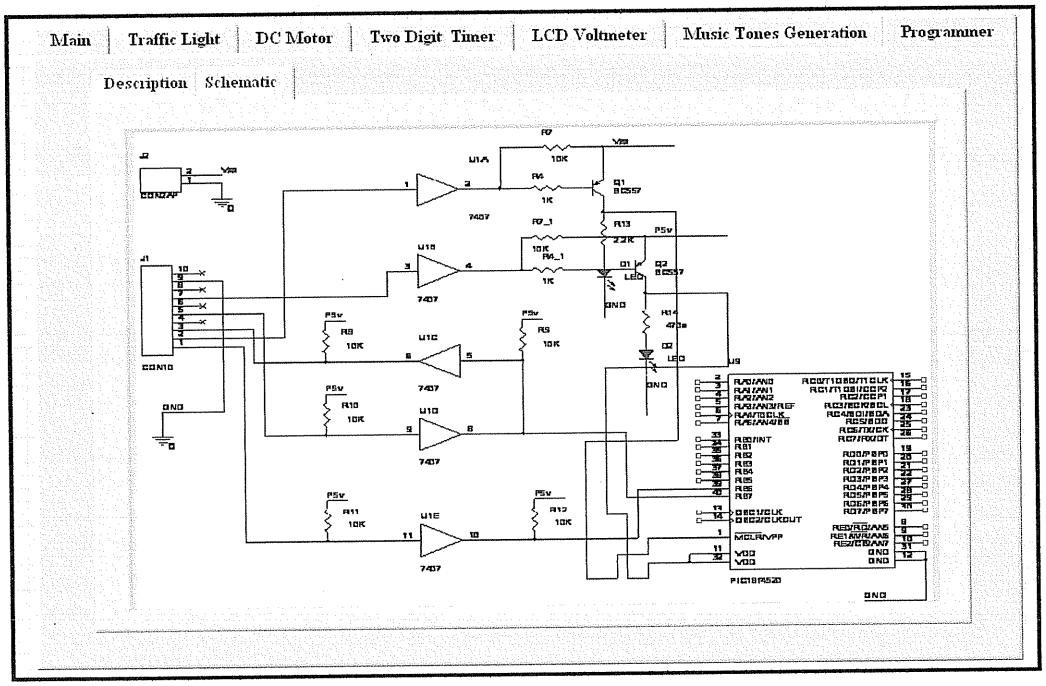


Figure 5.23: Parallel Programmer Schematic Front Panel

5.4 System Flowcharts

This section discusses the flowcharts of the system, including all its experiments to provide a detailed sequence of the system operations.

5.4.1 General System Flowchart

In the following flowchart, the general system flowchart will be discussed.

First of all the user will choose the required operation to be done from the available options in LabVIEW's designed interface, so he/she may choose to run a specific experiment or to reset the system.

In both cases, the data will be sent through DAQ to perform user's choice according to the values of RE0, RE1, and RE2 then the result will be visible to him.

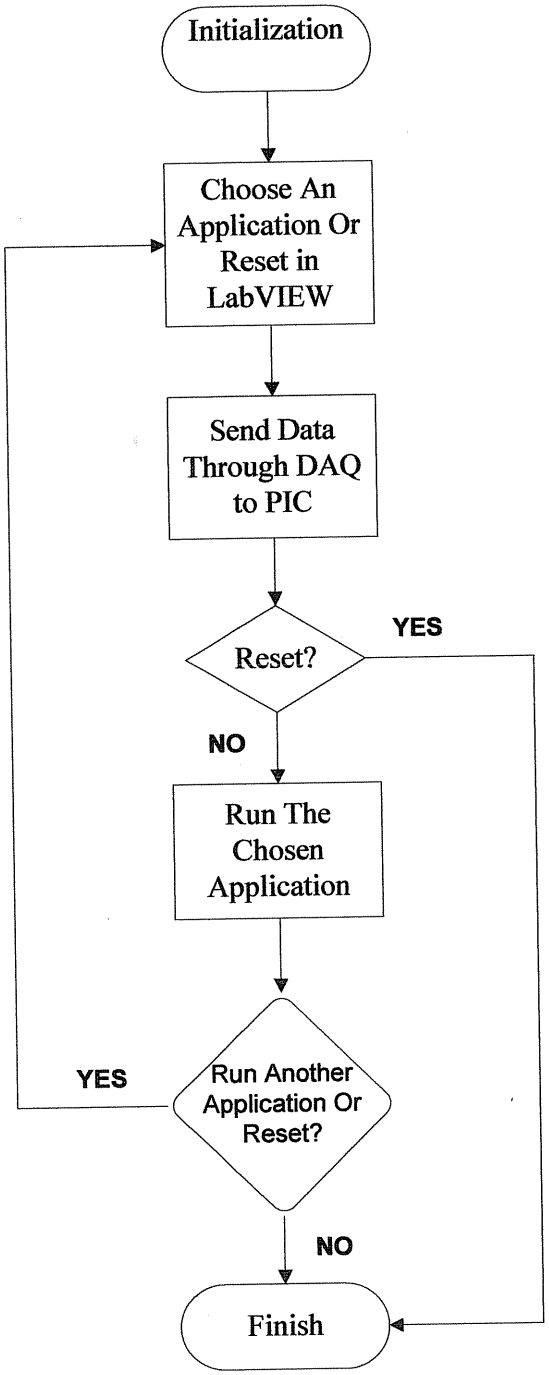


Figure 5.24: General System Flowchart

5.4.2 Traffic Light Flowchart

In the following flowchart, the operation of traffic light experiment will be discussed, when the user select traffic light experiment from LabVIEW, data that carries RE0=1, RE1=0 and RE2=0 will be passed through DAQ to specify this experiment, then it will be run and this process can be repeated.

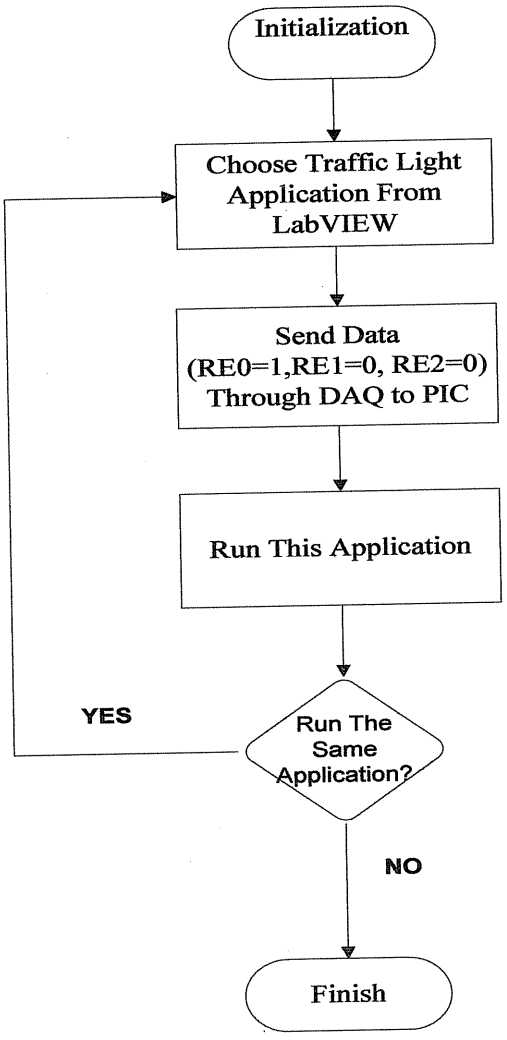


Figure5.25: Traffic Light Flowchart

5.4.3 Music Tones Generation Flowchart

In the following flowchart, the operation of Music Tones Generation will be discussed, when the user select this experiment from LabVIEW, data that carries RE0=0, RE1=1 and RE2=0 will be passed through DAQ to specify this experiment, then it will be run and this process can be repeated.

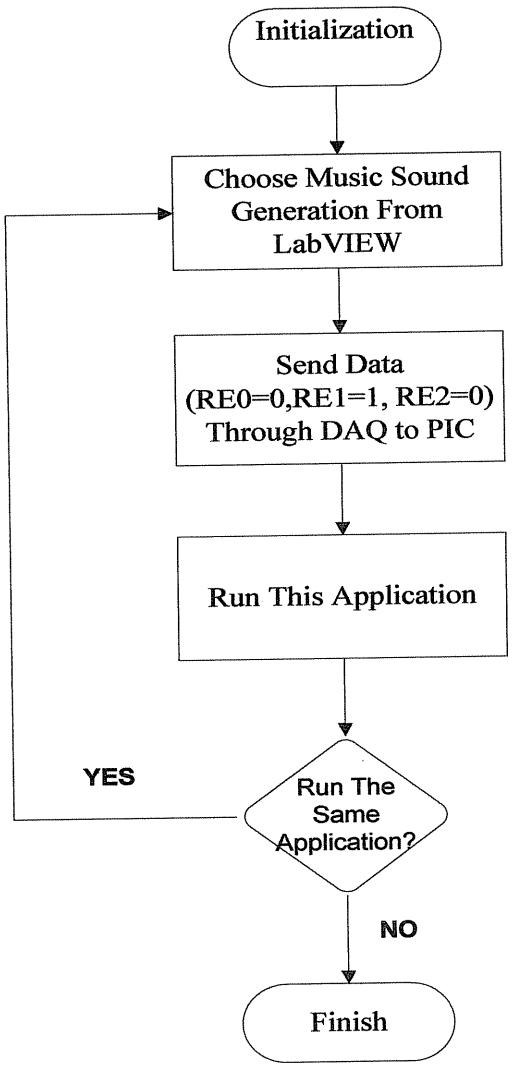


Figure 5.26: Music Tones Generation Flowchart

5.4.4 Two Digit Timer Flowchart

In the following flowchart, the operation of Two Digit Decimal Timer will be discussed, when the user select this experiment from LabVIEW, data that carries RE0=1, RE1=1 and RE2=0 will be passed through DAQ to specify this experiment, then it will be run and this process can be repeated.

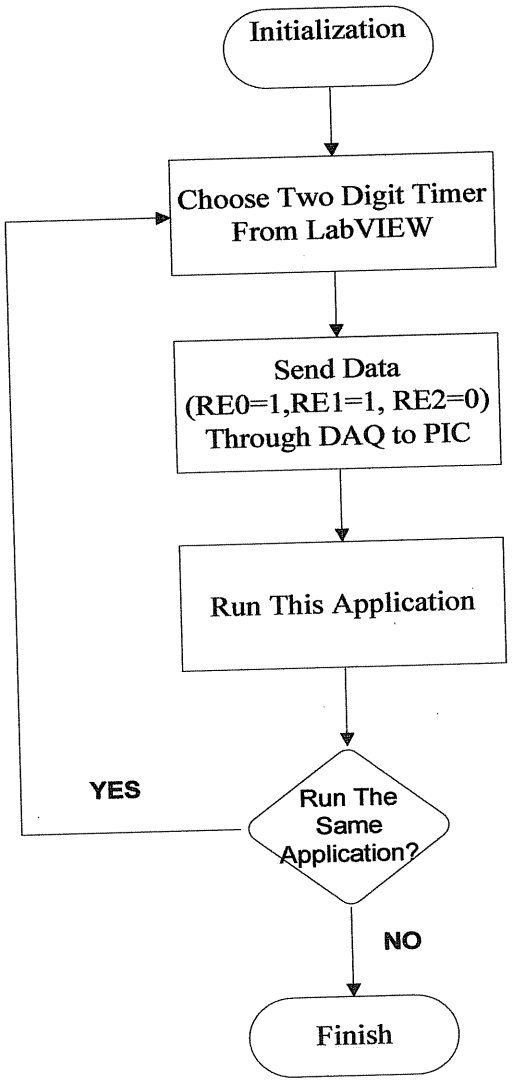


Figure 5.27: Two Digit Timer Flowchart

5.4.5 DC Motor Flowchart

In the following flowchart, the operation of DC Motor will be discussed, When the user select this experiment from LabVIEW, data that carries RE0=1, RE1=0 and RE2=1 will be passed through DAQ to specify this experiment, then it will be run and this process can be repeated.

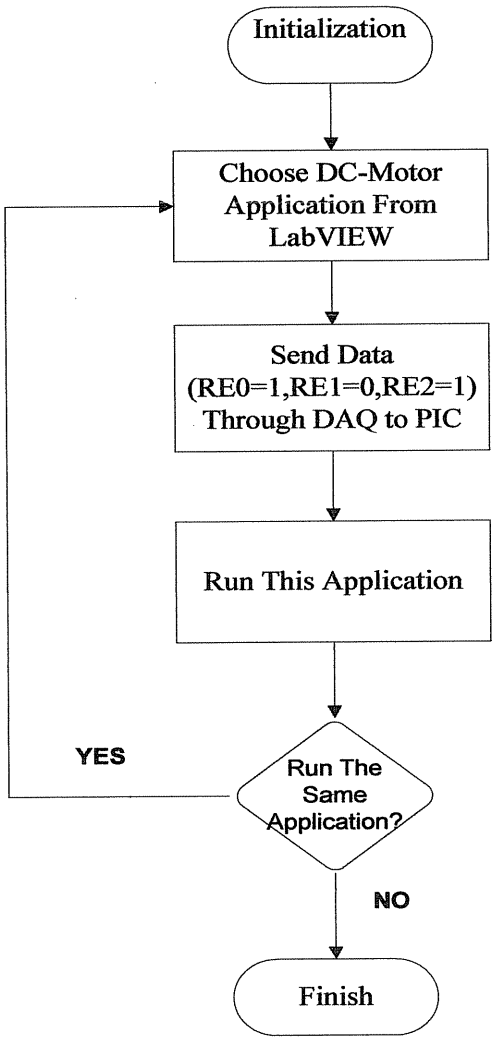


Figure 5.28: DC Motor Flowchart

5.4.6 LCD Voltmeter Flowchart

In the following flowchart, the operation of LCD Voltmeter will be discussed, when the user select this experiment from LabVIEW, data that carries RE0=0, RE1=0 and RE2=1 will be passed through DAQ to specify this experiment, then it will be run and this process can be repeated.

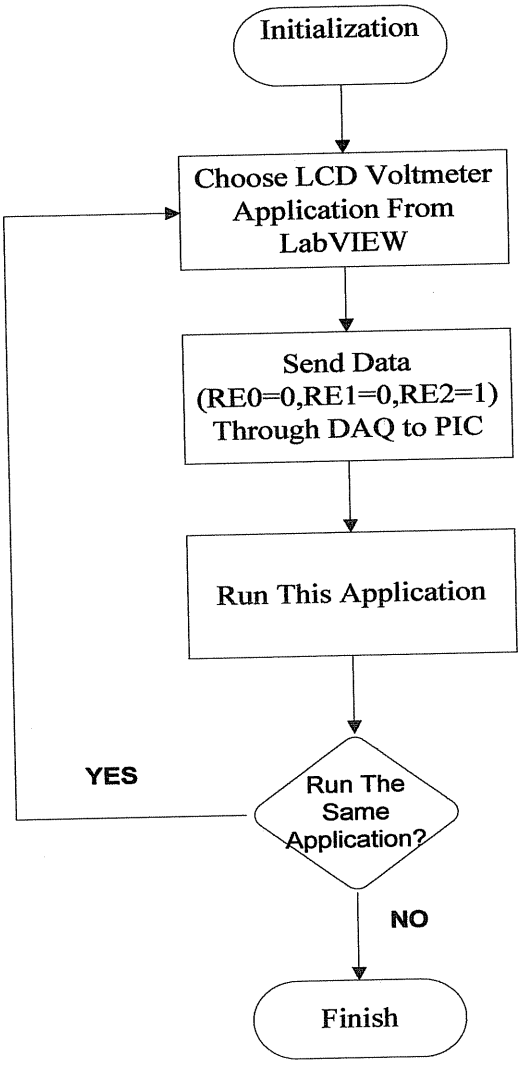


Figure 5.29: LCD Voltmeter Flowchart

5.4.7 System Reset Flowchart

In the following flowchart, the operation of System Reset will be discussed, When the user select this process from LabVIEW, data that carries RE0=0, RE1=0 and RE2=0 will be passed through DAQ to specify this experiment, then it will be applied in order to stop the last executed experiment.

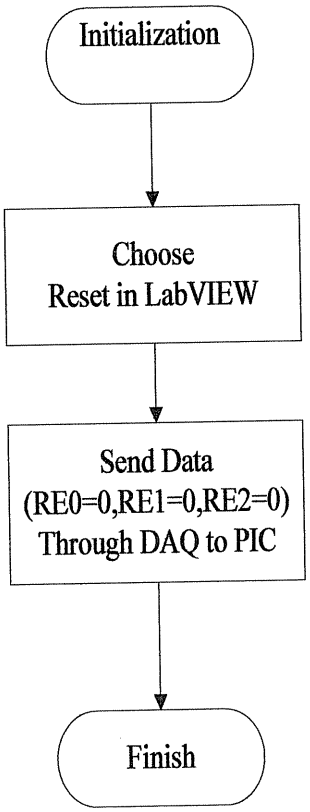


Figure 5.30: Reset Flowchart

5.5 Algorithms and Pseudocode

Main Algorithm

begin

SET internal oscillator to 4MHz
INIT timers
INIT ADC
INIT PORTE (RE0, RE1, RE2) as input

WHILE (system is running)

READ the value of PORTE // determine which experiment will be executed

CASE PORTE OF

0 : Reset the system
1 : RUN Traffic light experiment
2 : RUN Music Tones Generation experiment
3 : RUN Two digit timer experiment
4 : RUN LCD voltmeter experiment
5 : RUN DC Motor experiment

ENDCASE

ENDWHILE

end

- **Reset Algorithm**

Begin

SET PORTD to 0b xx000000 // disable traffic light experiment
SET RD6, RD7 to one // disable two digit timer experiment
SET RC2 to zero // disable sound generation experiment
CLEAR LCD // disable LCD voltmeter experiment
SET RC2 to zero // disable DC motor experiment

end

- **Traffic Light Algorithm**

Begin

// Disable all experiments
CALL Reset function
INIT PORTD (RD0 to RD5) as output // connected with LEDs

//first case of traffic light sequence
TURN ON the red LED from set1 and the green LED from set2
CALL delay for 15s

//second case of traffic light sequence
TURN ON the red LED from set1 and the yellow LED from set2
CALL delay for 2s

//third case of traffic light sequence
TURN ON the red & yellow LEDs from set1 and the red LED from set2
CALL delay for 2s

//fourth case of traffic light sequence
TURN ON the green LED from set1 and the red LED from set2
CALL delay for 15s

//fifth case of traffic light sequence
TURN ON the yellow LED from set1 and the red LED from set2
CALL delay for 2s

//sixth case of traffic light sequence
TURN ON the red LED from set1 and the red & yellow LEDs from set2
CALL delay for 2s

end

- **Sound Generation Algorithm**

Begin

// Disable all experiments
CALL Reset function
INIT RC5, RC6, and RC8 as input // used for the switches of the tones

// RC4 outputs square wave signal for each tone, generated using timer1
INIT RC2 and RC4 as output
SET RC4 to zero // enable encoder

CASE RC5, RC6, RC4 OF
000 : no tone generated
001 : DOH tone
010 : RAY tone
011 : ME tone
100 : FAH tone
101 : SOH tone
110 : LAH tone
111 : TE tone
ENDCASE

end

- **Two Digit Timer Algorithm**

begin

// Disable all experiment
CALL Reset function

// PORTB used to send ASCII code of the number to the seven segments
CONFIGURE PORTB as output
CLEAR timer

```
// read the status of clear and increment switches
INIT RD6, RD7 as input

    IF      RD6==0    THEN          // if clear button is pressed
            CLEAR timer
    ELSE
            Do nothing
    ENDIF

    IF      RD7==0    THEN          // if increment button is pressed
            INCREMENT timer
    ELSE
            Do nothing
    ENDIF

// enable both seven segments one after the other to display data
CONFIGURE RD6, RD7 as output
CONVERT count from hexadecimal to decimal

SET RD6 to zero             // enable the first seven segment
// find ASCII code for the first digit of the count
Display ASCII code of the timer count

CALL Delay for 5µs
SET RD7 to zero             // enable the second seven segment
// find ASCII code for the first digit of the count
Display ASCII code of the timer count

end
```

- **LCD Voltmeter Algorithm**

begin

// Disable all experiments
CALL Reset function
INIT RA0 as input // input to the analog signal
INIT RA1 to RA7 as output // to control and send data to LCD
INIT LCD
INIT ADC
READ the result from ADC
DISPLAY the result on LCD
end

- **DC Motor Algorithm**

begin

// Disable all experiments
CALL Reset function
INIT timer2 to be used for PWM
INIT RC3 as input // to get the direction of rotation
INIT RC0, RC1 as output
ACTIVATE timer2 to generate 20 kHz signal
READ the direction of rotation
SEND direction of rotation on RC0
SEND speed of rotation on RC1

end

Chapter Six

Implementation and Testing

Chapter Six
Implementation and Testing

6.1 Introduction

This chapter demonstrates the procedures used to test and examine the basic system subroutines. System testing is an important and crucial step in implementing a system. It senses the effectiveness of that system just before introducing it to the users.

Testing will include the components testing in which each component will be tested separately, subsystems testing and finally the integrated system testing, that contains system simulation.

6.2 Component Testing

6.2.1 LEDs Testing

Testing LED is done by building the circuit in Figure 6.1, that will check if the LED works correctly or not.

- **Testing Steps**

1. Connect one terminal of the 150 Ω resistor in series with 5 volt power supply, and the other terminal of the resistor with the LED anode side.
2. Connect the cathode side of the LED with ground of 0 volt.
3. If the LED is on; then it works correctly, otherwise it doesn't work correctly.

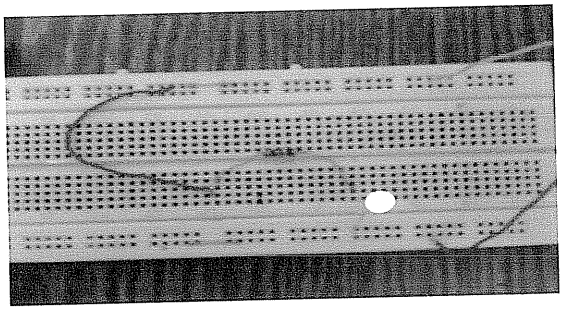


Figure 6.1: LED Testing

6.2.2 Diodes Testing

Analogue multimeter is still quite easy to perform a simple go / no-go test, this is performed a test that gives a very quick indication of whether the diode works properly or not.

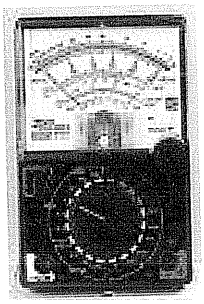


Figure 6.2: Analogue Multimeter

Just two tests are needed with the multimeter to ensure that the diode works satisfactorily.

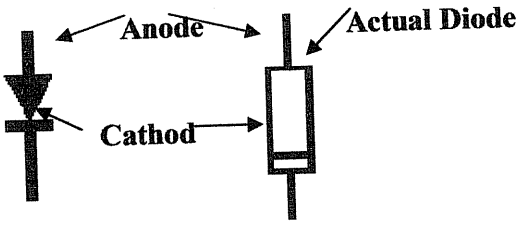


Figure 6.3: Diode Circuit Representation

- **Testing Steps of Diode**

1. Set the Multimeter to its ohms range - any range can be used.
2. Connect the cathode terminal of the diode to the positive terminal marked on the multimeter, and the anode to the negative or common terminal.
3. Set the Multimeter to read ohms, and a low reading should be obtained.
4. Reverse the connections.
5. This time a high resistance reading should be obtained

6.2.3 Transistor Testing

Again the test using a multimeter gives a very quick indication of whether the transistor is basically operational.

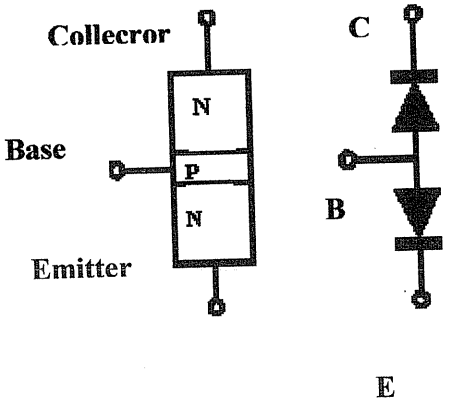


Figure 6.4: NPN Transistor

The test relies on the fact that a transistor can be considered to comprise of two back to back diodes, and by performing the diode test between the base and collector and the base and emitter of the transistor using an analogue multimeter, the basic integrity of the transistor can be ascertained.

- **Steps for Transistor (NPN)**

1. Set the Multimeter to its ohms range - any range can be used.
2. Connect the base terminal of the transistor to the terminal marked positive on the Multimeter.

3. Connect the terminal marked negative or common to the collector and measure the resistance, it should read open circuit (there should be a deflection for a PNP transistor).
4. With the terminal marked negative still connected to the base, repeat the measurement with the positive terminal connected to the emitter, the reading should again read open circuit (the multimeter should deflect for a PNP transistor).
5. Reverse the connection to the base of the transistor, this time connecting the negative or common terminal of the analogue test meter to the base of the transistor.
6. Connect the terminal marked positive, first to the collector and measure the resistance. Then take it to the emitter. In both cases the Multimeter should deflect (indicate open circuit for a PNP transistor).
7. Connect the Multimeter negative or common to the collector and Multimeter positive to the emitter. Check that the Multimeter reads open circuit. (The meter should read open circuit for both NPN and PNP types.

8. Reverse the connections so that the Multimeter negative or common is connected to the emitter and Multimeter positive to the collector. Check again that the Multimeter reads open circuit.

6.2.4 Speaker Testing

Using Function Generator Instrument that generates a square wave of specific frequency to test Speaker.

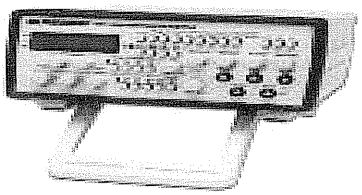


Figure 6.5: Function Generator

- **Testing Steps**

1. Prepare function generator operation that will generate square wave.
2. Precise function generator on a specific frequency from 50 Hz to 600 Hz.
3. Then connect speaker terminals to function generator terminals.
4. If there is a sound is heard then speaker works properly, this shown in Figure 6.6.

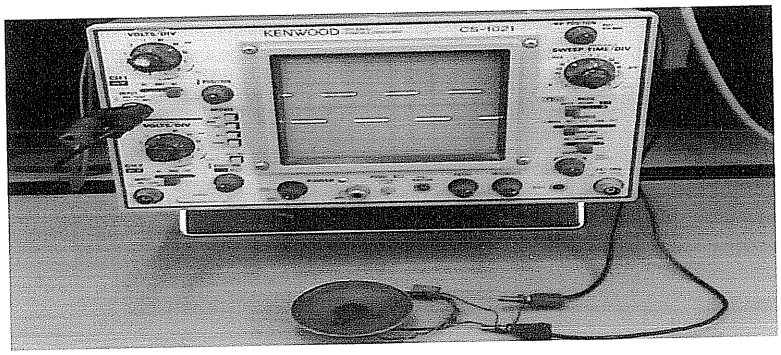


Figure 6.6: Speaker Testing

6.2.5 74LS148 Encoder Testing

74LS148 Encoder is a TTL encoder, feature priority at the input to insure that only the highest –order data lines is encoded, it encodes eight data lines to three

binary lines (octal) all these line are active low, testing this encoder shown in Figure 6.7.

- **Testing Steps**

1. Connect Vcc pin to power supply of 5 volts, and Vss to ground of 0 volt.
2. Connect all encoder inputs to ground (0 volt), since they are active low.
3. Then connect the third output of the encoder to cathode side of the LED.
4. Connect the anode side of the LED to the power supply.
5. If the LED is on then it works properly.

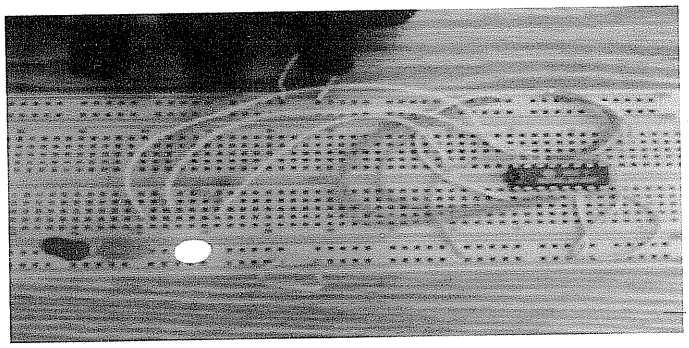


Figure 6.7: 74LS148 Encoder Testing

6.2.6 DC Motor Testing

Using power supply (5 volts) or any battery, to test the motion of DC Motor, As shown in Figure 6.8.

- **Testing Steps**

1. Connect the positive side of battery to one terminal of the DC motor.
2. Connect the negative side of the battery to the other motor terminal.
3. Then motor will move forward.
4. Then swap the battery leads the motor will move in reverse, the frequency that operates this DC motor is shown in the oscilloscope.

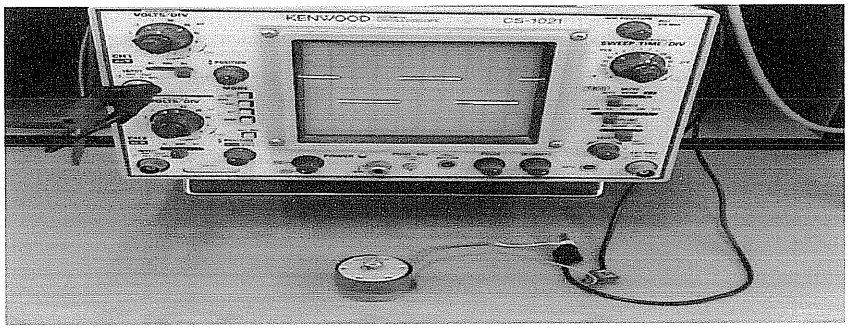


Figure 6.8: DC Motor Testing

62.7 H-Bridge Testing

H-bridge circuit is used to test H-bridge component, as shown in Figure 6.9, which help us to understand the operation of H-bridge, control A and B in H-bridge determine if it works correctly or not.

- **Testing Steps**

1. Generate a square wave using Function Generator.
2. Build H-bridge circuit as shown in Figure 6.9.
3. Connect one terminal of Function Generator to the input A, and the other to input B.
4. Then the motor should run in one direction.
5. Swap the inputs A and B, then motor should run in the opposite direction, if it does not run, or if it runs very slowly, and then it doesn't work properly.

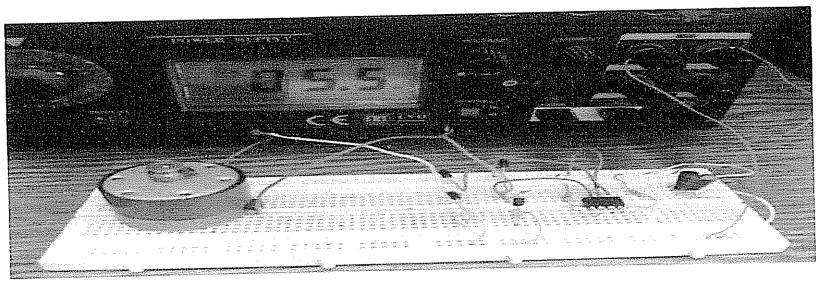


Figure 6.9: H-bridge Testing

6.2.8 LCD Testing

Testing LCD Display done in an easy way to determine if it works correctly or not as shown in Figure 6.10.

- **Testing Steps**

1. Connect Vcc of LCD to the power supply of 5 volts.
2. Connect Vss of LCD to the ground of 0 volt.
3. If the LCD is on, then it works correctly.

Figure 6.10: LCD Testing

6.2.9 Seven Segment Testing

Determine if the seven segment display work correctly or not by building the circuit in the Figure 6.11, then record its readings to determine if the result is correct or not.

- **Testing Steps**

1. Connect pin 3 to power supply since it is a common anode.
2. Connect all inputs in series with resistors.
3. Connect b, and c pins to ground (since they are active low).
4. Connect the other to power supply.
5. Then the result will be displayed on the seven segment number 1, if not it doesn't work correctly.

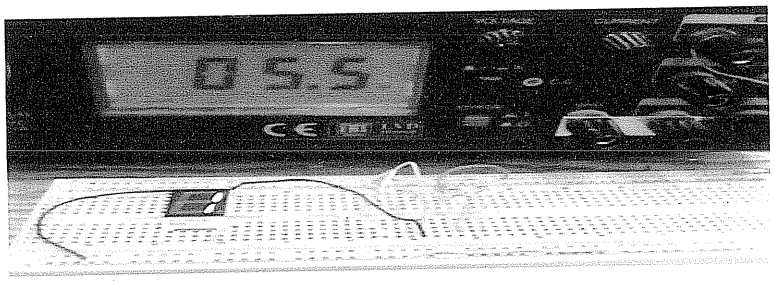


Figure 6.11: Seven Segment Testing

6.2.10 PIC18F4520 Testing

6.2.10.1 PIC18F4520 Ports Testing

Testing PIC Ports done by writing a simple code that tests if the ports works correctly or not.

Using the following code, the output will be all ones in all ports if an input pin is 0, as shown in Figure 6.12, RB0 is an input connected to switch, all other pins are work as output and connected to LEDs if all LEDs are on then it performed correctly.

- **Testing Code**

```
void main (void)
{
        TRISA=0;
        TRISB=0x01;
    TRISC=0;
    TRISD=0;
    TRISE=0;
    While(1)
    {
        if(PORTBbits.RB0==0)
                    {
                            PORTA=255;
                            PORTB=254;
                            PORTC=255;
                            PORTD=255;
                            PORTE=15;
                }
            }
    }
```

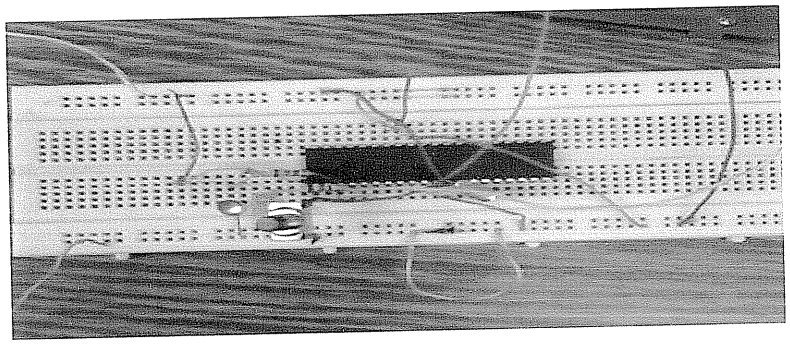


Figure 6.12: PIC18F4520 Ports Testing

6.2.10.2 PIC18F4520 Internal Oscillator Testing

PIC18F4520 can be used either with external or internal oscillator, in this system internal oscillator is used of 4 MHz frequency for two reasons, the first one is to be suitable for the applications that are running depending on timers to give the required delays, the second one is to use less number of I\O lines; since using external oscillator will consume RA6 and RA7 which are needed to be used as I/O lines in the system.

To test internal oscillator writing a specific code, this code use instructions that allows using internal oscillator, also allows viewing the frequency of this oscillator in both pins RA6 and RA7, in this code the system can determine specific frequency to use in the range 1 to 8 MHz (in this code using 4 MHz).

After loading the testing code on PIC18F4520, then connect V_{DD} to power supply, V_{SS} to ground, and RA6, RA7 to the terminals of oscilloscope to see the frequency, as shown in Figure 6.13.

As shown in Figure 6.13 the frequency of this square wave is 1 MHz (1/500ns), which is the value of the internal oscillator divided by 4.

- **Testing Code**

```
// use internal oscillator
#pragma config OSC = INTIO7
void main(void)
{
        //set internal clock to 4MHz, bit6, 5=1, bit4=0
        OSCCON=OSCCON | 0b01100000;
        OSCCON=OSCCON & 0b11101111;
}
```

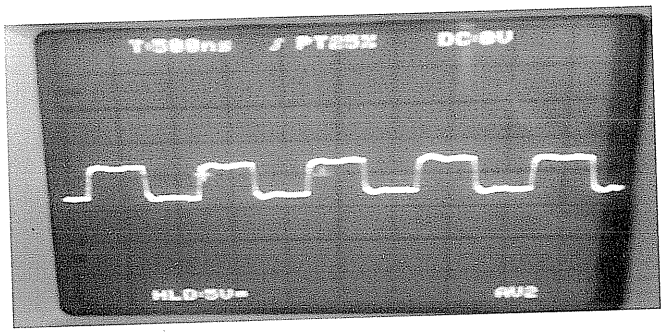


Figure 6.13: Internal Oscillator Frequency

6.3 Subsystem Testing

6.3.1 Parallel Programmer Testing

Testing this programmer is done by building the circuit shown in Figure 6.14, and connects PIC18F4520 to it, then run WinPIC800 software to load a simple code for testing.

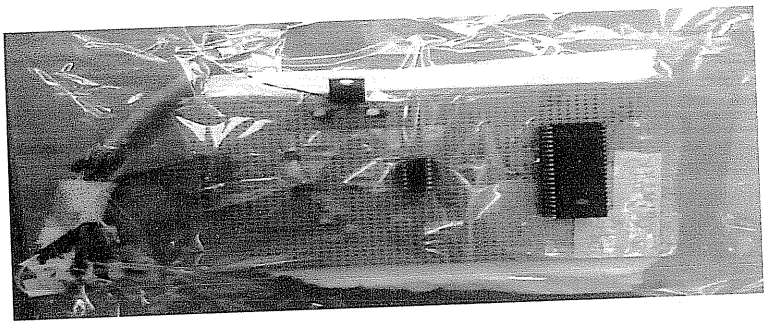


Figure 6.14: Parallel Programmer Testing

6.3.2 DAQ and PIC18F4520 Testing

Testing the operation of DAQ by sending specific values to PIC18F4520 from LabVIEW, that turns LEDs on or off as shown in the Figure 6.15.

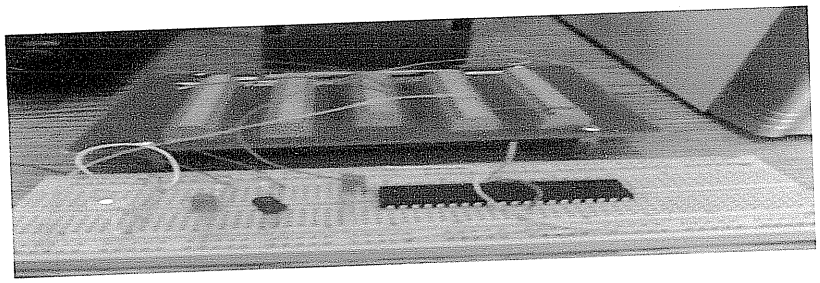


Figure 6.15: DAQ and PIC18F4520 Testing

6.3.3 Traffic Light Experiment

Testing this experiment depends on the circuit that is shown in Figure 4.9 That must be built, then the code is written and downloaded in order to test if this experiment works or not, testing code is shown in Appendix B.

This experiment uses timer0 to generate the required delays for each sequence of traffic light operation.

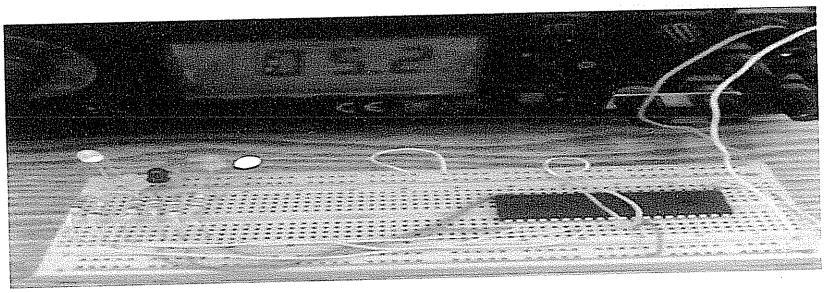


Figure 6.16: Traffic Light Experiment Testing

6.3.4 Music Tones Generation Experiment

Testing this experiment depends on the circuit that is shown in Figure 4.10 that must be built, then the code is written and downloaded in order to test if this experiment works or not, testing code is shown in Appendix B.

This experiment uses timer1 to generate the square wave for each tone.

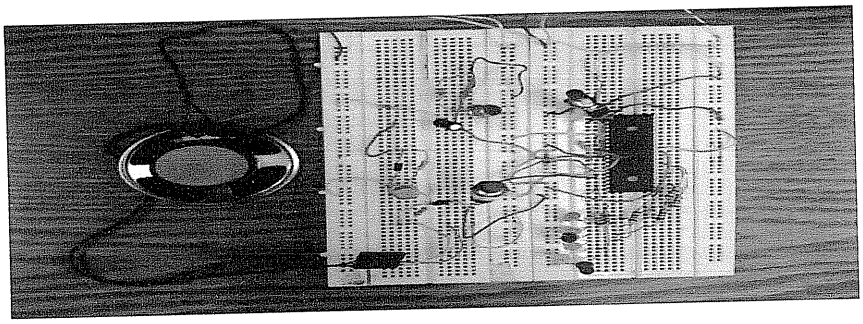


Figure 6.17: Music Tones Generation Experiment Testing

6.3.5 Two Digit Timer Experiment

Testing this experiment depends on the circuit that is shown in Figure 4.11 that must be built, then the code is written and downloaded in order to test if this experiment works or not, testing code is shown in Appendix B.

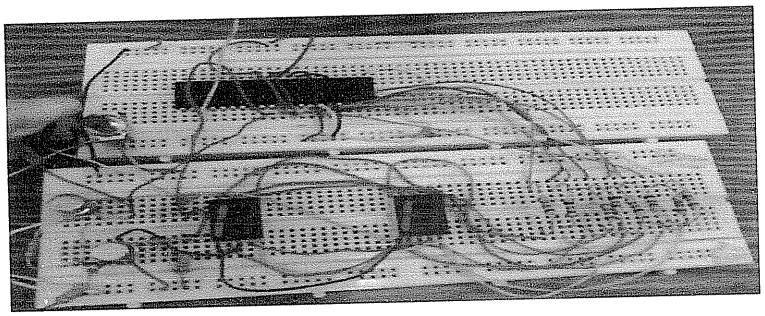


Figure 6.18: Two Digit Timer Experiment Testing

6.3.6 DC Motor Experiment

Timer2 can produce a timed sequence of pulses in (PWM) and this is a common approach used to control the torque of a DC motor, and some transistor driver circuits. These drivers will deliver enough current to control a small DC motor.

- **The system needs a 20kHz PWM period**

  - PIC18F4520 internal oscillator = 4 MHz thus Fosc/4 = 1MHz.
  - To have a full resolution (10-bit) we must choose the biggest value for the 8-bit period register.
  - If we choose 0xFF, the clock frequency will be divided by 256 to obtain the Timer2 frequency.
  - Then calculate the prescaler :

    (1000000/ 256) / 20000 = 0.1953 < 4
  - Then calculate if the result prescale is true or not, according to Timer2 counter register:

    (1000000/ 4) / 20000 = 12.5 < 256

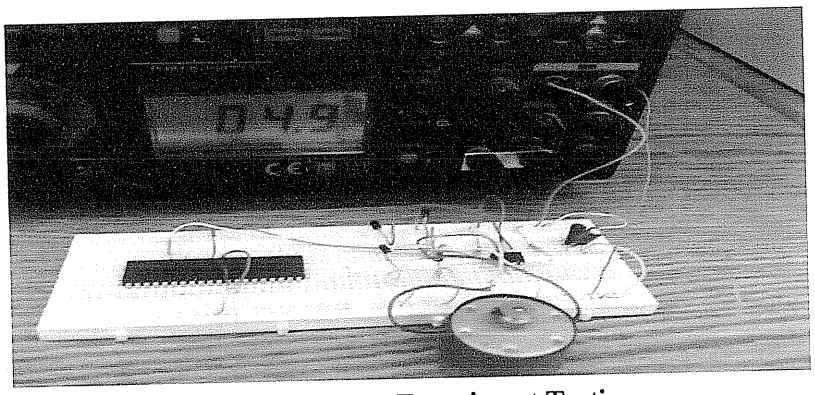


Figure 6.20: DC Motor Experiment Testing

6.3.7 LCD Voltmeter Experiment

Testing this experiment depends on the circuit that is shown in Figure 4.12 that must be built, then the code is written and downloaded in order to test if this experiment works or not, testing code is shown in Appendix B.

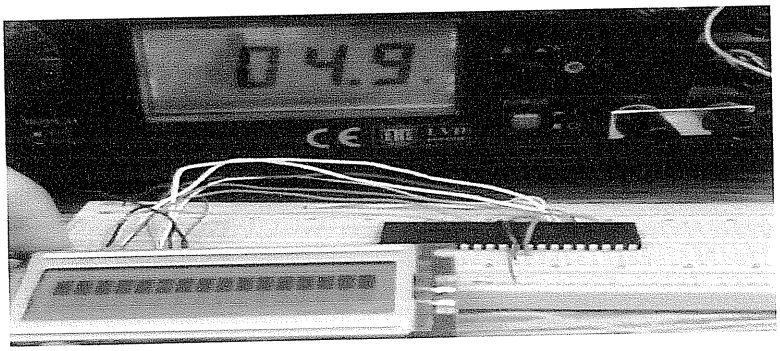


Figure 6.19: LCD Voltmeter Experiment

6.4 System Simulation

6.4.1 PIC18 Simulator IDE

It is a powerful application that supplies PIC18 developers with user-friendly graphical development environment for Windows with integrated simulator (emulator), Basic compiler, assembler, disassembler and debugger, which support system's PIC version.

This System uses PIC18 Simulator IDE for testing system experiments codes, since of its powerful features, for more details see appendix C.

6.4.2 Traffic Light Experiment Simulation

Using PIC18 Simulator IDE that simulates each code that is written to be loaded to PIC18, the system uses this program to test and generate the simulation copy of traffic light experiment.

As shown in Figure 6.22 the simulation figure contains the output of this experiment, which contains the true sequence of traffic light operation that is displayed on 8 x LED board and indicates that the result is true.

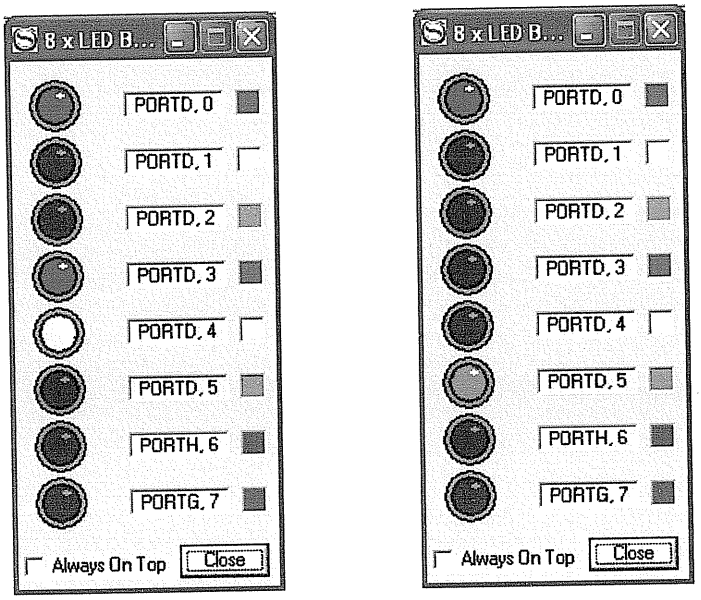


Figure 6.22: Traffic Light Experiment Simulation

6.4.3 Music Tones Generation Experiment Simulation

The system uses PIC18 Simulator IDE program to test and generate the simulation copy of music tones generation.

The simulation figure contains the output of this experiment, that indicates the square wave of each tone which is displayed on the Oscilloscope, Figure 6.23 shows the frequency of DOH tone, Figure 6.24 shows the frequency of FAH tone.

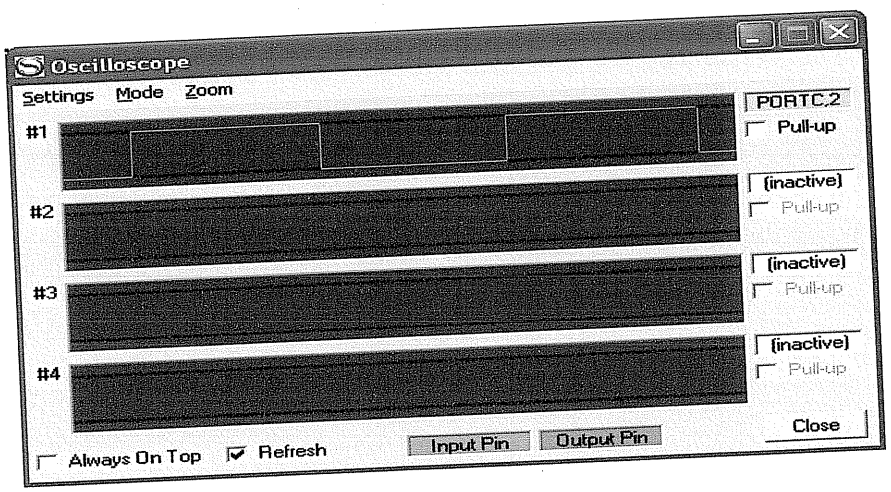


Figure 6.23: DOH Tone Simulation

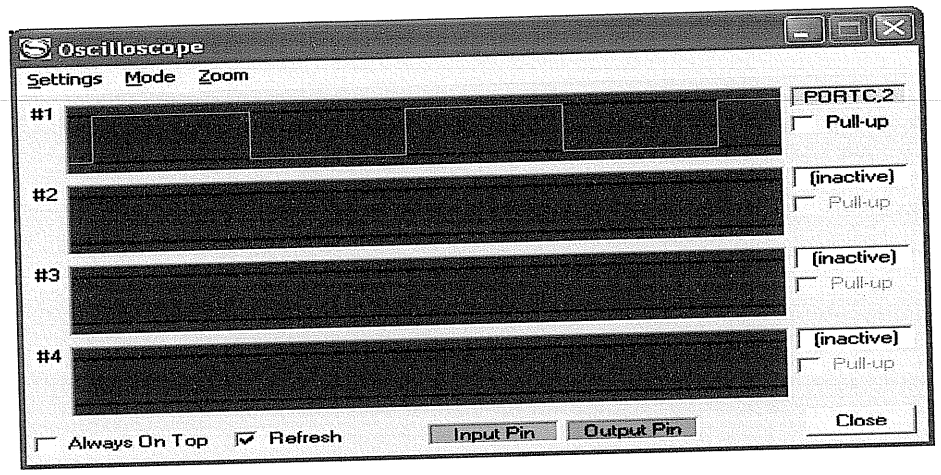


Figure 6.24: FAH Tone Simulation

6.4.4 Two Digit Timer Experiment Simulation

The system uses PIC18 Simulator IDE program to test and generate the simulation copy of timer that counts from 0.0 to 9.9.

The simulation figure contains the output of this experiment, that contains two seven segments which display the timer count, Figure 6.25 shows the status of timer when clear button is pressed, Figure 6.26 shows the timer count at 2.5 sec.

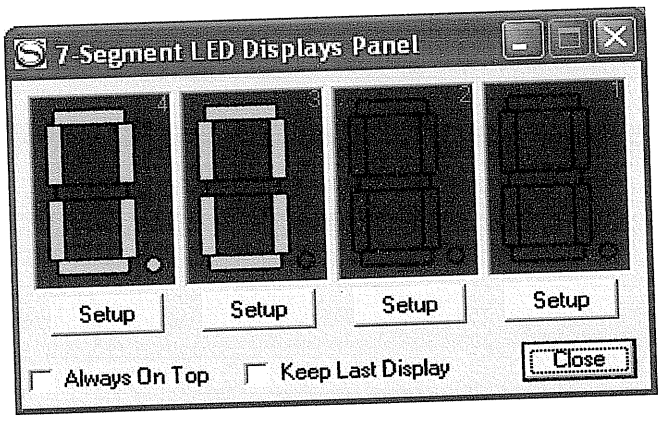


Figure 6.25: Clear Timer Simulation

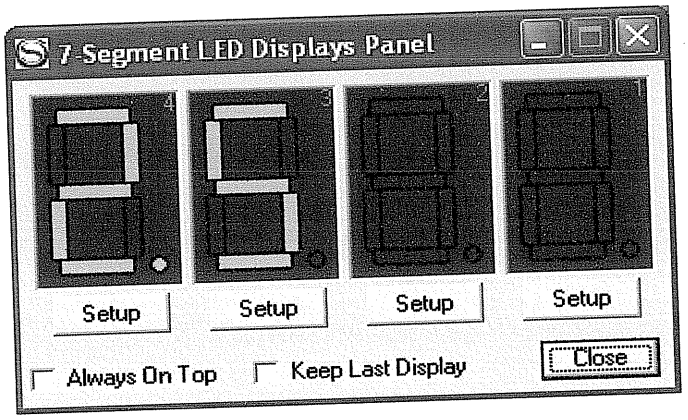


Figure 6.26: Timer Simulation

6.4.5 DC Motor Experiment Simulation

The system uses PIC18 Simulator IDE program to test and generate the simulation copy of DC Motor.

The simulation figure contains the output of this experiment, which contains the PWM signal, which is displayed on the oscilloscope; Figure 6.27 shows the square wave of PWM signal of frequency 20 KHz.

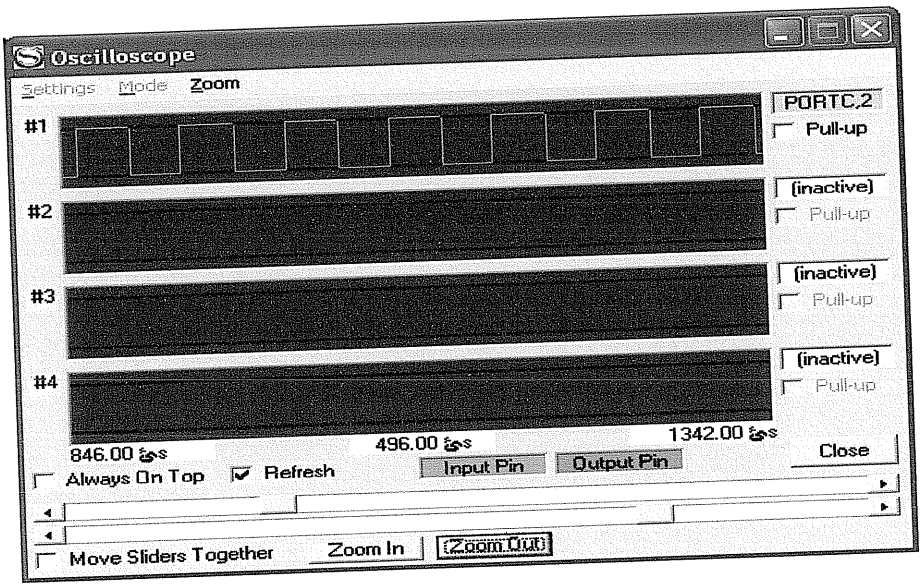


Figure 6.27: DC Motor Simulation

Chapter Seven

Conclusions and Future Works

Chapter Seven

Conclusions and Future Works

7.1 Introduction

This chapter introduces some significant points about the way of continuing do more and more in the field of the system concepts or tools. Also, it represents the conclusions extracted during designing and implementing it. The chapter illustrates the system implementation achievements and output.

7.2 Conclusions

Many experiences were added to the team cognitive knowledge through this project. Many conclusions can be stated here, but only significant and important ones are described here:

- In this project, we implement a complete training environment of PIC microcontroller under the control of LabVIEW-based GUI. We built several experiments to reach our goal which are traffic light, DC Motor, Two digit timer, Music Tones Generation and LCD voltmeter. Also, we create a complete LabVIEW environment with suitable GUI to control these experiments by run or reset each of them. So the integrated system will provide the student with a complete PIC environment to execute experiments.

- We learn how to combine all the knowledge that we have taken in the previous years in a practical way to help us in solving problems in our project.

- How to work as a team is a great principle that we learn in order to success in any project.

- For programming thePIC18f4520 microcontroller, we used MPLAB IDE program with MPLAB ICD2 debugging and programming device.

- The microcontroller can be programmed in different languages using MPLAB IDE, the language we used is C++. So all programs are written in C++ and downloaded on the microcontroller using the ICD2 connector.

- How to solve problems, is also an important side in this project, as the MPLAB ICD2 in-circuit debugger was damaged and no way to get another one in limited time, we decided to implement a programming circuit instead of this damaged tool.

- Also, how to combine hardware with software in order to implement an embedded system is a good idea to develop different applications.

- LabVIEW, is a powerful program that can help students in performing many applications in an easy way.

7.3 Problems

System completion in regard to its objectives is an implementation dependent issue. Problems are natural things. Skipping these problems is a success. No degradation affects the system if problems appear. Here are problems faced the project team during the system implementation:

Hardware Problems

- Delivering parts is the first main problem since PIC15F4520 was not available in Hebron, not a little time was wasted while waiting them.

- Damaging of the MPLAB ICD2 In-Circuit debugger was a big problem that we faced, since no spare on was available in the university, and the security rules of Israel prevented any company there to sell this tool to any Palestinian organization and because of time limitation, we can't wait 8 weeks till it arrives!

- Limitations in the number of the available DAQ lines, forced us to reduce the number of experiments also made the feedback operation too hard.

Software Problems

- In PIC 18F4520 programming a lot of considerations should be taken in using the analog and digital ports, the values given to special purpose registers and others. So during writing and running the

programs on the microcontroller we faced some initialization and declaration problems.

- In LabVIEW, we faced many problems such as making events, using multiple DAQs in the same environment… All this problems required an additional effort from us in order to solve them.

- LabVIEW as a laboratory virtual instrument doesn't have the ability to implement a programmer, since it doesn't support the implementation of some major components such as linker and loader.

7.4 Future Works

If we had the opportunity to develop this project we would implement the following improvements:

- Increasing the number of experiments in our embedded system will improve this system greatly.

- Implemented a driver in LabVIEW that download programs directly on the PIC microcontroller we be noticeably different.

- Making a feedback system that takes the output of the experiments and displays them in LabVIEW using an advanced DAQ with more I/O lines, will be a good improvement.

References

[1] http://www.ni.com/LabVIEW

[2] **LabVIEW for Everyone: Graphical Programming Made Easy and Fun, Third Edition.**

[3] http://www.labviewexperts.com

[4] http://www.ni.com/DAQ

[5] http://www.microchip.com/downloads/en/DeviceDoc/51281d.pdf

[6] http://www.microchip.com/downloads/en/DeviceDoc/39631a.pdf

[7] http://www.robotroom.com/HBridge.html

Appendices

Appendix A: Terminologies and Abbreviations

Appendix B: Source Codes

Appendix C: Software Tools

Appendix D: Data Sheets

Appendix A

Terminologies and Abbreviations

Block diagram	Pictorial description or representation of a program or algorithm. It consists of executable icons called nodes and wires that carry data between the nodes, also it is the source code for the VI.
Front Panel	Interactive user interface of a VI, it's appearance imitates physical instruments such as oscilloscope and multimeters.
Data flow	Programming system that consists of executable nodes that execute only when they receive all required input data, and produce output automatically when they execute. LabVIEW is a data flow system.
NI-DAQ	Software driver included with all NI measurement devices. It is an extensive library of VIs and functions that you can call from an Application Development Environment (ADE), such as LabVIEW to program all the features of the NI measurement device, such as acquiring, configuring, generating and sending data from and to devices.
CBL	Computer-based Learning
VI	Virtual Instrument
NI	National Instruments
DAQ	Data Acquisition card

ADC	Analog to Digital Converter
TTL	Transistor-transistor Logic
Hz	Hertz
Sec	Second
LCD	Liquid Crystal Display
PWM	Pulse Width Modulation
PIC	Peripheral Interface Controller
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
IDE	Integrated Development Environment
ICD	In-circuit Debugger
DC	Direct Current

Appendix B

Source Code

// Main File Code

```
#include <p18f4520.h>
#include <adc.h>
#include <timers.h>
#include <applications.h>
// use internal oscillator
#pragma config OSC = INTIO67

//MCLR pin enabled; RE3 input pin disabled
#pragma config MCLRE = ON

//no use of debugger RB6 and RB7 configured as general purpose I/O pins
#pragma config DEBUG = OFF

//no Watchdog, no low voltage programming
#pragma config WDT = OFF
#pragma config LVP = OFF

//PORTB<4:0> pins are configured as digital I/O on Reset
#pragma config PBADEN=OFF


void main(void)
{
        int app=0;
        int d;
        //set internal clock to 4MHz , bit6,5=1,bit4=0
        OSCCONbits.IRCF0=0;
        OSCCONbits.IRCF1=1;
        OSCCONbits.IRCF2=1;
        ADCON1=0x0E;
        TIMR0_Initialize();
        TIMR2_Initialize();
        TIMR1_Initialize();
        ADC_Initialize();
        PORTE_Initialize();

        while(1)
        {
                // determine which application will be executed.
                app=PORTE && 0b00000111;

                switch(app)
                        {
                        // disable all apllications
                        case 0:
```

```
{
        Disable_Applications();
        break;
}
// traffic light
case 1:
        {
                Traffic_Light_Initialize();
                Traffic_Light();
                break;
        }

// sound generation music tones
case 2:
        {
                Sound_Generation_Initialize();
                Sound_Generation();
                break;
        }

// 0.0 - 9.9 second timer
case 3:
        {
                Timer_Initialize();
                Timer();
                break;
        }

// LCD voltmeter
case 4:
        {
                Volt_LCD_Initialize();
                ADC();
                break;
        }

// DC motor
case 5:
        {
                DC_Motor_Initialize();
                d=PORTCbits.RC3;
                if(d==0)
                        // move Backward
                        DC_Motor(100,'B');
                else
                        if(d==1)
                        // move forward
                        DC_Motor(100,'F');
```

```
                                        break;
                                    }


                }// end switch.
        }// end while loop
CloseADC(); // Disable A/D converter
CloseTimer0(); // Disable timer 0
CloseTimer1(); // Disable timer 1
CloseTimer2(); // Disable timer 2
}


// Application File Code


#include <p18f4520.h>
#include <delays.h>
#include <timers.h>
#include <pwm.h>
#include <xlcd.h>
#include <adc.h>

void Disable_Applications(void)
        {
                //disable traffic light application;
                // disable seven segmants of timer application.
                PORTD=0b11000000;

                //disable sound generation application
                PORTCbits.RC4=1;

                //disable LCD
                OpenXLCD(FOUR_BIT & LINES_5X7);
                WriteCmdXLCD(DOFF);

                //disable DC motor
                PORTCbits.RC0=0;
                PORTCbits.RC1=0;
                TRISCbits.TRISC3=0;

        }

  void Delay(int value)
        {
                WriteTimer0(value);
                while(INTCONbits.T0IF==0);
```

```
                INTCONbits.T0IF=0;
        }

void Delay2(int value)
        {
                WriteTimer1(value);
                while(PIR1bits.TMR1IF==0);
                PIR1bits.TMR1IF=0;
        }

void PORTE_Initialize(void)
        {

                TRISEbits.TRISE0=1;
                TRISEbits.TRISE1=1;
                TRISEbits.TRISE2=1;
        }

void TIMR0_Initialize(void)
        {
                OpenTimer0(
                TIMER_INT_OFF &
                T0_16BIT &
                T0_SOURCE_INT &
                T0_PS_1_256   );
        }


void TIMR2_Initialize(void)
        {
                OpenTimer2(
                TIMER_INT_OFF &
                T2_PS_1_1&                              // Prescale 1:4
                T2_POST_1_1  );
        }
void TIMR1_Initialize(void)
        {
                OpenTimer1(
                TIMER_INT_ON &
                T1_16BIT_RW &
                T1_PS_1_4       );
                // enable interupt when overflow.
                 //PIE1bits.TMR1IE=1;
        }
```

```
void Traffic_Light_Initialize(void)
        {
                // Disable all other applications
                Disable_Applications();

                //enable leds on PORTD(0-5)for traffic light application.
                // and disable all other ports.
                TRISD=TRISD && 0b11000000;

        }

 void Traffic_Light(void)
        {
                                        PORTD=0b11100001;
                                        Delay(6942);

                                        PORTD=0b11010001;
                                        Delay(57724);

                                        PORTD=0b11001011;
                                        Delay(57724);

                                        PORTD=0b11001100;
                                        Delay(6942);

                                        PORTD=0b11001010;
                                        Delay(57724);

                                        PORTD=0b11011001;
                                        Delay(57724);

        }

  void Sound_Generation_Initialize(void)
        {

                Disable_Applications();
                // initialize PORTC as input form switches
                // and one out put to speaker.
                TRISC=TRISC | 0b11100000;
                TRISC=TRISC & 0b11101011;
                PORTCbits.RC4=0;

        }


   void Sound_Generation(void)
        {
```

```
        int tone=0;
        tone=PORTC & 0b11100000;
switch(tone)
        {
        case 0:
                {
                while(tone==0)
                {
                        PORTCbits.RC2=0;
                        tone=PORTC & 0b11100000;
                        }
                        break;
                }

        // DOH note
        case 32:
                {
                while(tone==32)
                        {
                          PORTCbits.RC2=0;
                          Delay2(65059);
                         PORTCbits.RC2=1;
                           Delay2(65059);
                           tone=PORTC & 0b11100000;
                        }
                PORTCbits.RC2=0;
                break;
                }

        //RAY
        case 64:
                {
                while(tone==64)
                        {
                          PORTCbits.RC2=0;
                          Delay2(65111);
                         PORTCbits.RC2=1;
                           Delay2(65111);
                           tone=PORTC & 0b11100000;

                        }
                PORTCbits.RC2=0;
                break;
                }

        //ME
        case 96:
                {
```

```
                                        PORTCbits.RC2=0;
                                        Delay2(65251);
                                      PORTCbits.RC2=1;
                                        Delay2(65251);
                                        tone=PORTC & 0b11100000;
                                }
                                PORTCbits.RC2=0;
                                break;
                        }

                //TE
                case 224:
                        {
                        while(tone==224)
                                {
                                        PORTCbits.RC2=0;
                                        Delay2(65283);
                                      PORTCbits.RC2=1;
                                        Delay2(65283);
                                        tone=PORTC & 0b11100000;
                                }
                        PORTCbits.RC2=0;
                        break;
                        }
                }//end switch.
}

void Timer_Initialize(void)
        {
        Disable_Applications();
        // initialize PORTB as output for seven segmants.
        TRISB=0b00000000;
        }


int ASCII_Table(int value)
        {
                if(value == 0)
                        return 0xC0;

                else if(value == 1)
                        return 0xF9;

                else if(value == 2)
                        return 0xA4;

                else if(value == 3)
                        return 0xB0;
```

```
        else if(value == 4)
                return 0x99;

        else if(value == 5)
                return 0x92;

        else if(value == 6)
                return 0x82;

        else if(value == 7)
                return 0xF8;

        else if(value == 8)
                return 0x80;

        else
                return 0x90;

}

void Timer(void)
{
        int TIMER=0x00,temp1=0x00,temp2=0x00;
        int TIM01S=0;
        //clear timer.
        PORTB=0xFF;

        // input
        TRISDbits.TRISD6=1;
        TRISDbits.TRISD7=1;

        // delay 2microsecond.
        Delay1KTCYx(2);

        // if RD6 switch pressed then clear timer.
        if(PORTDbits.RD6==0)
                PORTB=0xFF;

        //if RD7 switch pressed then increment counter
        if(PORTDbits.RD7==0)
                {
                        TIMER++;
                        // convert from Hexdecimal  to decimal(DAA)
                        temp1=TIMER && 0x000F;
                        temp1=9-temp1;

                        if(temp1<0)
```

```
        while(tone==96)
                {
                PORTCbits.RC2=0;
                Delay2(65138);
                PORTCbits.RC2=1;
                Delay2(65138);
                tone=PORTC & 0b11100000;
                }

                PORTCbits.RC2=0;
                break;
          }

//FAH
case 128:
        {
        while(tone==128)
                {
                        PORTCbits.RC2=0;
                        Delay2(65179);
                      PORTCbits.RC2=1;
                        Delay2(65179);
                        tone=PORTC & 0b11100000;
                }
                 PORTCbits.RC2=0;
                 break;
           }

//SOH
case 160:
        {
        while(tone==160)
                {
                        PORTCbits.RC2=0;
                        Delay2(65216);
                     PORTCbits.RC2=1;
                        Delay2(65216);
                        tone=PORTC & 0b11100000;
                }
                PORTCbits.RC2=0;
                break;
           }

//LAH
case 192:
        {
        while(tone==192)
                {
```

```
                TIMER+=6;

        temp2=TIMER && 0x00F0;
        temp2=temp2>>4;
        temp2=0x0009-temp2;

        if(temp2<0)
                TIMER+=0x0060;
}

// display value in timer on seven segments.
TRISDbits.TRISD6=0;
TRISDbits.TRISD7=0;

// disable seven segments.
PORTDbits.RD6=1;
PORTDbits.RD7=1;

// 0.1 second counter
TIM01S=10;
while(TIM01S)
        {
                temp1=TIMER && 0x000F;
                PORTB=ASCII_Table(temp1);

                // enable CA1
                PORTDbits.RD6=0;

                //delay 5ms
                Delay(65516);

                //Turn off portb
                PORTB=0xFF;
                //disable CA1
                PORTDbits.RD6=1;
                temp2=TIMER && 0x00F0;
                temp2=temp2>>4;

                temp2=ASCII_Table(temp2);
                // display decimal  point
                temp2=temp2 && 0b01111111;
                PORTB=temp2;
                // enable CA2
                PORTDbits.RD7=0;
                //Turn off portb
                PORTB=0xFF;
                //disable CA2
                PORTDbits.RD7=1;
```

```
                                        TIM01S--;
                                    }
        }


void ADC_Initialize(void)
        {

                OpenADC(ADC_FOSC_8    &
                ADC_RIGHT_JUST &
                ADC_8_TAD ,
                ADC_CH0    &
                ADC_INT_OFF & ADC_VREFPLUS_VDD & ADC_VREFMINUS_VSS ,
                14);
        }

void LCD_Initialize(void)
        {
                Disable_Applications();
                OpenXLCD(FOUR_BIT & LINES_5X7);
                TRISA=0b00000001; // input to analog signal
        }

void ADC(void)
        {

                int result;

                LCD_Initialize();

                // set +Vref to Vdd and -Vref to Vss
                //ADCON1=ADCON1 && 0b11001111;



                Delay10TCYx( 5 ); // Delay for 50TCY
                ConvertADC(); // Start conversion

                while( BusyADC() ); // Wait for completion
                        result = ReadADC(); // Read result
                result=result*5/1024;
                while(BusyXLCD ());
                WriteDataXLCD(result);

        }
```

```
void DelayFor18TCY( void )
        {
                        Nop();
                        Nop();
                        Nop();
                        Nop();
                        Nop();
                        Nop();
                        Nop();
                        Nop();
                        Nop();
                        Nop();
                        Nop();
        }

void DelayPORXLCD (void)
        {
        Delay1KTCYx(15); // Delay of 15ms
        // Cycles = (TimeDelay * Fosc) / 4
        // Cycles = (15ms * 16MHz) / 4
        // Cycles = 60,000
        return;
        }

void DelayXLCD (void)
        {
        Delay1KTCYx(5); // Delay of 5ms
        // Cycles = (TimeDelay * Fosc) / 4
        // Cycles = (5ms * 16MHz) / 4
        // Cycles = 20,000
        return;
        }


void DC_Motor_Initialize(void)
{
                Disable_Applications();
                // PORTC (RC0,RC1,RC3)

                // use RC3 to detrmine the direction Backward or forward
                TRISCbits.TRISC3 = 1;

                // Use of CCP1 as PWM module at ~20kHz
                OpenPWM1(100);

                // Set 0% duty cycle to PWM1
```

```
                SetDCPWM1(0);

                // Make RC1 (right pwm) an output
                TRISCbits.TRISC1 = 0;

                // Make RC0 (right sign) an output
                TRISCbits.TRISC0 = 0;

}

void DC_Motor(unsigned int speed,char dir)
{

        if (dir=='B' )
                {
                        PORTCbits.RC0 =0;
                   SetDCPWM1(speed);

                }

  if (dir=='F')
                {
                        PORTCbits.RC0 =1;
                SetDCPWM1(speed);

                }
}
```

Appendix C

Software Tools

- **MPLAB IDE**

MPLAB IDE is a software program that runs on a PC to develop applications for Microchip microcontrollers. It is called an Integrated Development Environment, or IDE, because it provides a single integrated "environment" to develop code for embedded microcontrollers.

Microchip development tools consists of the following

- MPLAB IDE: Integrated Development Environment (IDE).
- MPLAB C18: C compiler.
- MPLAB ICD2: Programmer & Debugger tool.

Each one of this tools is used for specific operation, in this system all of these tools are be used.

First, run MPLAB IDE, create a project with specific name in specific location, open new text editor window to write the code, attach files (headers, linkers and other library that are been used) to the project which they are added to the project window as shown in the following figure.

Second, compile the project using MPLAB C18 and C compiler, which build an executable file for the microcontroller, in machine language (assembly language), and hex file.

Finally, Program the PIC microcontroller using MPLAB ICD2, then the PIC microcontroller will execute the instructions one after each other.

The organization of MPLAB IDE tools by function helps make pull-down menus and customizable quick keys easy to find and use. MPLAB IDE tools allow you to:

- Create and Edit Source Files
- Group Files into Projects
- Debug Source Code
- Assemble, compile, and link source code
- Debug the executable logic by watching program flow with the simulator, or in real time with the MPLAB-ICE emulator
- Make timing measurements
- View variables in watch windows
- Find quick answers to questions from the MPLAB IDE on-line Help and much more.

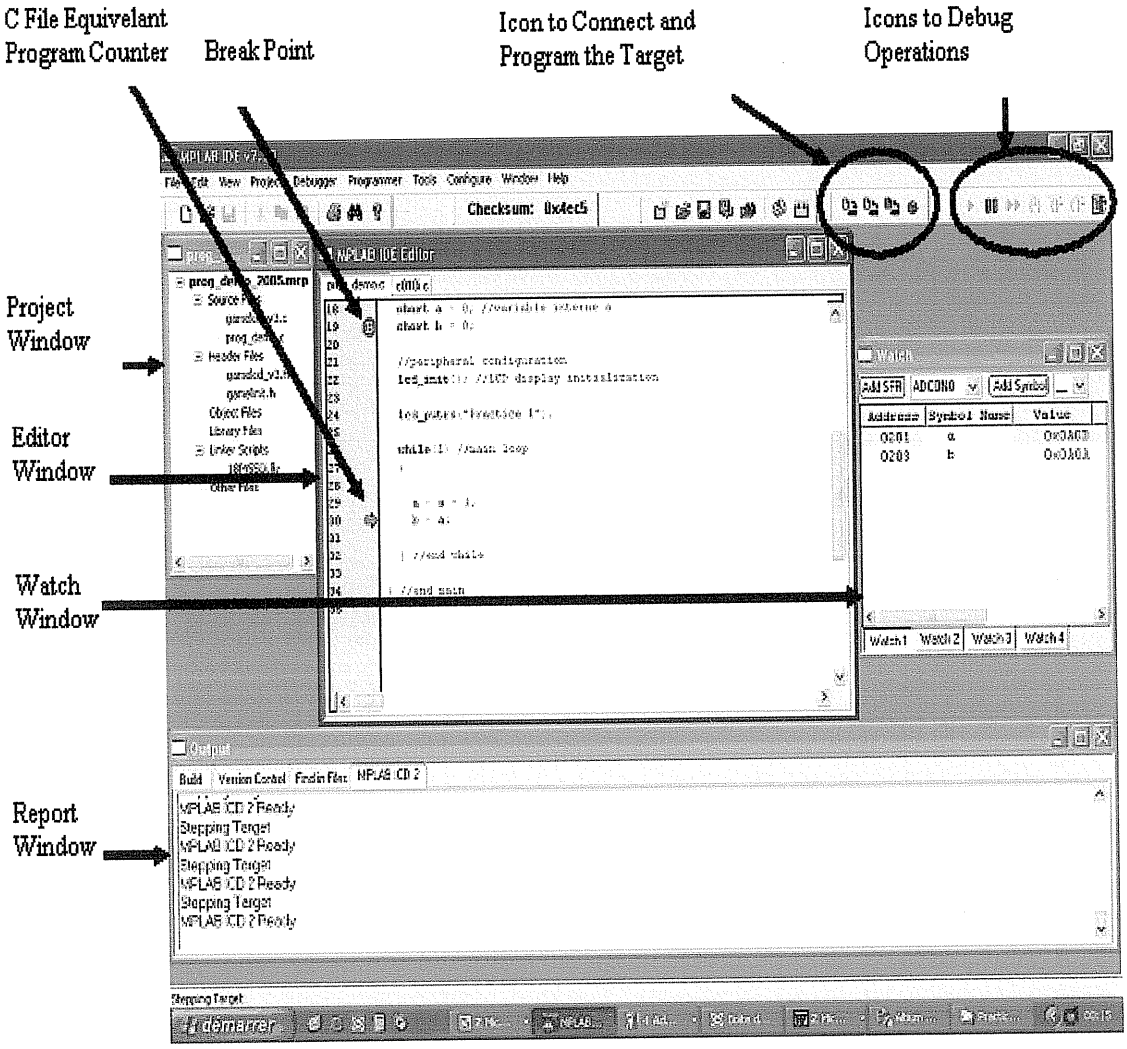


Figure: MPLAB IDE Component Window

- **WinPIC800**

This software is supported with parallel programmer, which is used to download hex code to PIC microcontroller; it supports PIC18 versions when the user chooses setting up hardware as ProPIC2 as shown in the following figure.

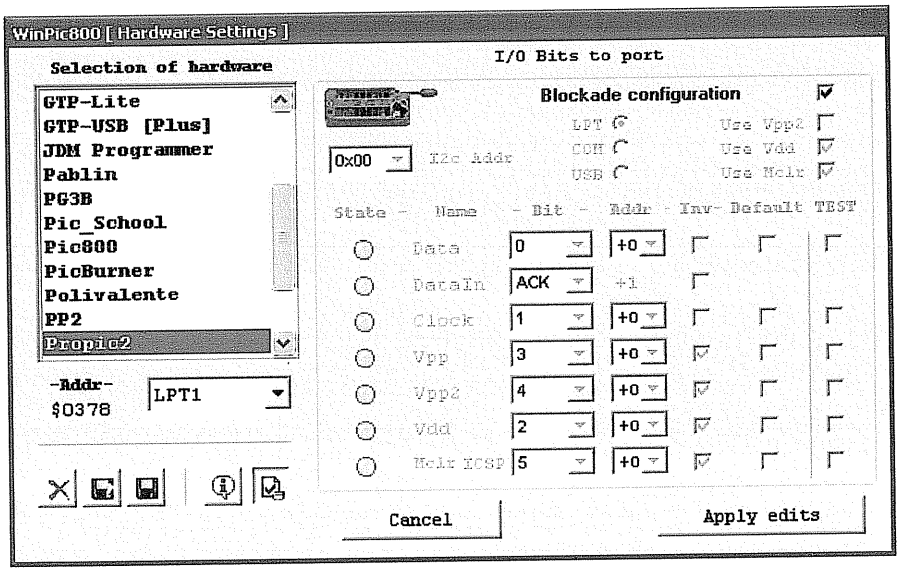


Figure: WinPIC800 Hardware settings

This software can only download hex code.

WinPIC800 software has the ability to:

- Read the hex code that already downloaded on the PIC and

display the code the user

- Program the PIC.
- Verify the code that downloaded on the PIC for the purpose of programming.
- Erase the code that is on the PIC.

As shown in Figure 5.3 WinPIC800 shows program memory of PIC microcontroller after erasing memory.

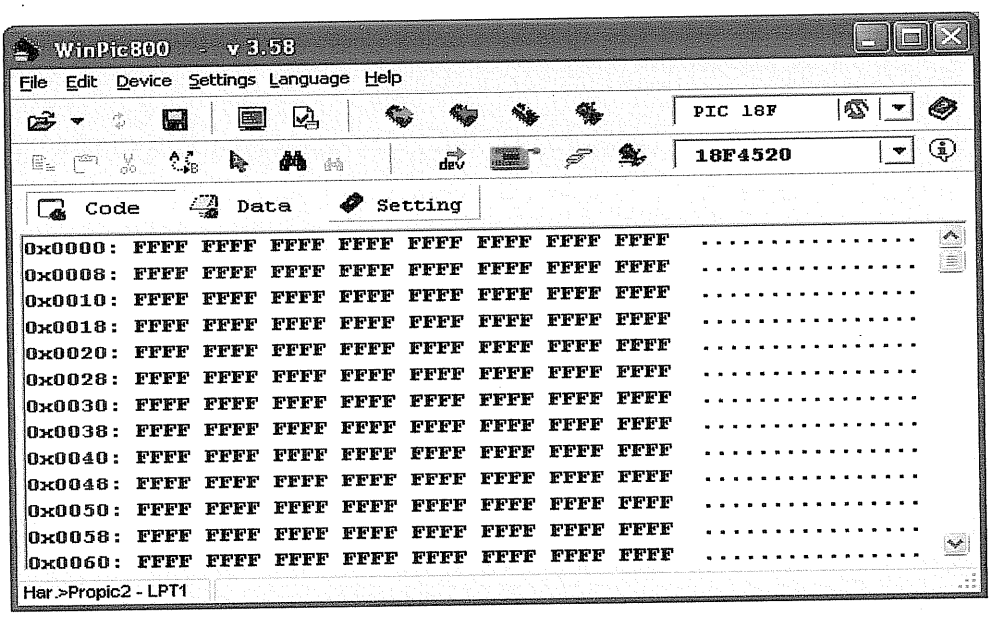


Figure: WinPIC800 window

- **PIC18 Simulator IDE**

Is powerful application that supplies PIC18 developers with user-friendly graphical development environment for Windows with integrated simulator (emulator), Basic compiler, assembler, disassembler and debugger, which support system's PIC version.

- **PIC18 Simulator IDE Main Features**

1. Main simulation interface showing internal microcontroller architecture
   - FLASH program memory editor.
   - EEPROM data memory editor.
   - Hardware stack viewer

2. Microcontroller pin out interface for simulation of digital I/O and analog inputs.

3. Breakpoints manager for code debugging with breakpoints support.

4. PIC18 assembler, interactive assembler editor for beginners, PIC18 disassembler.

5. Powerful PIC18 Basic compiler with smart Basic source editor
   - PIC18 Basic compiler features: high level language support for using internal EEPROM memory, using internal A/D converter module, using interrupts, interfacing character LCDs, and using internal PWM modules.

6. LCD module simulation interface for character LCD modules.

7. Oscilloscope and signal generator simulation tools.

8. 7-segment LED displays simulation interface.

9. Support for external simulation modules.

10. Extensive program options, color themes.

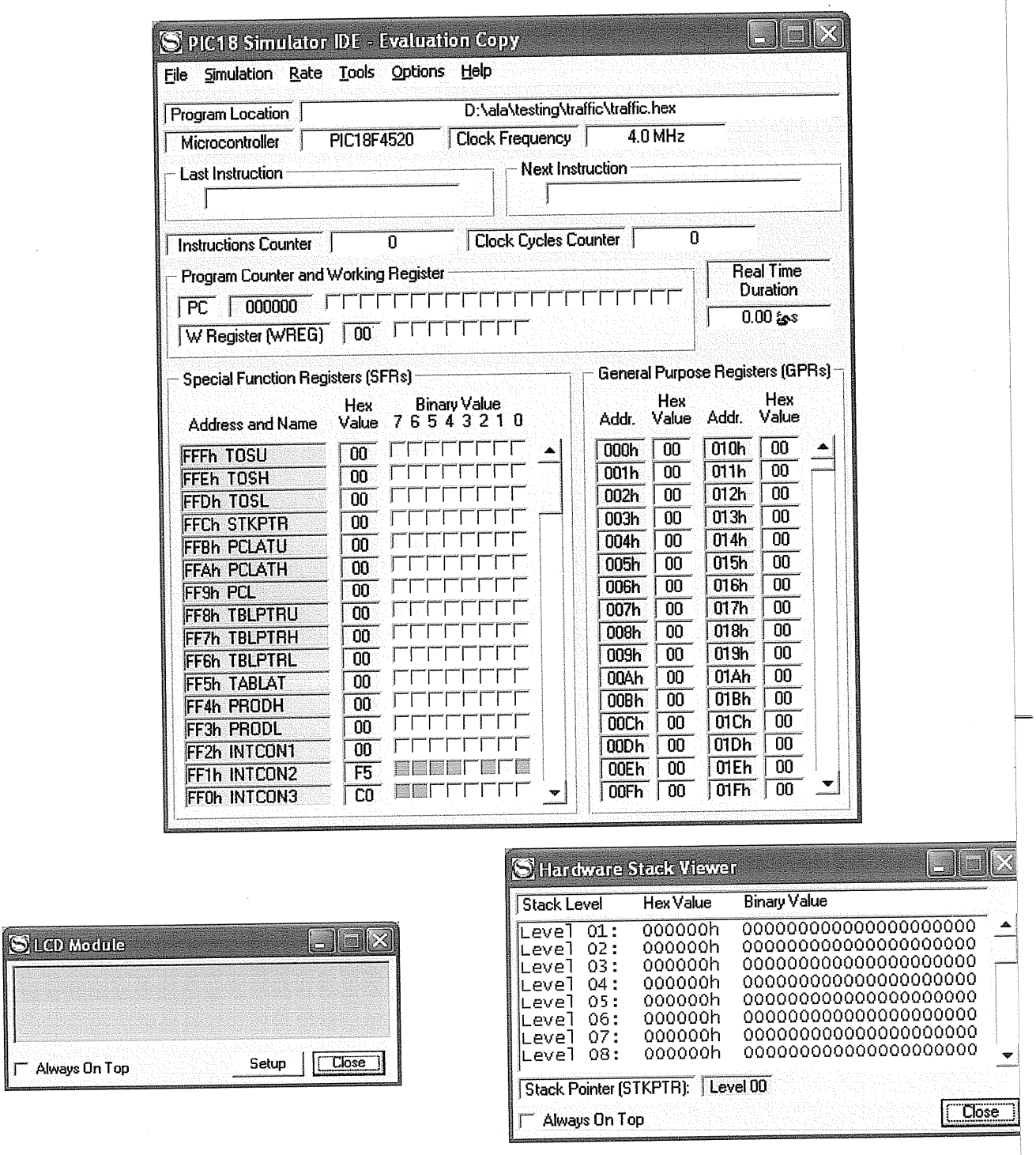


Figure: PIC18 Simulator IDE Window

Appendix D

Data Sheets

Low-Cost E Series Multifunction DAQ
12-Bit, 200 kS/s, 16 Analog Inputs

NI 6023E, NI 6024E, NI 6025E

- 16 analog inputs at 200 kS/s, 12-bit resolution
- Up to 2 analog outputs, 12-bit resolution
- 8 digital I/O lines (5 V/TTL/CMOS); two 24-bit counter/timers
- Digital triggering
- 4 analog input signal ranges
- NI-DAQ driver simplifies configuration and measurements

Models
- NI PCI-6023E
- NI PCI-6024E
- NI DAQCard-6024E for PCMCIA
- NI PCI-6025E
- NI PXI-6025E

*See ordering information

Operating Systems
- Windows 2000/NT/XP/Me/9x
- Mac OS 9*
- Real-time performance with LabVIEW (page 134)
- Others such as Linux (page 187)

Recommended Software
- LabVIEW
- LabWindows/CVI
- Measurement Studio for Visual Basic
- VI Logger

Other Compatible Software
- Visual Basic
- C/C++

Driver Software (included)
- NI-DAQ

Calibration Certificate Included
See page 21

Overview and Applications

National Instruments 6023E, NI 6024E and NI 6025E devices use E Series technology to deliver high performance, reliable data acquisition capabilities. These devices are used in a broad variety of applications including:

- Continuous high-speed data logging at up to 200 kS/s
- Externally timed and/or triggered data acquisition
- High-voltage and sensor measurements when used with NI signal conditioning (see page 244)
- High-channel-count system scalability with RTSI or PXI trigger bus

Features

NI 6023E, NI 6024E, and NI 6025E devices feature a highly precise voltage reference used during self-calibration. A simple software call initiates self-calibration, which minimizes errors caused by temperature drift and time. These devices feature the NI-PGIA, which is an instrumentation-class amplifier that guarantees settling times at all gains. Typical commercial off-the-shelf amplifier components might not meet the settling time requirements for high-gain measurement applications. Without the NI-PGIA, 12-bit devices with a 100X gain can have an effective resolution of only 10 bits. For a full description of NI accuracy advantages, see page 188. These devices offer several methods for connecting your signals including a differential mode for eight AI channels and maximum noise elimination, as well as referenced and nonreferenced single-ended mode for 16 AI channels.

NI 6023E, NI 6024E, and NI 6025E devices feature digital triggering, and two 24-bit 20 MHz counter/timers. NI 6023E and NI 6024E devices feature eight digital I/O lines compatible with both 5 V TTL and CMOS while NI 6025E devices feature 32 digital I/O lines. NI 6024E and NI 6025E devices feature two 12-bit analog outputs.

INFO CODES
For more information, or to order products online visit *ni.com/info* and enter:

pci6023e
pci6024e
daqcard6024e
pci6025e
pxi6025e

BUY ONLINE!

For a detailed list of differences between Performance E Series and Low-Cost E Series, see Table 1 on page 191.

Driver Software

NI-DAQ is the robust driver software included with all National Instruments data acquisition and signal conditioning products. This easy-to-use software tightly integrates the full functionality of your DAQ hardware to LabVIEW,

Family	Bus	Analog Inputs	Resolution	Sampling Rate S/s	Input Range	Analog Outputs	Resolution	Output Rate	Output Range	Digital I/O	Counter/ Timers	Triggers
NI 6023E	PCI	16 SE/8 DI	12 bits	200 kS/s	±0.05 to ±10 V	–	–	–	–	8	2, 24-bit	Digital
NI 6024E	PCI, PCMCIA	16 SE/8 DI	12 bits	200 kS/s	±0.05 to ±10 V	2	12 bits	10 kS/s[1]	±10 V	8	2, 24-bit	Digital
NI 6025E	PCI, PXI	16 SE/8 DI	12 bits	200 kS/s	±0.05 to ±10 V	2	12 bits	10 kS/s[1]	±10 V	32	2, 24-bit	Digital

[1]10 kS/s maximum when using the single DMA channel for analog output. 1 kS/s maximum when using the single DMA channel for either analog input or counter/timer operations. 1 kS/s maximum for DAQCard-6024E in all cases.

Table 1. NI 6023E, NI 6024E, and NI 6025E Channel, Speed, and Resolution Specifications (see page 233 for detailed specifications)

Low-Cost E Series Multifunction DAQ
12-Bit, 200 kS/s, 16 Analog Inputs

Nominal Range (V)		Absolute Accuracy							Relative Accuracy	
		% of Reading		Offset	Noise + Quantization (mV)		Temp	Absolute Accuracy	Resolution (mV)	
Positive FS	Negative FS	24 Hrs	1 Year	(mV)	Single Pt.	Averaged	Drift (%/°C)	at Full Scale (mV)	Single Pt.	Averaged
10.0	-10.0	0.0872	0.0914	6.380	3.910	0.975	0.0010	16.504	5.890	1.280
5.0	-5.0	0.0272	0.0314	3.200	1.950	0.488	0.0005	5.263	2.950	0.642
0.5	-0.5	0.0872	0.0914	0.340	0.195	0.049	0.0010	0.846	0.295	0.064
0.05	-0.05	0.0872	0.0914	0.054	0.063	0.006	0.0010	0.106	0.073	0.008

Note: Accuracies are valid for measurements following an internal E Series Calibration. Averaged numbers assume dithering and averaging of 100 single-channel readings. Measurement accuracies are listed for operational temperatures within ±1 °C of internal calibration temperature and ±10 °C of external or factory-calibration temperature. One-year calibration interval recommended. The Absolute Accuracy at Full Scale calculations were performed for a maximum range input voltage (for example, 10 V for the ±10 V range) after one year, assuming 100 pt averaging of data. See the overview on page 194 for example calculations.

Table 2. NI 6023E, PCI-6024E, and NI 6025E Analog Input Accuracy Specifications

Nominal Range (V)		Absolute Accuracy							Relative Accuracy	
		% of Reading		Offset	Noise + Quantization (mV)		Temp	Absolute Accuracy	Resolution (mV)	
Positive FS	Negative FS	24 Hrs	1 Year	(mV)	Single Pt.	Averaged	Drift (%/°C)	at Full Scale (mV)	Single Pt.	Averaged
10.0	-10.0	0.0872	0.0914	8.830	3.910	1.042	0.0010	19.012	5.890	1.370
5.0	-5.0	0.0272	0.0314	4.420	1.950	0.521	0.0005	6.517	2.950	0.686
0.5	-0.5	0.0872	0.0914	0.462	0.452	0.052	0.0010	0.972	0.516	0.069
0.05	-0.05	0.0872	0.0914	0.066	0.063	0.007	0.0010	0.119	0.073	0.009

Note: Accuracies are valid for measurements following an internal E Series Calibration. Averaged numbers assume dithering and averaging of 100 single-channel readings. Measurement accuracies are listed for operational temperatures within ±1 °C of internal calibration temperature and ±10 °C of external or factory-calibration temperature. One-year calibration interval recommended. The Absolute Accuracy at Full Scale calculations were performed for a maximum range input voltage (for example, 10 V for the ±10 V range) after one year, assuming 100 pt averaging of data. See the overview on page 194 for example calculations.

Table 3. DAQCard-6024E Analog Input Accuracy Specifications

Nominal Range (V)		Absolute Accuracy					Absolute Accuracy at Full Scale (mV)
		% of Reading				Temp	
Positive FS	Negative FS	24 Hrs	90 Days	1 Year	Offset (mV)	Drift (%/°C)	
10	-10	0.0177	0.0197	0.0219	5.93	0.0005	8.127

Note: Temp Drift applies only if ambient is greater than ±10 °C of previous external calibration. See page 194 for example calculations.

Table 4. PCI-6024E, and NI 6025E Analog Output Accuracy Specifications

Nominal Range (V)		Absolute Accuracy					Absolute Accuracy at Full Scale (mV)
		% of Reading				Temp	
Positive FS	Negative FS	24 Hrs	90 Days	1 Year	Offset (mV)	Drift (%/°C)	
10	-10	0.0177	0.0197	0.0219	8.37	0.0005	10.568

Note: Temp Drift applies only if ambient is greater than ±10 °C of previous external calibration. See page 194 for example calculations.

Table 5. DAQCard-6024E Analog Output Accuracy Specifications

LabWindows/CVI, and Measurement Studio for Visual Basic. High-performance features include multidevice synchronization, networked measurements, and DMA data management. Bundled with NI-DAQ, the Measurement & Automation Explorer utility simplifies the configuration of your measurement hardware with device test panels, interactive measurements, and scaled I/O channels. NI-DAQ also provides numerous example programs for LabVIEW and other application development environments to get you started with your application quickly.

Related Products

For related products, please refer to:

- SCXI Signal Conditioning – page 246
- SCC Signal Conditioning – page 320
- Analog Output Multifunction DAQ – page 365
- High-Speed Digital I/O – page 378

See page 221 for connector diagrams.
See page 233 for detailed specifications.

Ordering Information

NI PCI-6023E 777742-01
NI PCI-6024E 777743-01
NI DAQCard-6024E[1] 778269-01
NI PCI-6025E[1] 777744-01
NI PXI-6025E[1] 777798-01

Includes NI-DAQ driver software.
[1]Windows only.

For information on extended warranty and value-added services, see page 20.

Recommended Configurations

Family	DAQ Device	Accessory	Cable
NI 6023E	PCI-6023E	CB-68LP (777145-01)	R6868 (182482-01)
NI 6024E	PCI-6024E	CB-68LP (777145-01)	R6868 (182482-01)
	DAQCard-6024E	CB-68LP (777145-01)	RC68-68 (187252-01)
NI 6025E	PCI-6025E	Two CB-50LPs (777101-01)	R1005050 (182762-01)
	PXI-6025E	Two CB-50LPs (777101-01)	R1005050 (182762-01)

For E Series accessory and cable information, see page 221.

Multifunction DAQ Overview

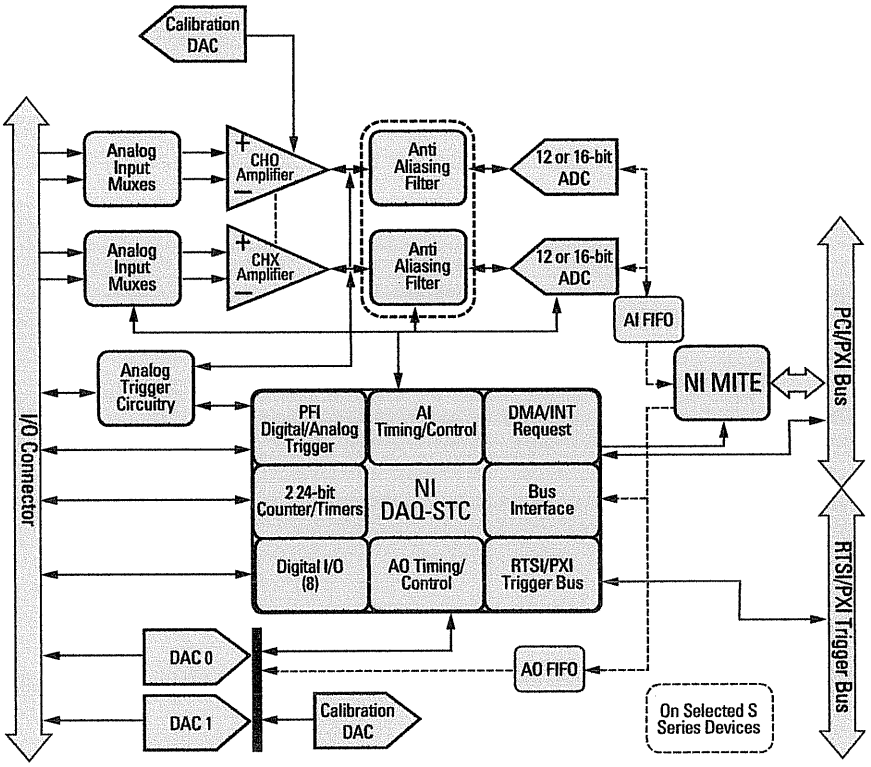


Diagram 1. S Series Diagram

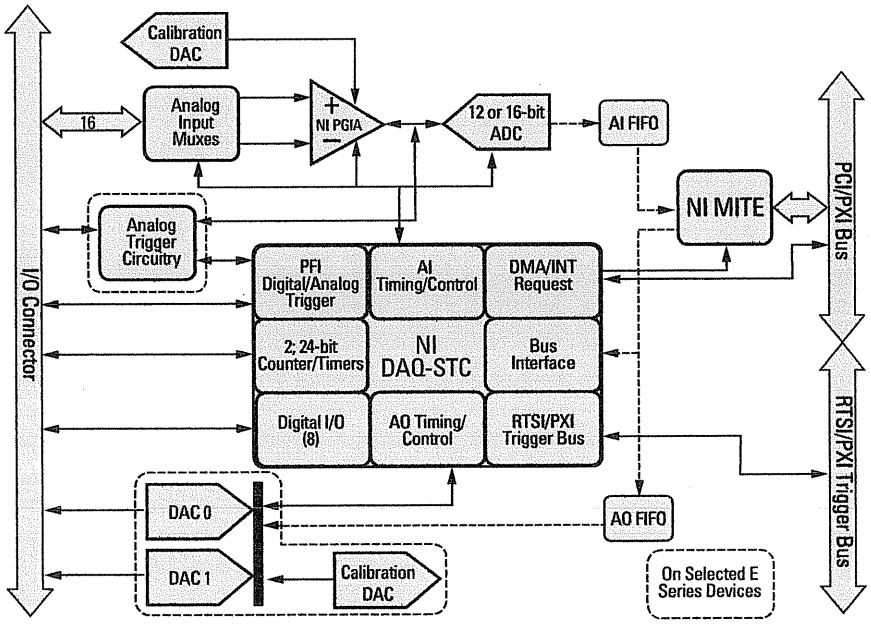


Diagram 2. E Series Diagram

Multifunction DAQ Overview

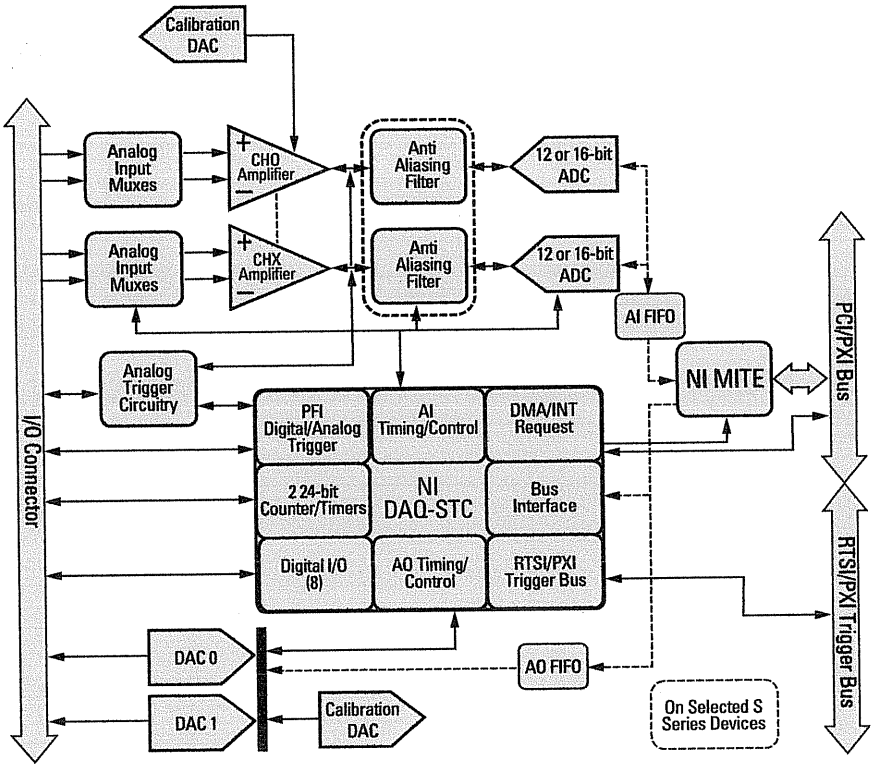


Diagram 1. S Series Diagram

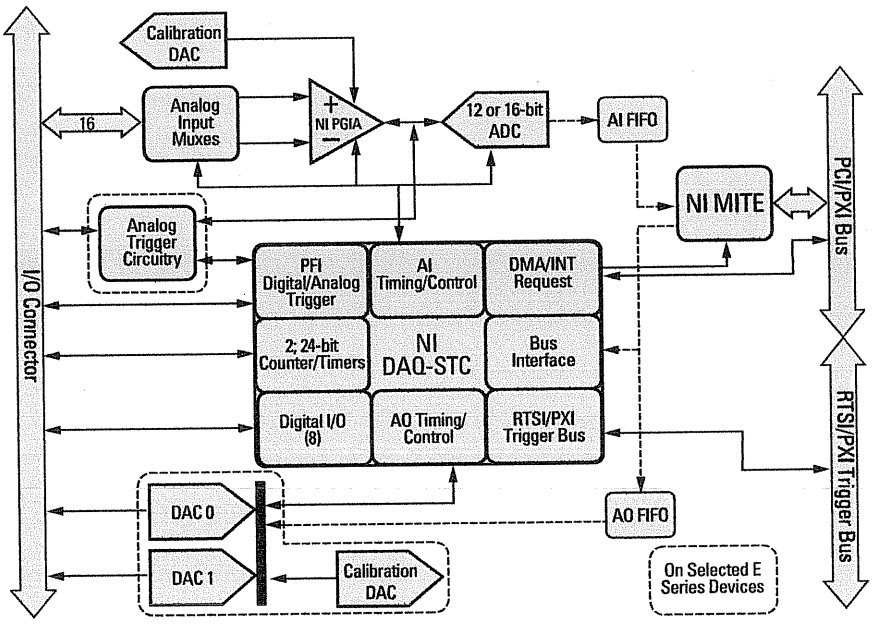


Diagram 2. E Series Diagram

HD44780U (LCD-II)

(Dot Matrix Liquid Crystal Display Controller/Driver)

HITACHI

Description

The HD44780U dot-matrix liquid crystal display controller and driver LSI displays alphanumerics, Japanese kana characters, and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8-bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver.

A single HD44780U can display up to one 8-character line or two 8-character lines.

The HD44780U has pin function compatibility with the HD44780S which allows the user to easily replace an LCD-II with an HD44780U. The HD44780U character generator ROM is extended to generate 208 5 × 8 dot character fonts and 32 5 × 10 dot character fonts for a total of 240 different character fonts.

The low power supply (2.7V to 5.5V) of the HD44780U is suitable for any portable battery-driven product requiring low power dissipation.

Features

- 5 × 8 and 5 × 10 dot matrix possible
- Low power operation support:
  — 2.7 to 5.5V
- Wide range of liquid crystal display driver power
  — 3.0 to 11V
- Liquid crystal drive waveform
  — A (One line frequency AC waveform)
- Correspond to high speed MPU bus interface
  — 2 MHz (when V_{cc} = 5V)
- 4-bit or 8-bit MPU interface enabled
- 80 × 8-bit display RAM (80 characters max.)
- 9,920-bit character generator ROM for a total of 240 character fonts
  — 208 character fonts (5 × 8 dot)
  — 32 character fonts (5 × 10 dot)

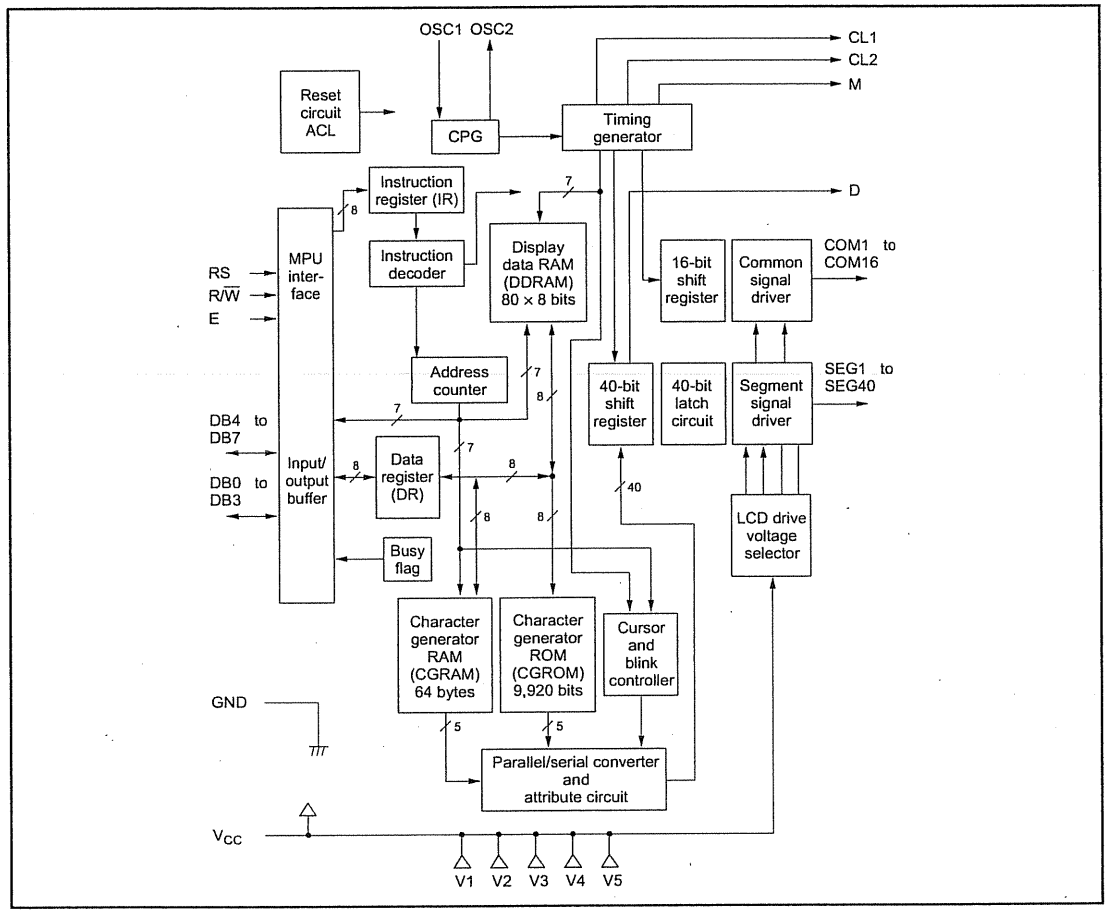
- 64 × 8-bit character generator RAM
  - — 8 character fonts (5 × 8 dot)
  - — 4 character fonts (5 × 10 dot)
- 16-common × 40-segment liquid crystal display driver
- Programmable duty cycles
  - — 1/8 for one line of 5 × 8 dots with cursor
  - — 1/11 for one line of 5 × 10 dots with cursor
  - — 1/16 for two lines of 5 × 8 dots with cursor
- Wide range of instruction functions:
  - — Display clear, cursor home, display on/off, cursor on/off, display character blink, cursor shift, display shift
- Pin function compatibility with HD44780S
- Automatic reset circuit that initializes the controller/driver after power on
- Internal oscillator with external resistors
- Low power consumption

Ordering Information

Type No.	Package	CGROM
HD44780UA00FS	FP-80B	Japanese standard font
HCD44780UA00	Chip	
HD44780UA00TF	TFP-80F	
HD44780UA02FS	FP-80B	European standard font
HCD44780UA02	Chip	
HD44780UA02TF	TFP-80F	
HD44780UBxxFS	FP-80B	Custom font
HCD44780UBxx	Chip	
HD44780UBxxTF	TFP-80F	

Note: xx: ROM code No.

HD44780U Block Diagram



Character Generator ROM (CGROM)

The character generator ROM generates 5 × 8 dot or 5 × 10 dot character patterns from 8-bit character codes (Table 4). It can generate 208 5 × 8 dot character patterns and 32 5 × 10 dot character patterns. User-defined character patterns are also available by mask-programmed ROM.

Character Generator RAM (CGRAM)

In the character generator RAM, the user can rewrite character patterns by program. For 5 × 8 dots, eight character patterns can be written, and for 5 × 10 dots, four character patterns can be written.

Write into DDRAM the character codes at the addresses shown as the left column of Table 4 to show the character patterns stored in CGRAM.

See Table 5 for the relationship between CGRAM addresses and data and display patterns.

Areas that are not used for display can be used as general data RAM.

Modifying Character Patterns

- Character pattern development procedure

The following operations correspond to the numbers listed in Figure 7:

1. Determine the correspondence between character codes and character patterns.
2. Create a listing indicating the correspondence between EPROM addresses and data.
3. Program the character patterns into the EPROM.
4. Send the EPROM to Hitachi.
5. Computer processing on the EPROM is performed at Hitachi to create a character pattern listing, which is sent to the user.
6. If there are no problems within the character pattern listing, a trial LSI is created at Hitachi and samples are sent to the user for evaluation. When it is confirmed by the user that the character patterns are correctly written, mass production of the LSI proceeds at Hitachi.

- Programming character patterns

  This section explains the correspondence between addresses and data used to program character patterns in EPROM. The HD44780U character generator ROM can generate 208 5 × 8 dot character patterns and 32 5 × 10 dot character patterns for a total of 240 different character patterns.

  — Character patterns

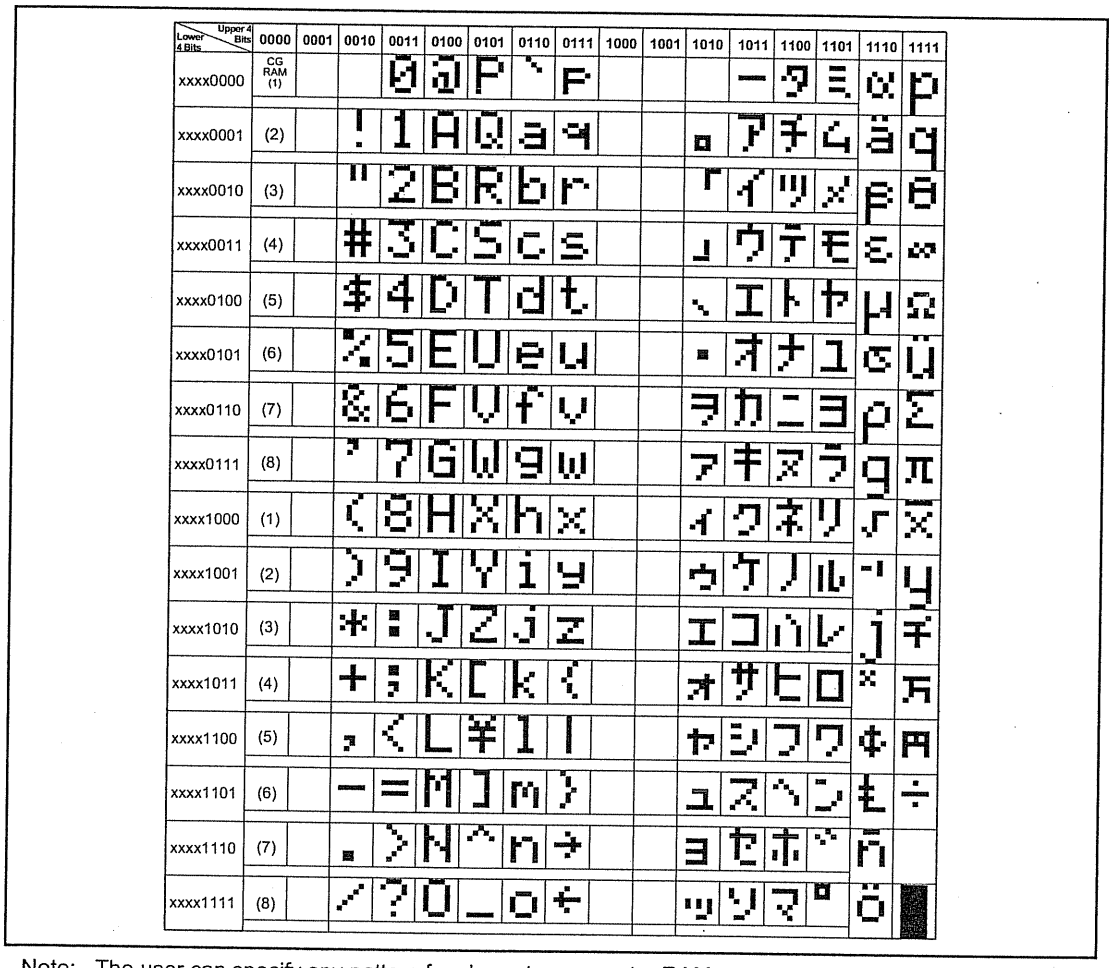
  EPROM address data and character pattern data correspond with each other to form a 5 × 8 or 5 × 10 dot character pattern (Tables 2 and 3).

Table 2 Example of Correspondence between EPROM Address Data and Character Pattern (5 × 8 Dots)

EPROM Address												Data					
A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	O4	O3	O2	O1	LSB O0	
								0	0	0	0	1	0	0	0	0	
								0	0	0	1	1	0	0	0	0	
								0	0	1	0	1	0	1	1	0	
								0	0	1	1	1	1	0	0	1	
								0	1	0	0	1	0	0	0	1	
								0	1	0	1	1	0	0	0	1	
								0	1	1	0	1	1	1	1	0	
0	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	← Cursor position
								1	0	0	0	0	0	0	0	0	
								1	0	0	1	0	0	0	0	0	
								1	0	1	0	0	0	0	0	0	
								1	0	1	1	0	0	0	0	0	
								1	1	0	0	0	0	0	0	0	
								1	1	0	1	0	0	0	0	0	
								1	1	1	0	0	0	0	0	0	
								1	1	1	1	0	0	0	0	0	
Character code								Line position									

Notes: 1. EPROM addresses A11 to A4 correspond to a character code.
2. EPROM addresses A3 to A0 specify a line position of the character pattern.
3. EPROM data O4 to O0 correspond to character pattern data.
4. EPROM data O5 to O7 must be specified as 0.
5. A lit display position (black) corresponds to a 1.
6. Line 9 and the following lines must be blanked with 0s for a 5 × 8 dot character fonts.

Table 4 **Correspondence between Character Codes and Character Patterns (ROM Code: A00)**



Note: The user can specify any pattern for character-generator RAM.

MOTOROLA

10-LINE-TO-4-LINE AND 8-LINE-TO-3-LINE PRIORITY ENCODERS

The SN54/74LS147 and the SN54/74LS148 are Priority Encoders. They provide priority decoding of the inputs to ensure that only the highest order data line is encoded. Both devices have data inputs and outputs which are active at the low logic level.

The LS147 encodes nine data lines to four-line (8-4-2-1) BCD. The implied decimal zero condition does not require an input condition because zero is encoded when all nine data lines are at a high logic level.

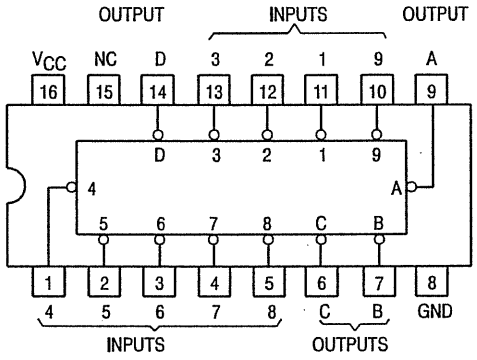
The LS148 encodes eight data lines to three-line (4-2-1) binary (octal). By providing cascading circuitry (Enable Input EI and Enable Output EO) octal expansion is allowed without needing external circuitry.

The SN54/74LS748 is a proprietary Motorola part incorporating a built-in deglitcher network which minimizes glitches on the GS output. The glitch occurs on the negative going transition of the EI input when data inputs 0–7 are at logical ones.

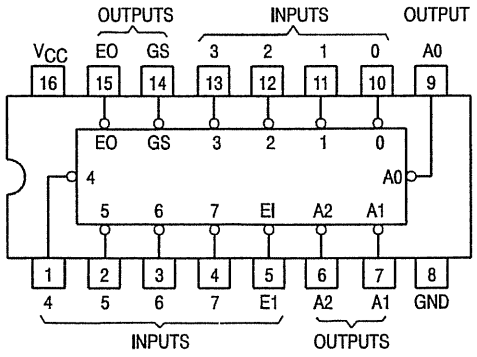
The only dc parameter differences between the LS148 and the LS748 are that (1) Pin 10 (input 0) has a fan-in of 2 on the LS748 versus a fan-in of 1 on the LS148; (2) Pins 1, 2, 3, 4, 11, 12 and 13 (inputs 1, 2, 3, 4, 5, 6, 7) have a fan-in of 3 on the LS748 versus a fan-in of 2 on the LS148.

The only ac difference is that t_{PHL} from EI to EO is changed from 40 to 45 ns.

SN54/74LS147
(TOP VIEW)



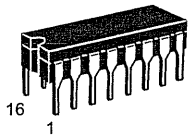
SN54/74LS148
SN54/74LS748
(TOP VIEW)



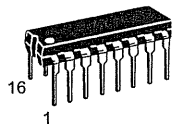
SN54/74LS147
SN54/74LS148
SN54/74LS748

10-LINE-TO-4-LINE AND 8-LINE-TO-3-LINE PRIORITY ENCODERS

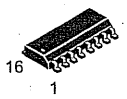
LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08



D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

SN54LSXXXJ	Ceramic
SN74LSXXXN	Plastic
SN74LSXXXD	SOIC

SN54/74LS147
FUNCTION TABLE

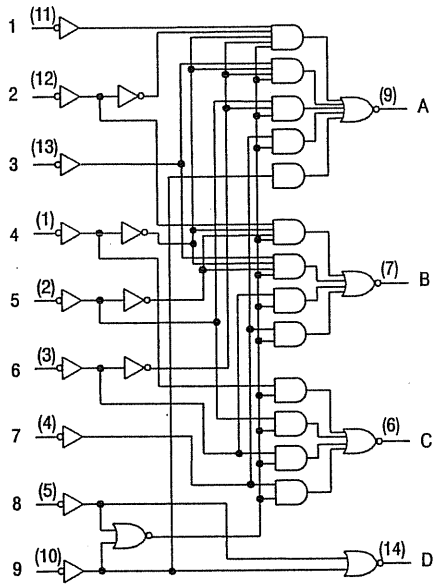
INPUTS									OUTPUTS			
1	2	3	4	5	6	7	8	9	D	C	B	A
H	H	H	H	H	H	H	H	H	H	H	H	H
X	X	X	X	X	X	X	X	L	L	H	H	L
X	X	X	X	X	X	X	L	H	L	H	H	H
X	X	X	X	X	X	L	H	H	H	L	L	L
X	X	X	X	X	L	H	H	H	H	L	L	H
X	X	X	X	L	H	H	H	H	H	L	H	L
X	X	X	L	H	H	H	H	H	H	L	H	H
X	X	L	H	H	H	H	H	H	H	H	L	L
X	L	H	H	H	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Logic Level, L = LOW Logic Level, X = Irrelevant

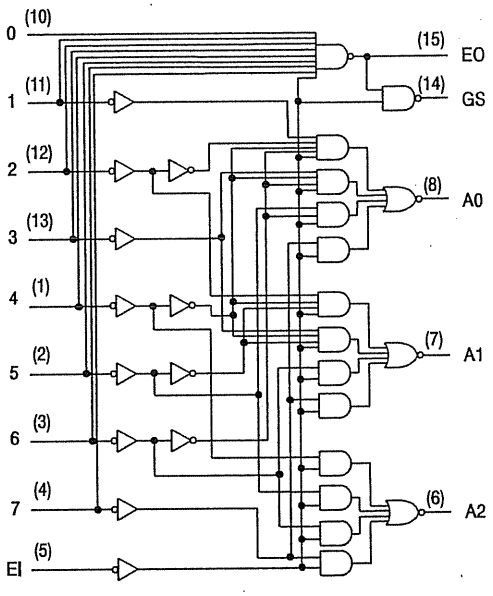
SN54/74LS148
SN54/74LS748
FUNCTION TABLE

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

FUNCTIONAL BLOCK DIAGRAMS

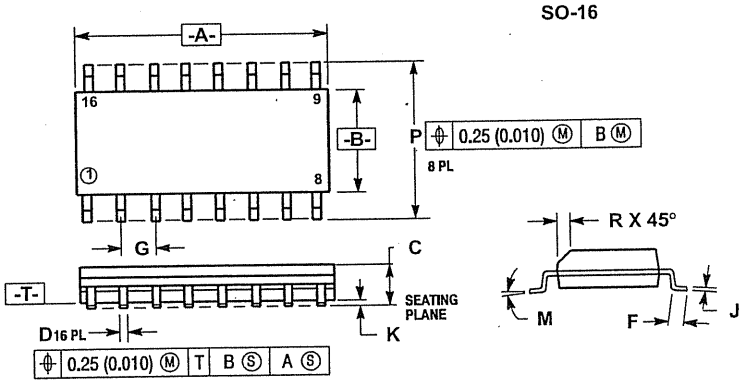


SN54/74LS147



SN54/74LS148

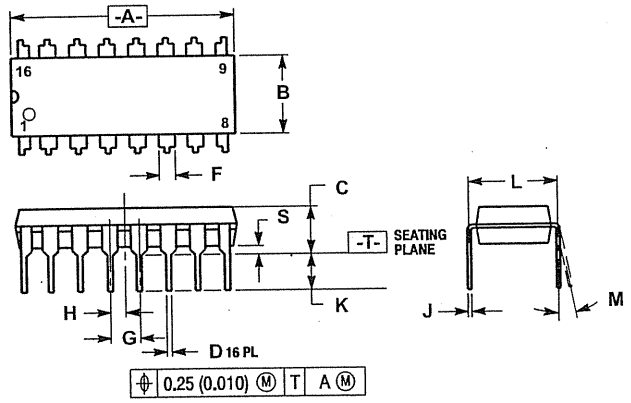
Case 751B-03 D Suffix
16-Pin Plastic
SO-16



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DIMENSION A AND B DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 (0.006) PER SIDE.
5. 751B-01 IS OBSOLETE, NEW STANDARD 751B-03.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	9.80	10.00	0.386	0.393
B	3.80	4.00	0.150	0.157
C	1.35	1.75	0.054	0.068
D	0.35	0.49	0.014	0.019
F	0.40	1.25	0.016	0.049
G	1.27 BSC		0.050 BSC	
J	0.19	0.25	0.008	0.009
K	0.10	0.25	0.004	0.009
M	0°	7°	0°	7°
P	5.80	6.20	0.229	0.244
R	0.25	0.50	0.010	0.019

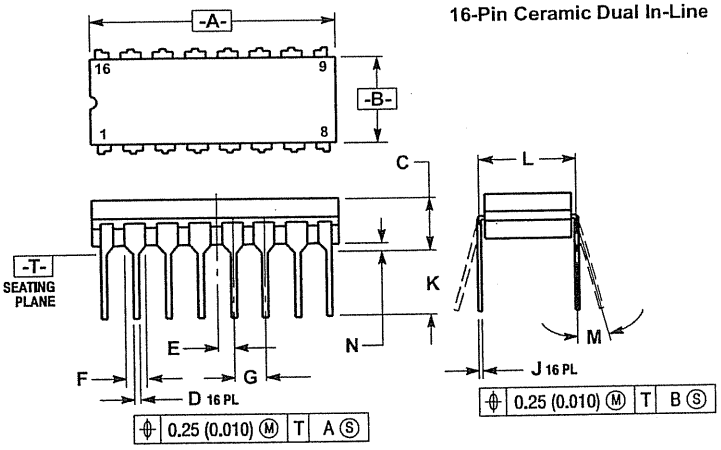
Case 648-08 N Suffix
16-Pin Plastic



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
4. DIMENSION "B" DOES NOT INCLUDE MOLD FLASH.
5. ROUNDED CORNERS OPTIONAL.
6. 648-01 THRU -07 OBSOLETE, NEW STANDARD 648-08.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	18.80	19.55	0.740	0.770
B	6.35	6.85	0.250	0.270
C	3.69	4.44	0.145	0.175
D	0.39	0.53	0.015	0.021
F	1.02	1.77	0.040	0.070
G	2.54 BSC		0.100 BSC	
H	1.27 BSC		0.050 BSC	
J	0.21	0.38	0.008	0.015
K	2.80	3.30	0.110	0.130
L	7.50	7.74	0.295	0.305
M	0°	10°	0°	10°
S	0.51	1.01	0.020	0.040

Case 620-09 J Suffix
16-Pin Ceramic Dual In-Line



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
4. DIM F MAY NARROW TO 0.76 (0.030) WHERE THE LEAD ENTERS THE CERAMIC BODY.
5. 620-01 THRU -08 OBSOLETE, NEW STANDARD 620-09.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	19.05	19.55	0.750	0.770
B	6.10	7.36	0.240	0.290
C	—	4.19	—	0.165
D	0.39	0.53	0.015	0.021
E	1.27 BSC		0.050 BSC	
F	1.40	1.77	0.055	0.070
G	2.54 BSC		0.100 BSC	
J	0.23	0.27	0.009	0.011
K	—	5.08	—	0.200
L	7.62 BSC		0.300 BSC	
M	0°	15°	0°	15°
N	0.39	0.88	0.015	0.035

MICROCHIP

PIC18F2420/2520/4420/4520

28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology

Power Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode current down to 0.1 µA typical
- Timer1 Oscillator: 1.8 µA, 32 kHz, 2V
- Watchdog Timer: 2.1 µA
- Two-Speed Oscillator Start-up

Peripheral Highlights:

- High-current sink/source 25 mA/25 mA
- Three programmable external interrupts
- Four input change interrupts
- Up to 2 Capture/Compare/PWM (CCP) modules, one with Auto-Shutdown (28-pin devices)
- Enhanced Capture/Compare/PWM (ECCP) module (40/44-pin devices only):
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI™ (all 4 modes) and I^2C™ Master and Slave Modes
- Enhanced Addressable USART module:
  - Supports RS-485, RS-232 and LIN 1.2
  - RS-232 operation using internal oscillator block (no external crystal required)
  - Auto-Wake-up on Start bit
  - Auto-Baud Detect
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D):
  - Auto-acquisition capability
  - Conversion available during Sleep
- Dual analog comparators with input multiplexing)

Flexible Oscillator Structure:

- Four Crystal modes, up to 40 MHz
- 4X Phase Lock Loop (available for crystal and internal oscillators)
- Two External RC modes, up to 4 MHz
- Two External Clock modes, up to 40 MHz
- Internal oscillator block:
  - 8 user selectable frequencies, from 31 kHz to 8 MHz
  - Provides a complete range of clock speeds from 31 kHz to 32 MHz when used with PLL
  - User tunable to compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if peripheral clock stops

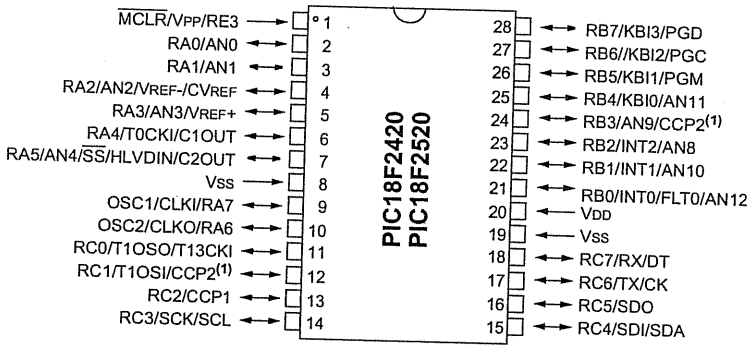
Special Microcontroller Features:

- C compiler optimized architecture:
  - Optional extended instruction set designed to optimize re-entrant code
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Flash/Data EEPROM Retention: 100 years typical
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Single-supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Wide operating voltage range: 2.0V to 5.5V
- Programmable 16-level High/Low-Voltage Detection (HLVD) module:
  - Supports interrupt on High/Low-Voltage Detection
- Programmable Brown-out Reset (BOR
  - With software enable option

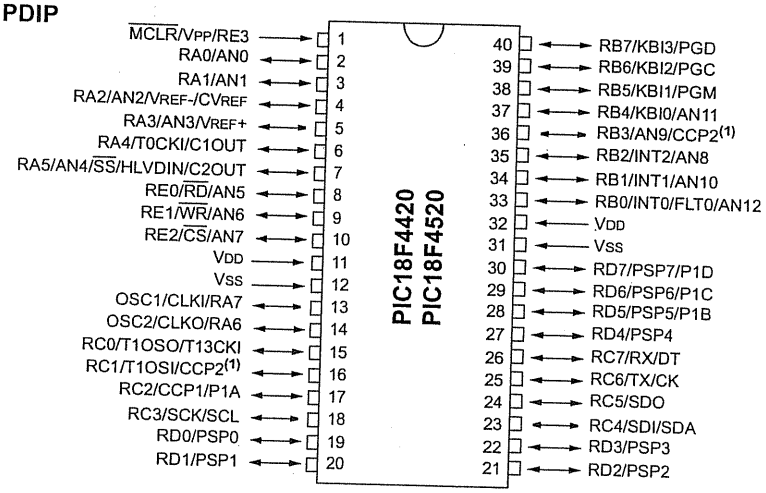
Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ ECCP (PWM)	MSSP		EUSART	Comp.	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I^2C			
PIC18F2420	16K	8192	768	256	25	10	2/0	Y	Y	1	2	1/3
PIC18F2520	32K	16384	1536	256	25	10	2/0	Y	Y	1	2	1/3
PIC18F4420	16K	8192	768	256	36	13	1/1	Y	Y	1	2	1/3
PIC18F4520	32K	16384	1536	256	36	13	1/1	Y	Y	1	2	1/3

Pin Diagrams

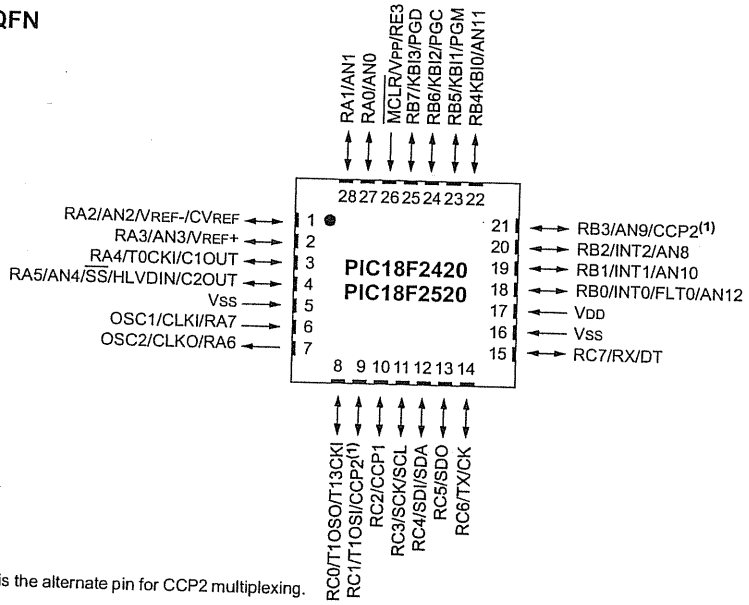
28-pin PDIP, SOIC



40-pin PDIP



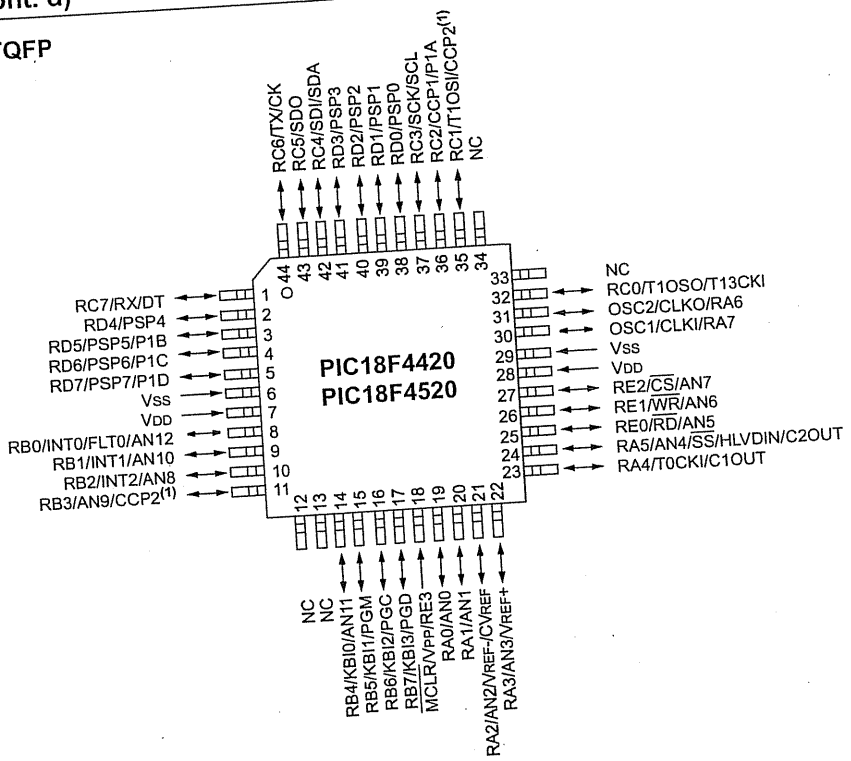
28-pin QFN



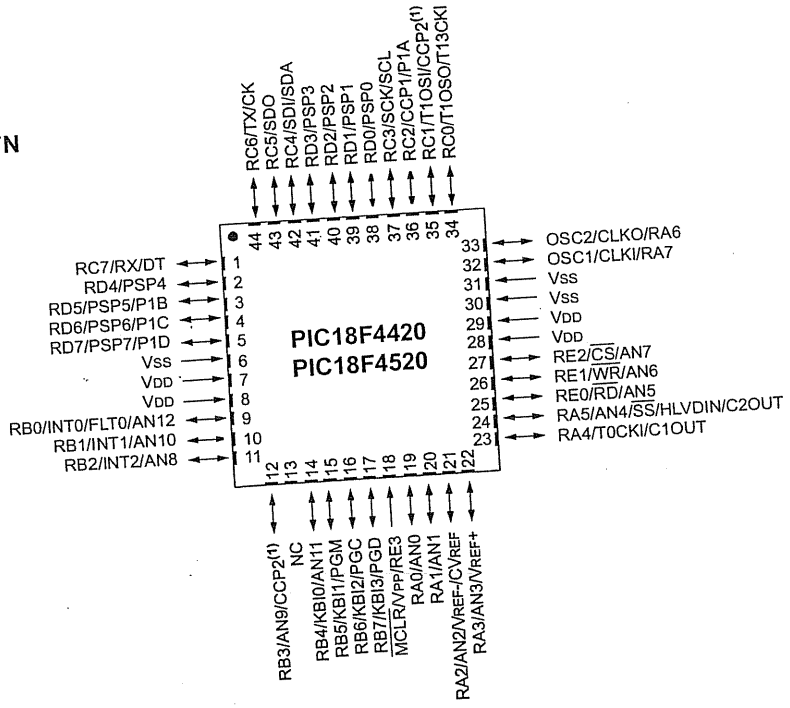
Note 1: RB3 is the alternate pin for CCP2 multiplexing.

Pin Diagrams (Cont.'d)

44-pin TQFP



44-pin QFN



Note 1: RB3 is the alternate pin for CCP2 multiplexing.

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F2420
- PIC18F2520
- PIC18F4420
- PIC18F4520
- PIC18LF2420
- PIC18LF2520
- PIC18LF4420
- PIC18LF4520

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance, Enhanced Flash program memory. On top of these features, the PIC18F2420/2520/4420/4520 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F2420/2520/4420/4520 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching:** The power managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 26.0 "Electrical Characteristics"** for values.

1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F2420/2520/4420/4520 family offer ten different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes, using crystals or ceramic resonators
- Two External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O)
- Two External RC Oscillator modes with the same pin options as the External Clock modes
- An internal oscillator block which provides an 8 MHz clock and an INTRC source (approximately 31 kHz), as well as a range of 6 user selectable clock frequencies, between 125 kHz to 4 MHz, for a total of 8 clock frequencies. This option frees the two oscillator pins for use as additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the high-speed crystal and internal oscillator modes, which allows clock speeds of up to 40 MHz. Used with the internal oscillator, the PLL gives users a complete selection of clock speeds, from 31 kHz to 32 MHz – all without using an external crystal or clock circuit.

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.2 Other Special Features

- **Memory Endurance:** The Enhanced Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 100,000 for program memory and 1,000,000 for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-programmability:** These devices can write to their own program memory spaces under internal software control. By using a bootloader routine located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18F2420/2520/4420/4520 family introduces an optional extension to the PIC18 instruction set, which adds 8 new instructions and an Indexed Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.
- **Enhanced CCP module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include Auto-Shutdown, for disabling PWM outputs on interrupt or other select conditions and Auto-Restart, to reactivate outputs once the condition has cleared.
- **Enhanced Addressable USART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include automatic baud rate detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the USART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reduce code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit prescaler, allowing an extended time-out range that is stable across operating voltage and temperature. See **Section 26.0 "Electrical Characteristics"** for time-out periods.

1.3 Details on Individual Family Members

Devices in the PIC18F2420/2520/4420/4520 family are available in 28-pin and 40/44-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in five ways:

1. Flash program memory (16 Kbytes for PIC18F2420/4420 devices and 32 Kbytes for PIC18F2520/4520).
2. A/D channels (10 for 28-pin devices, 13 for 40/44-pin devices).
3. I/O ports (3 bidirectional ports on 28-pin devices, 5 bidirectional ports on 40/44-pin devices).
4. CCP and Enhanced CCP implementation (28-pin devices have 2 standard CCP modules, 40/44-pin devices have one standard CCP module and one ECCP module).
5. Parallel Slave Port (present only on 40/44-pin devices).

All other features for devices in this family are identical. These are summarized in Table 1-1.

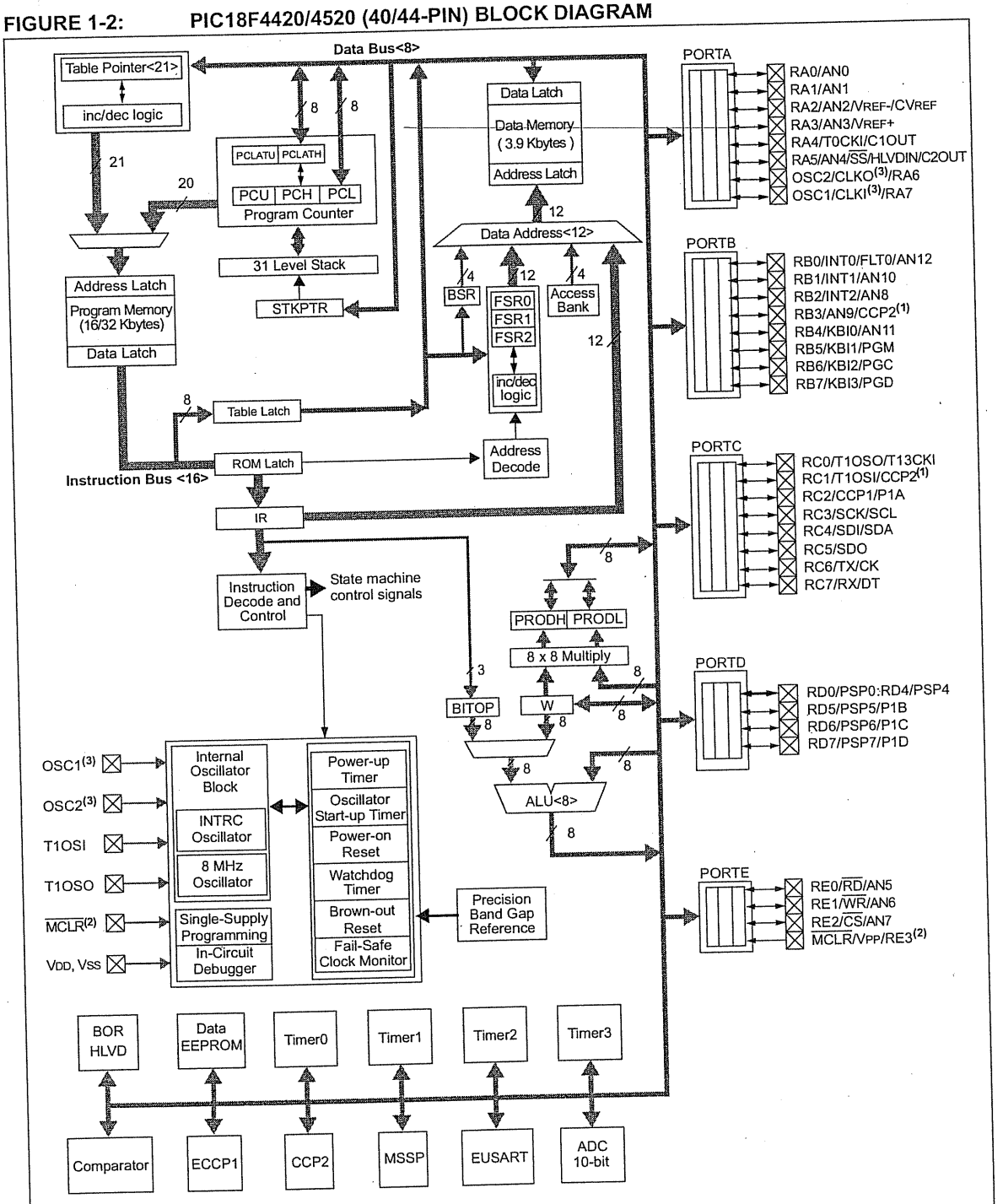
The pinouts for all devices are listed in Table 1-2 and Table 1-3.

Like all Microchip PIC18 devices, members of the PIC18F2420/2520/4420/4520 family are available as both standard and low-voltage devices. Standard devices with Enhanced Flash memory, designated with an "F" in the part number (such as PIC18**F**2420), accommodate an operating VDD range of 4.2V to 5.5V. Low-voltage parts, designated by "LF" (such as PIC18**LF**2420), function over an extended VDD range of 2.0V to 5.5V.

TABLE 1-1: DEVICE FEATURES

Features	PIC18F2420	PIC18F2520	PIC18F4420	PIC18F4520
Operating Frequency	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz
Program Memory (Bytes)	16384	32768	16384	32768
Program Memory (Instructions)	8192	16384	8192	16384
Data Memory (Bytes)	768	1536	768	1536
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Parallel Communications (PSP)	No	No	Yes	Yes
10-bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable High/Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin PDIP 28-pin SOIC 28-pin QFN	28-pin PDIP 28-pin SOIC 28-pin QFN	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

FIGURE 1-2: PIC18F4420/4520 (40/44-PIN) BLOCK DIAGRAM



Note 1: CCP2 is multiplexed with RC1 when configuration bit CCP2MX is set, or RB3 when CCP2MX is not set.

2: RE3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.

3: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

TABLE 1-3: PIC18F4420/4520 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
MCLR/VPP/RE3	1	18	18			Master Clear (input) or programming voltage (input).
MCLR				I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP				P		Programming voltage input.
RE3				I	ST	Digital input.
OSC1/CLKI/RA7	13	32	30			Oscillator crystal or external clock input.
OSC1				I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; analog otherwise.
CLKI				I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
RA7				I/O	TTL	General purpose I/O pin.
OSC2/CLKO/RA6	14	33	31			Oscillator crystal or clock output.
OSC2				O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO				O	—	In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6				I/O	TTL	General purpose I/O pin.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
O = Output
CMOS = CMOS compatible input or output
I = Input
P = Power

Note 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4420/4520 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTA is a bidirectional I/O port.
RA0/AN0	2	19	19			
RA0				I/O	TTL	Digital I/O.
AN0				I	Analog	Analog input 0.
RA1/AN1	3	20	20			
RA1				I/O	TTL	Digital I/O.
AN1				I	Analog	Analog input 1.
RA2/AN2/VREF-/CVREF	4	21	21			
RA2				I/O	TTL	Digital I/O.
AN2				I	Analog	Analog input 2.
VREF-				I	Analog	A/D reference voltage (low) input.
CVREF				O	Analog	Comparator reference voltage output.
RA3/AN3/VREF+	5	22	22			
RA3				I/O	TTL	Digital I/O.
AN3				I	Analog	Analog input 3.
VREF+				I	Analog	A/D reference voltage (high) input.
RA4/T0CKI/C1OUT	6	23	23			
RA4				I/O	ST	Digital I/O.
T0CKI				I	ST	Timer0 external clock input.
C1OUT				O	—	Comparator 1 output.
RA5/AN4/$\overline{SS}$/HLVDIN/C2OUT	7	24	24			
RA5				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
$\overline{SS}$				I	TTL	SPI slave select input.
HLVDIN				I	Analog	High/Low-Voltage Detect input.
C2OUT				O	—	Comparator 2 output.
RA6						See the OSC2/CLKO/RA6 pin.
RA7						See the OSC1/CLKI/RA7 pin.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
O = Output
CMOS = CMOS compatible input or output
I = Input
P = Power

Note 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4420/4520 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT0/FLT0/AN12	33	9	8			
RB0				I/O	TTL	Digital I/O.
INT0				I	ST	External interrupt 0.
FLT0				I	ST	PWM Fault input for Enhanced CCP1.
AN12				I	Analog	Analog input 12.
RB1/INT1/AN10	34	10	9			
RB1				I/O	TTL	Digital I/O.
INT1				I	ST	External interrupt 1.
AN10				I	Analog	Analog input 10.
RB2/INT2/AN8	35	11	10			
RB2				I/O	TTL	Digital I/O.
INT2				I	ST	External interrupt 2.
AN8				I	Analog	Analog input 8.
RB3/AN9/CCP2	36	12	11			
RB3				I/O	TTL	Digital I/O.
AN9				I	Analog	Analog input 9.
CCP2(1)				I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
RB4/KBI0/AN11	37	14	14			
RB4				I/O	TTL	Digital I/O.
KBI0				I	TTL	Interrupt-on-change pin.
AN11				I	Analog	Analog input 11.
RB5/KBI1/PGM	38	15	15			
RB5				I/O	TTL	Digital I/O.
KBI1				I	TTL	Interrupt-on-change pin.
PGM				I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC	39	16	16			
RB6				I/O	TTL	Digital I/O.
KBI2				I	TTL	Interrupt-on-change pin.
PGC				I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD	40	17	17			
RB7				I/O	TTL	Digital I/O.
KBI3				I	TTL	Interrupt-on-change pin.
PGD				I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
O = Output
CMOS = CMOS compatible input or output
I = Input
P = Power

Note 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4420/4520 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTC is a bidirectional I/O port.
RC0/T1OSO/T13CKI	15	34	32			
RC0				I/O	ST	Digital I/O.
T1OSO				O	—	Timer1 oscillator output.
T13CKI				I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2	16	35	35			
RC1				I/O	ST	Digital I/O.
T1OSI				I	CMOS	Timer1 oscillator input.
CCP2(2)				I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
RC2/CCP1/P1A	17	36	36			
RC2				I/O	ST	Digital I/O.
CCP1				I/O	ST	Capture 1 input/Compare 1 output/PWM 1 output.
P1A				O	—	Enhanced CCP1 output.
RC3/SCK/SCL	18	37	37			
RC3				I/O	ST	Digital I/O.
SCK				I/O	ST	Synchronous serial clock input/output for SPI™ mode.
SCL				I/O	ST	Synchronous serial clock input/output for I^2C™ mode.
RC4/SDI/SDA	23	42	42			
RC4				I/O	ST	Digital I/O.
SDI				I	ST	SPI data in.
SDA				I/O	ST	I^2C data I/O.
RC5/SDO	24	43	43			
RC5				I/O	ST	Digital I/O.
SDO				O	—	SPI data out.
RC6/TX/CK	25	44	44			
RC6				I/O	ST	Digital I/O.
TX				O	—	EUSART asynchronous transmit.
CK				I/O	ST	EUSART synchronous clock (see related RX/DT).
RC7/RX/DT	26	1	1			
RC7				I/O	ST	Digital I/O.
RX				I	ST	EUSART asynchronous receive.
DT				I/O	ST	EUSART synchronous data (see related TX/CK).

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power

Note 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4420/4520 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTD is a bidirectional I/O port or a Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when PSP module is enabled.
RD0/PSP0	19	38	38			
RD0				I/O	ST	Digital I/O.
PSP0				I/O	TTL	Parallel Slave Port data.
RD1/PSP1	20	39	39			
RD1				I/O	ST	Digital I/O.
PSP1				I/O	TTL	Parallel Slave Port data.
RD2/PSP2	21	40	40			
RD2				I/O	ST	Digital I/O.
PSP2				I/O	TTL	Parallel Slave Port data.
RD3/PSP3	22	41	41			
RD3				I/O	ST	Digital I/O.
PSP3				I/O	TTL	Parallel Slave Port data.
RD4/PSP4	27	2	2			
RD4				I/O	ST	Digital I/O.
PSP4				I/O	TTL	Parallel Slave Port data.
RD5/PSP5/P1B	28	3	3			
RD5				I/O	ST	Digital I/O.
PSP5				I/O	TTL	Parallel Slave Port data.
P1B				O	—	Enhanced CCP1 output.
RD6/PSP6/P1C	29	4	4			
RD6				I/O	ST	Digital I/O.
PSP6				I/O	TTL	Parallel Slave Port data.
P1C				O	—	Enhanced CCP1 output.
RD7/PSP7/P1D	30	5	5			
RD7				I/O	ST	Digital I/O.
PSP7				I/O	TTL	Parallel Slave Port data.
P1D				O	—	Enhanced CCP1 output.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
O = Output
CMOS = CMOS compatible input or output
I = Input
P = Power

Note 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

TABLE 1-3: PIC18F4420/4520 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTE is a bidirectional I/O port.
RE0/$\overline{RD}$/AN5	8	25	25			
RE0				I/O	ST	Digital I/O.
$\overline{RD}$				I	TTL	Read control for Parallel Slave Port (see also $\overline{WR}$ and $\overline{CS}$ pins).
AN5				I	Analog	Analog input 5.
RE1/$\overline{WR}$/AN6	9	26	26			
RE1				I/O	ST	Digital I/O.
$\overline{WR}$				I	TTL	Write control for Parallel Slave Port (see $\overline{CS}$ and $\overline{RD}$ pins).
AN6				I	Analog	Analog input 6.
RE2/$\overline{CS}$/AN7	10	27	27			
RE2				I/O	ST	Digital I/O.
$\overline{CS}$				I	TTL	Chip Select control for Parallel Slave Port (see related $\overline{RD}$ and $\overline{WR}$).
AN7				I	Analog	Analog input 7.
RE3	—	—	—	—	—	See $\overline{MCLR}$/VPP/RE3 pin.
VSS	12, 31	6, 30, 31	6, 29	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	7, 8, 28, 29	7, 28	P	—	Positive supply for logic and I/O pins.
NC	—	13	12,13, 33, 34	—	—	No connect.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
O = Output
CMOS = CMOS compatible input or output
I = Input
P = Power

Note 1: Default assignment for CCP2 when configuration bit CCP2MX is set.
2: Alternate assignment for CCP2 when configuration bit CCP2MX is cleared.

2.0 OSCILLATOR CONFIGURATIONS

2.1 Oscillator Types

PIC18F2420/2520/4420/4520 devices can be operated in ten different oscillator modes. The user can program the configuration bits, FOSC3:FOSC0, in Configuration Register 1H to select one of these ten modes:

1. LP Low-Power Crystal
2. XT Crystal/Resonator
3. HS High-Speed Crystal/Resonator
4. HSPLL High-Speed Crystal/Resonator with PLL enabled
5. RC External Resistor/Capacitor with FOSC/4 output on RA6
6. RCIO External Resistor/Capacitor with I/O on RA6
7. INTIO1 Internal Oscillator with FOSC/4 output on RA6 and I/O on RA7
8. INTIO2 Internal Oscillator with I/O on RA6 and RA7
9. EC External Clock with FOSC/4 output
10. ECIO External Clock with I/O on RA6

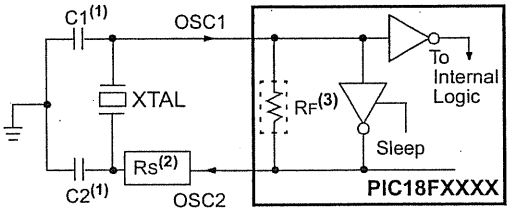
2.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2-1 shows the pin connections.

The oscillator design requires the use of a parallel cut crystal.

Note: Use of a series cut crystal may give a frequency out of the crystal manufacturer's specifications.

FIGURE 2-1: CRYSTAL/CERAMIC RESONATOR OPERATION (XT, LP, HS OR HSPLL CONFIGURATION)



Note 1: See Table 2-1 and Table 2-2 for initial values of C1 and C2.

2: A series resistor (RS) may be required for AT strip cut crystals.

3: RF varies with the oscillator mode chosen.

TABLE 2-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS

Typical Capacitor Values Used:			
Mode	**Freq**	**OSC1**	**OSC2**
XT	3.58 MHz	15 pF	15 pF
	4.19 MHz	15 pF	15 pF
	4 MHz	30 pF	30 pF
	4 MHz	50 pF	50 pF

Capacitor values are for design guidance only.

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes following Table 2-2 for additional information.

Note: When using resonators with frequencies above 3.5 MHz, the use of HS mode, rather than XT mode, is recommended. HS mode may be used at any VDD for which the controller is rated. If HS is selected, it is possible that the gain of the oscillator will overdrive the resonator. Therefore, a series resistor should be placed between the OSC2 pin and the resonator. As a good starting point, the recommended value of RS is 330Ω.

TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq	Typical Capacitor Values Tested:	
		C1	C2
LP	32 kHz	30 pF	30 pF
XT	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	10 MHz	15 pF	15 pF
	20 MHz	15 pF	15 pF
	25 MHz	0 pF	5 pF
	25 MHz	15 pF	15 pF

Capacitor values are for design guidance only.

These capacitors were tested with the crystals listed below for basic start-up and operation. **These values are not optimized.**

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes following this table for additional information.

Crystals Used:	
32 kHz	4 MHz
25 MHz	10 MHz
1 MHz	20 MHz

Note 1: Higher capacitance increases the stability of the oscillator but also increases the start-up time.

2: When operating below 3V VDD, or when using certain ceramic resonators at any voltage, it may be necessary to use the HS mode or switch to a crystal oscillator.

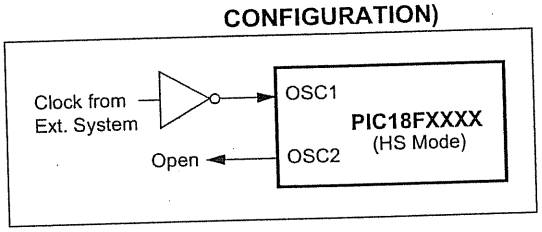
3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

4: Rs may be required to avoid overdriving crystals with low drive level specification.

5: Always verify oscillator performance over the VDD and temperature range that is expected for the application.

An external clock source may also be connected to the OSC1 pin in the HS mode, as shown in Figure 2-2.

FIGURE 2-2: EXTERNAL CLOCK INPUT OPERATION (HS OSC CONFIGURATION)

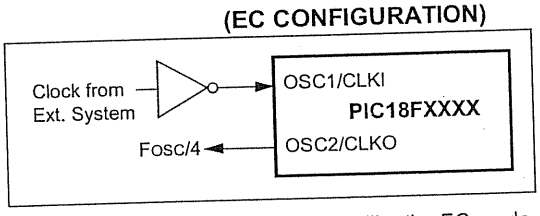


2.3 External Clock Input

The EC and ECIO Oscillator modes require an external clock source to be connected to the OSC1 pin. There is no oscillator start-up time required after a Power-on Reset or after an exit from Sleep mode.

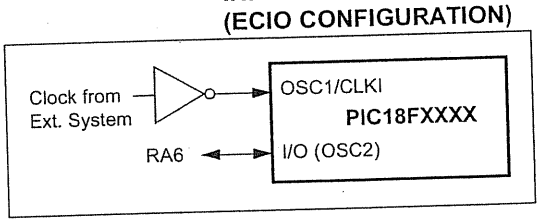
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-3 shows the pin connections for the EC Oscillator mode.

FIGURE 2-3: EXTERNAL CLOCK INPUT OPERATION (EC CONFIGURATION)



The ECIO Oscillator mode functions like the EC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6). Figure 2-4 shows the pin connections for the ECIO Oscillator mode.

FIGURE 2-4: EXTERNAL CLOCK INPUT OPERATION (ECIO CONFIGURATION)



2.4 RC Oscillator

For timing insensitive applications, the "RC" and "RCIO" device options offer additional cost savings. The actual oscillator frequency is a function of several factors:

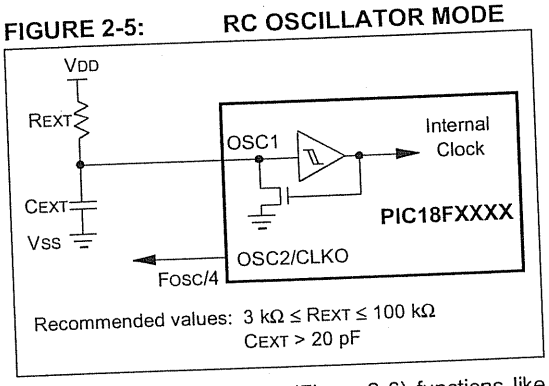
- supply voltage
- values of the external resistor (REXT) and capacitor (CEXT)
- operating temperature

Given the same device, operating voltage and temperature and component values, there will also be unit-to-unit frequency variations. These are due to factors such as:

- normal manufacturing variation
- difference in lead frame capacitance between package types (especially for low CEXT values)
- variations within the tolerance of limits of REXT and CEXT

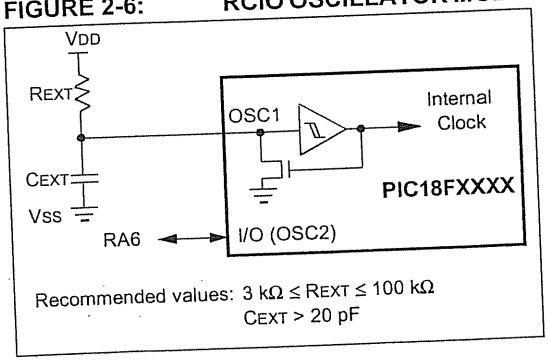
In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-5 shows how the R/C combination is connected.

FIGURE 2-5: RC OSCILLATOR MODE



The RCIO Oscillator mode (Figure 2-6) functions like the RC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

FIGURE 2-6: RCIO OSCILLATOR MODE



2.5 PLL Frequency Multiplier

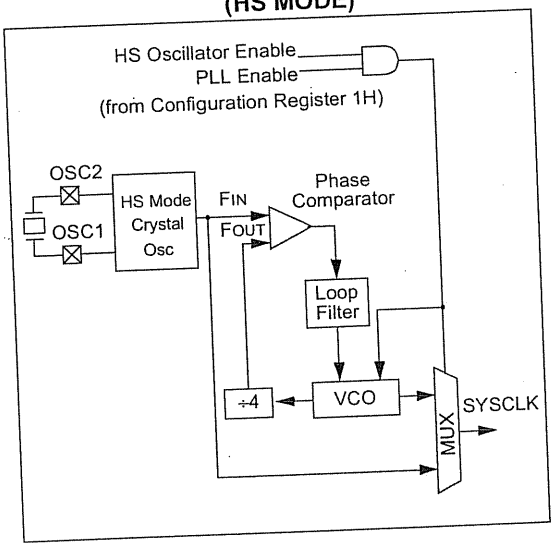
A Phase Locked Loop (PLL) circuit is provided as an option for users who wish to use a lower frequency oscillator circuit or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with EMI due to high-frequency crystals or users who require higher clock speeds from an internal oscillator.

2.5.1 HSPLL OSCILLATOR MODE

The HSPLL mode makes use of the HS mode oscillator for frequencies up to 10 MHz. A PLL then multiplies the oscillator output frequency by 4 to produce an internal clock frequency up to 40 MHz. The PLLEN bit is not available in this oscillator mode.

The PLL is only available to the crystal oscillator when the FOSC3:FOSC0 configuration bits are programmed for HSPLL mode (= 0110).

FIGURE 2-7: PLL BLOCK DIAGRAM (HS MODE)



2.5.2 PLL AND INTOSC

The PLL is also available to the internal oscillator block in selected oscillator modes. In this configuration, the PLL is enabled in software and generates a clock output of up to 32 MHz. The operation of INTOSC with the PLL is described in **Section 2.6.4 "PLL in INTOSC Modes"**.

2.6 Internal Oscillator Block

The PIC18F2420/2520/4420/4520 devices include an internal oscillator block which generates two different clock signals; either can be used as the microcontroller's clock source. This may eliminate the need for external oscillator circuits on the OSC1 and/or OSC2 pins.

The main output (INTOSC) is an 8 MHz clock source, which can be used to directly drive the device clock. It also drives a postscaler, which can provide a range of clock frequencies from 31 kHz to 4 MHz. The INTOSC output is enabled when a clock frequency from 125 kHz to 8 MHz is selected.

The other clock source is the internal RC oscillator (INTRC), which provides a nominal 31 kHz output. INTRC is enabled if it is selected as the device clock source; it is also enabled automatically when any of the following are enabled:

- Power-up Timer
- Fail-Safe Clock Monitor
- Watchdog Timer
- Two-Speed Start-up

These features are discussed in greater detail in **Section 23.0 "Special Features of the CPU"**.

The clock source frequency (INTOSC direct, INTRC direct or INTOSC postscaler) is selected by configuring the IRCF bits of the OSCCON register (page 30).

2.6.1 INTIO MODES

Using the internal oscillator as the clock source eliminates the need for up to two external oscillator pins, which can then be used for digital I/O. Two distinct configurations are available:

- In INTIO1 mode, the OSC2 pin outputs FOSC/4, while OSC1 functions as RA7 for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6, both for digital input and output.

2.6.2 INTOSC OUTPUT FREQUENCY

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 8.0 MHz.

The INTRC oscillator operates independently of the INTOSC source. Any changes in INTOSC across voltage and temperature are not necessarily reflected by changes in INTRC and vice versa.

2.6.3 OSCTUNE REGISTER

The internal oscillator's output has been calibrated at the factory but can be adjusted in the user's application. This is done by writing to the OSCTUNE register (Register 2-1).

When the OSCTUNE register is modified, the INTOSC frequency will begin shifting to the new frequency. The INTRC clock will reach the new frequency within 8 clock cycles (approximately 8 * 32 µs = 256 µs). The INTOSC clock will stabilize within 1 ms. Code execution continues during this shift. There is no indication that the shift has occurred.

The OSCTUNE register also implements the INTSRC and PLLEN bits, which control certain features of the internal oscillator block. The INTSRC bit allows users to select which internal oscillator provides the clock source when the 31 kHz frequency option is selected. This is covered in greater detail in **Section 2.7.1 "Oscillator Control Register"**.

The PLLEN bit controls the operation of the frequency multiplier, PLL, in internal oscillator modes.

2.6.4 PLL IN INTOSC MODES

The 4x frequency multiplier can be used with the internal oscillator block to produce faster device clock speeds than are normally possible with an internal oscillator. When enabled, the PLL produces a clock speed of up to 32 MHz.

Unlike HSPLL mode, the PLL is controlled through software. The control bit, PLLEN (OSCTUNE<6>), is used to enable or disable its operation.

The PLL is available when the device is configured to use the internal oscillator block as its primary clock source (FOSC3:FOSC0 = 1001 or 1000). Additionally, the PLL will only function when the selected output frequency is either 4 MHz or 8 MHz (OSCCON<6:4> = 111 or 110). If both of these conditions are not met, the PLL is disabled.

The PLLEN control bit is only functional in those internal oscillator modes where the PLL is available. In all other modes, it is forced to '0' and is effectively unavailable.

2.6.5 INTOSC FREQUENCY DRIFT

The factory calibrates the internal oscillator block output (INTOSC) for 8 MHz. However, this frequency may drift as VDD or temperature changes, which can affect the controller operation in a variety of ways. It is possible to adjust the INTOSC frequency by modifying the value in the OSCTUNE register. This has no effect on the INTRC clock source frequency.

Tuning the INTOSC source requires knowing when to make the adjustment, in which direction it should be made and in some cases, how large a change is needed. Three compensation techniques are discussed in **Section 2.6.5.1 "Compensating with the USART"**, **Section 2.6.5.2 "Compensating with the Timers"** and **Section 2.6.5.3 "Compensating with the CCP Module in Capture Mode"**, but other techniques may be used.

REGISTER 2-1: OSCTUNE: OSCILLATOR TUNING REGISTER

R/W-0	R/W-0(1)	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INTSRC	PLLEN(1)	—	TUN4	TUN3	TUN2	TUN1	TUN0
bit 7							bit 0

bit 7 **INTSRC:** Internal Oscillator Low-Frequency Source Select bit
1 = 31.25 kHz device clock derived from 8 MHz INTOSC source (divide-by-256 enabled)
0 = 31 kHz device clock derived directly from INTRC internal oscillator

bit 6 **PLLEN:** Frequency Multiplier PLL for INTOSC Enable bit(1)
1 = PLL enabled for INTOSC (4 MHz and 8 MHz only)
0 = PLL disabled

Note 1: Available only in certain oscillator configurations; otherwise, this bit is unavailable and reads as '0'. See **Section 2.6.4 "PLL in INTOSC Modes"** for details.

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

bit 4-0 **TUN4:TUN0:** Frequency Tuning bits
01111 = Maximum frequency
•
•
•
00001
00000 = Center frequency. Oscillator module is running at the calibrated frequency.
11111
•
•
•
10000 = Minimum frequency

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.6.5.1 Compensating with the USART

An adjustment may be required when the USART begins to generate framing errors or receives data with errors while in Asynchronous mode. Framing errors indicate that the device clock frequency is too high; to adjust for this, decrement the value in OSCTUNE to reduce the clock frequency. On the other hand, errors in data may suggest that the clock speed is too low; to compensate, increment OSCTUNE to increase the clock frequency.

2.6.5.2 Compensating with the Timers

This technique compares device clock speed to some reference clock. Two timers may be used; one timer is clocked by the peripheral clock, while the other is clocked by a fixed reference source, such as the Timer1 oscillator.

Both timers are cleared, but the timer clocked by the reference generates interrupts. When an interrupt occurs, the internally clocked timer is read and both timers are cleared. If the internally clocked timer value is greater than expected, then the internal oscillator block is running too fast. To adjust for this, decrement the OSCTUNE register.

2.6.5.3 Compensating with the CCP Module in Capture Mode

A CCP module can use free running Timer1 (or Timer3), clocked by the internal oscillator block and an external event with a known period (i.e., AC power frequency). The time of the first event is captured in the CCPRxH:CCPRxL registers and is recorded for use later. When the second event causes a capture, the time of the first event is subtracted from the time of the second event. Since the period of the external event is known, the time difference between events can be calculated.

If the measured time is much greater than the calculated time, the internal oscillator block is running too fast; to compensate, decrement the OSCTUNE register. If the measured time is much less than the calculated time, the internal oscillator block is running too slow; to compensate, increment the OSCTUNE register.

4.2 Master Clear (MCLR)

The MCLR pin provides a method for triggering an external Reset of the device. A Reset is generated by holding the pin low. These devices have a noise filter in the MCLR Reset path which detects and ignores small pulses.

The MCLR pin is not driven low by any internal Resets, including the WDT.

In PIC18F2420/2520/4420/4520 devices, the MCLR input can be disabled with the MCLRE configuration bit. When MCLR is disabled, the pin becomes a digital input. See **Section 10.5 "PORTE, TRISE and LATE Registers"** for more information.

4.3 Power-on Reset (POR)

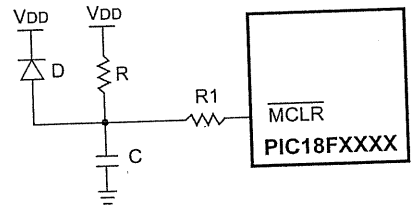
A Power-on Reset pulse is generated on-chip whenever VDD rises above a certain threshold. This allows the device to start in the initialized state when VDD is adequate for operation.

To take advantage of the POR circuitry, tie the MCLR pin through a resistor (1 kΩ to 10 kΩ) to VDD. This will eliminate external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for VDD is specified (parameter D004). For a slow rise time, see Figure 4-2.

When the device starts normal operation (i.e., exits the Reset condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in Reset until the operating conditions are met.

POR events are captured by the POR bit (RCON<1>). The state of the bit is set to '0' whenever a POR occurs; it does not change for any other Reset event. POR is not reset to '1' by any hardware event. To capture multiple events, the user manually resets the bit to '1' in software following any POR.

FIGURE 4-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)



Note 1: External Power-on Reset circuit is required only if the VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.

3: R1 ≥ 1 kΩ will limit any current flowing into MCLR from external capacitor C, in the event of MCLR/VPP pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

4.0 RESET

The PIC18F2420/2520/4420/4520 devices differentiate between various kinds of Reset:

a) Power-on Reset (POR)
b) $\overline{\text{MCLR}}$ Reset during normal operation
c) $\overline{\text{MCLR}}$ Reset during power managed modes
d) Watchdog Timer (WDT) Reset (during execution)
e) Programmable Brown-out Reset (BOR)
f) RESET Instruction
g) Stack Full Reset
h) Stack Underflow Reset

This section discusses Resets generated by $\overline{\text{MCLR}}$, POR and BOR and covers the operation of the various start-up timers. Stack Reset events are covered in **Section 5.1.2.4 "Stack Full and Underflow Resets"**. WDT Resets are covered in **Section 23.2 "Watchdog Timer (WDT)"**.

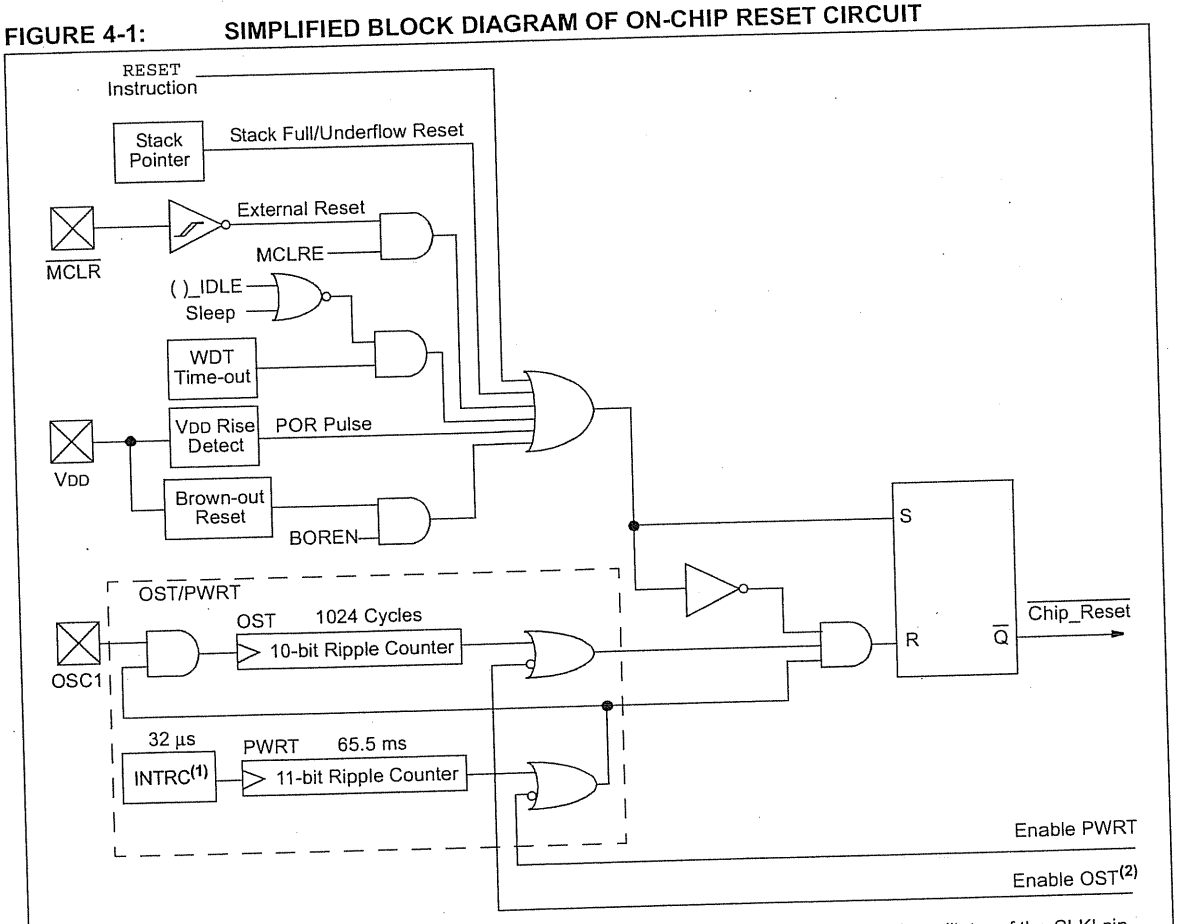
A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 4-1.

4.1 RCON Register

Device Reset events are tracked through the RCON register (Register 4-1). The lower five bits of the register indicate that a specific Reset event has occurred. In most cases, these bits can only be cleared by the event and must be set by the application after the event. The state of these flag bits, taken together, can be read to indicate the type of Reset that just occurred. This is described in more detail in **Section 4.6 "Reset State of Registers"**.

The RCON register also has control bits for setting interrupt priority (IPEN) and software control of the BOR (SBOREN). Interrupt priority is discussed in **Section 9.0 "Interrupts"**. BOR is covered in **Section 4.4 "Brown-out Reset (BOR)"**.

FIGURE 4-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



Note 1: This is the INTRC source from the internal oscillator block and is separate from the RC oscillator of the CLKI pin.
2: See Table 4-2 for time-out situations.

REGISTER 4-1: RCON REGISTER

R/W-0	R/W-1(1)	U-0	R/W-1	R-1	R-1	R/W-0(2)	R/W-0
IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

bit 6 **SBOREN:** BOR Software Enable bit(1)
If BOREN1:BOREN0 = 01:
1 = BOR is enabled
0 = BOR is disabled
If BOREN1:BOREN0 = 00, 10 or 11:
Bit is disabled and read as '0'.

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

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
1 = The RESET instruction was not executed (set by firmware only)
0 = The RESET instruction was executed causing a device Reset (must be set in software after a Brown-out Reset occurs)

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
1 = Set by power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
1 = Set by power-up or by the CLRWDT instruction
0 = Set by execution of the SLEEP instruction

bit 1 **$\overline{POR}$:** Power-on Reset Status bit(2)
1 = A Power-on Reset has not occurred (set by firmware only)
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
1 = A Brown-out Reset has not occurred (set by firmware only)
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

Note 1: If SBOREN is enabled, its Reset state is '1'; otherwise, it is '0'.
2: The actual Reset value of $\overline{POR}$ is determined by the type of device Reset. See the notes following this register and **Section 4.6 "Reset State of Registers"** for additional information.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note 1: It is recommended that the $\overline{POR}$ bit be set after a Power-on Reset has been detected so that subsequent Power-on Resets may be detected.
2: Brown-out Reset is said to have occurred when $\overline{BOR}$ is '0' and $\overline{POR}$ is '1' (assuming that $\overline{POR}$ was set to '1' by software immediately after $\overline{POR}$).

11.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register (Register 11-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 11-1. Figure 11-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT:** Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0:** Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

11.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register (Register 11-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 11-1. Figure 11-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT:** Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0:** Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

11.1 Timer0 Operation

Timer0 can operate as either a timer or a counter; the mode is selected with the T0CS bit (T0CON<5>). In Timer mode (T0CS = 0), the module increments on every clock by default unless a different prescaler value is selected (see **Section 11.3 "Prescaler"**). If the TMR0 register is written to, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

The Counter mode is selected by setting the T0CS bit (= 1). In this mode, Timer0 increments either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (T0CON<4>); clearing this bit selects the rising edge. Restrictions on the external clock input are discussed below.

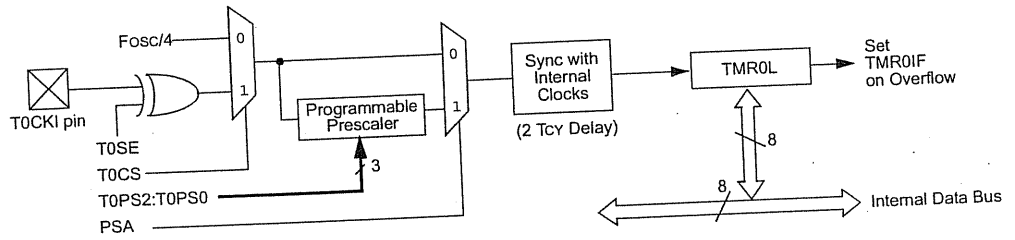
An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the internal phase clock (Tosc). There is a delay between synchronization and the onset of incrementing the timer/counter.

11.2 Timer0 Reads and Writes in 16-Bit Mode

TMR0H is not the actual high byte of Timer0 in 16-bit mode; it is actually a buffered version of the real high byte of Timer0 which is not directly readable nor writable (refer to Figure 11-2). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte were valid, due to a rollover between successive reads of the high and low byte.

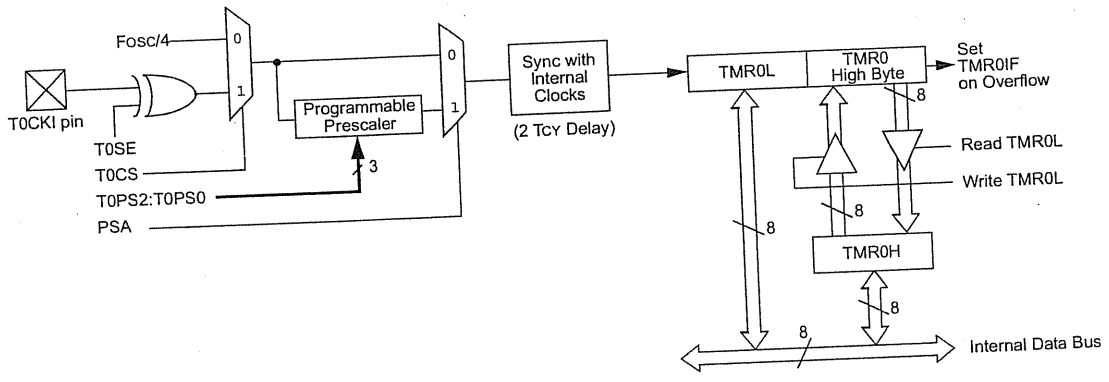
Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

FIGURE 11-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

FIGURE 11-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

12.0 TIMER1 MODULE

The Timer1 timer/counter module incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR1H and TMR1L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Reset on CCP Special Event Trigger
- Device clock status flag (T1RUN)

A simplified block diagram of the Timer1 module is shown in Figure 12-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 12-2.

The module incorporates its own low-power oscillator to provide an additional clocking option. The Timer1 oscillator can also be used as a low-power clock source for the microcontroller in power managed operation.

Timer1 can also be used to provide Real-Time Clock (RTC) functionality to applications with only a minimal addition of external components and code overhead.

Timer1 is controlled through the T1CON Control register (Register 12-1). It also contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

REGISTER 12-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of TImer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

bit 6 **T1RUN:** Timer1 System Clock Status bit
1 = Device clock is derived from Timer1 oscillator
0 = Device clock is derived from another source

bit 5-4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled
0 = Timer1 oscillator is shut off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T13CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

12.1 Timer1 Operation

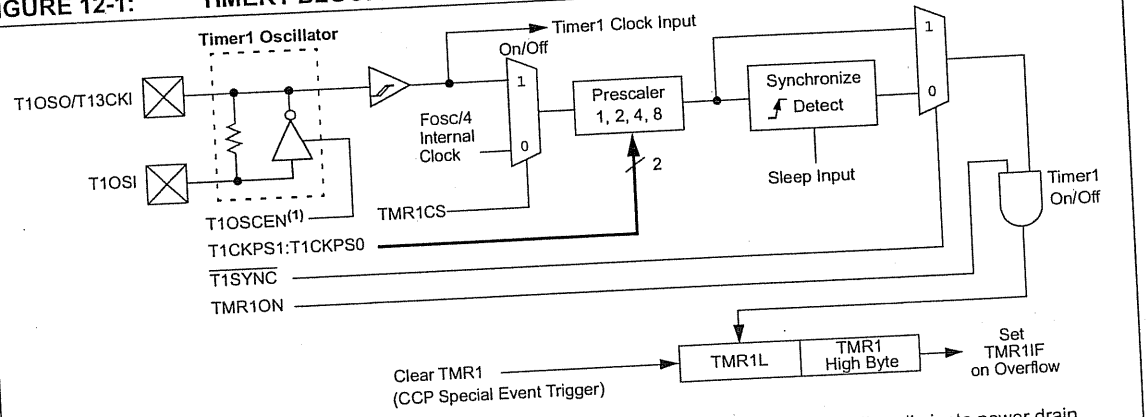
Timer1 can operate in one of these modes:

- Timer
- Synchronous Counter
- Asynchronous Counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>). When TMR3CS is cleared (= 0), Timer1 increments on every internal instruction cycle (Fosc/4). When the bit is set, Timer1 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

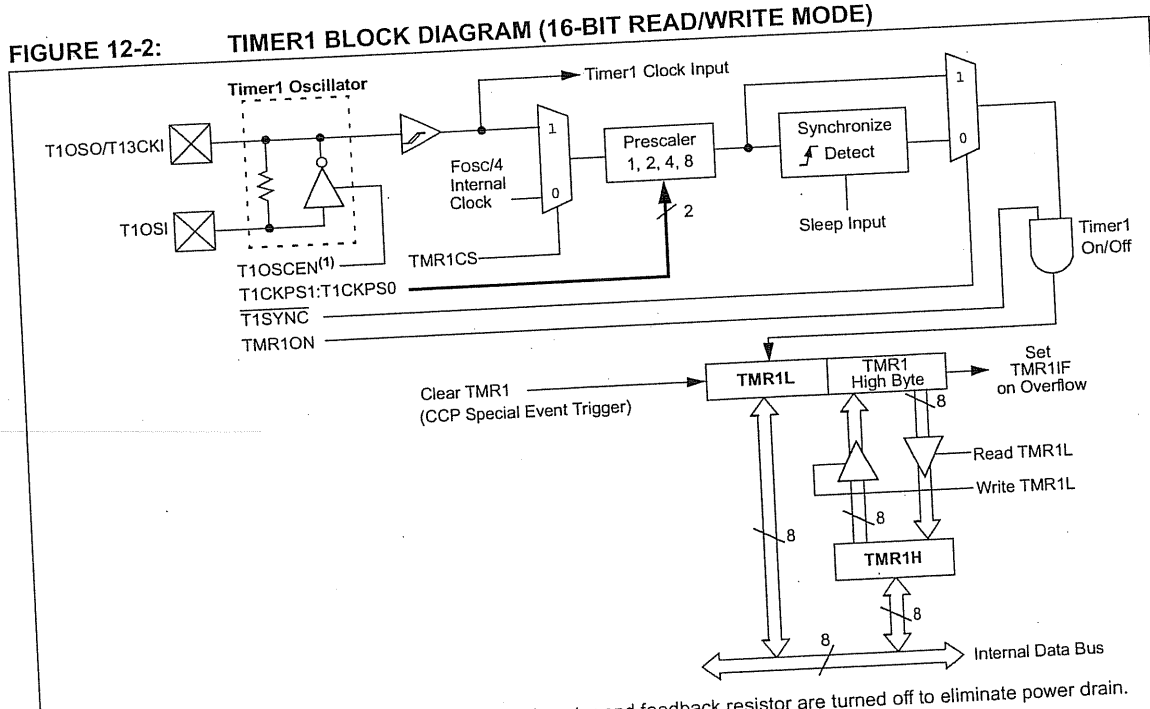
When Timer1 is enabled, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

FIGURE 12-1: TIMER1 BLOCK DIAGRAM



Note 1: When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

FIGURE 12-2: TIMER1 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)



Note 1: When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

Timer1 Operation

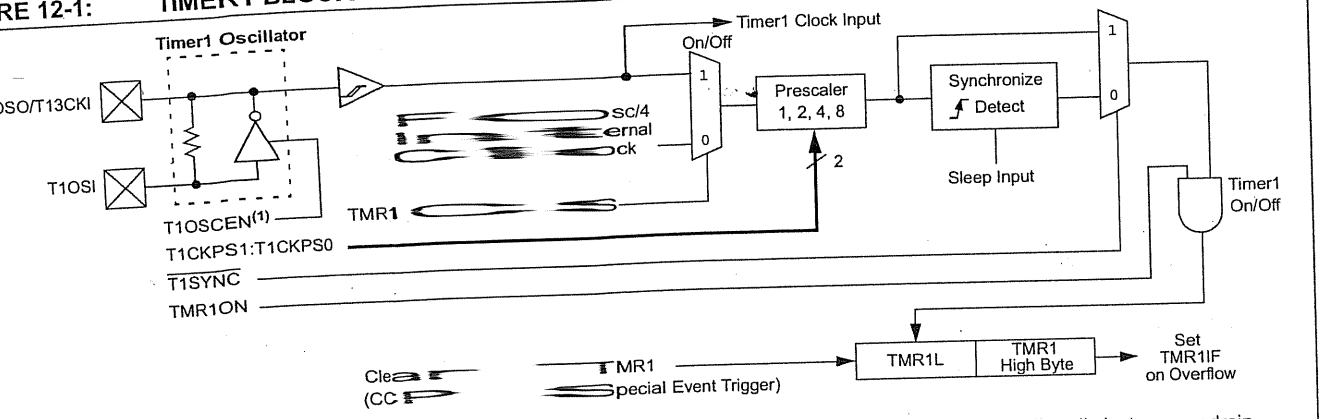
can operate in one of these modes:

hronous Counter
chronous Counter

erating mode is determined by the clo select
R1CS (T1CON<1>). When TMR3CS is cleared
Timer1 increments on every internal in truction cycle (Fosc/4). When the bit is set, Timer1 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

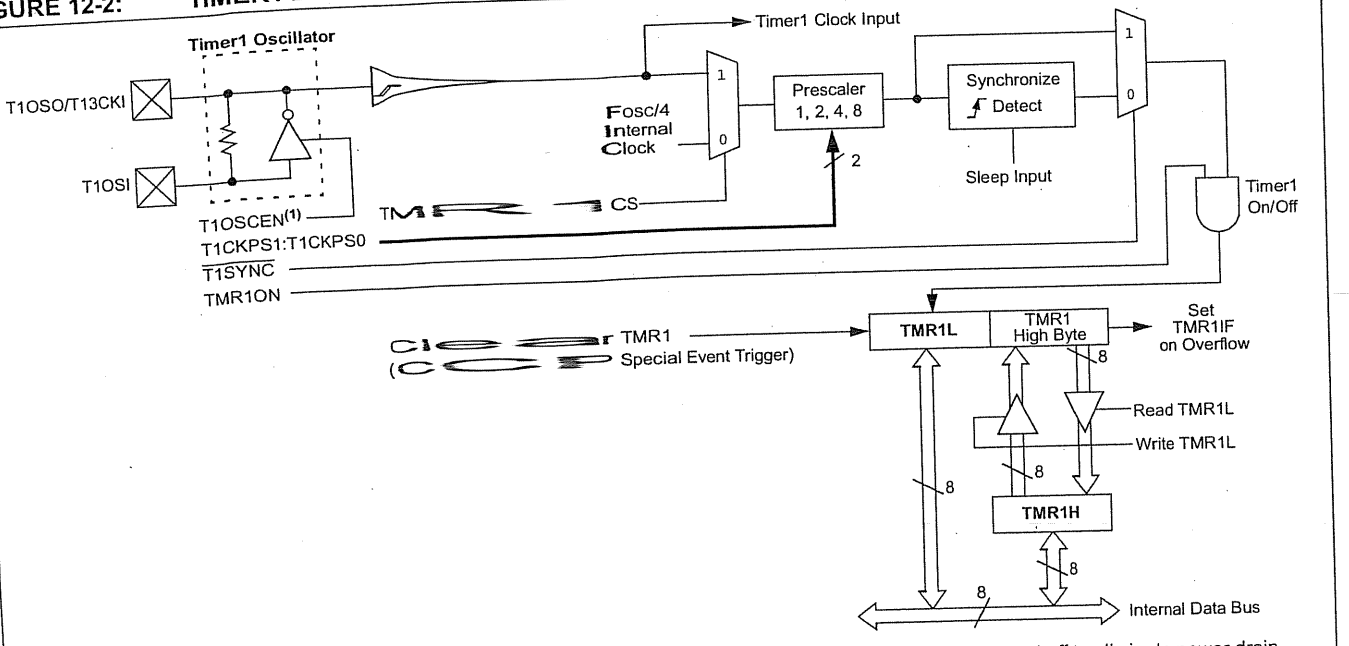
When Timer1 is enabled, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

RE 12-1: TIMER1 BLOCK DIA GRAM



ote 1: When enable bit, T1OSCEN, is clear , the inverter and feedback resistor are turned off to eliminate power drain.

GURE 12-2: TIMER1 BLOCK D AGRAM (16-BIT READ/WRITE MODE)



Note 1: When enable bit, T1OSCEN, is cl red, the inverter and feedback resistor are turned off to eliminate power drain.

12.2 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 12-2). When the RD16 control bit (T1CON<7>) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 high byte buffer. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, has become invalid due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H Buffer register. The Timer1 high byte is updated with the contents of TMR1H when a write occurs to TMR1L. This allows a user to write all 16 bits to both the high and low bytes of Timer1 at once.

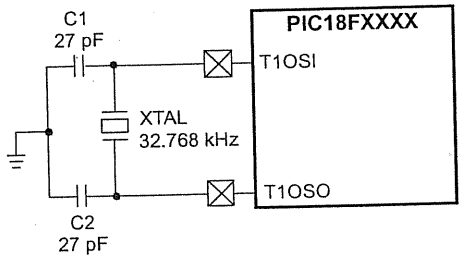
The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

12.3 Timer1 Oscillator

An on-chip crystal oscillator circuit is incorporated between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting the Timer1 Oscillator Enable bit, T1OSCEN (T1CON<3>). The oscillator is a low-power circuit rated for 32 kHz crystals. It will continue to run during all power managed modes. The circuit for a typical LP oscillator is shown in Figure 12-3. Table 12-1 shows the capacitor selection for the Timer1 oscillator.

The user must provide a software time delay to ensure proper start-up of the Timer1 oscillator.

FIGURE 12-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR



Note: See the Notes with Table 12-1 for additional information about capacitor selection.

TABLE 12-1: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR

Osc Type	Freq	C1	C2
LP	32 kHz	27 pF(1)	27 pF(1)

Note 1: Microchip suggests these values as a starting point in validating the oscillator circuit.

2: Higher capacitance increases the stability of the oscillator but also increases the start-up time.

3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

4: Capacitor values are for design guidance only.

12.3.1 USING TIMER1 AS A CLOCK SOURCE

The Timer1 oscillator is also available as a clock source in power managed modes. By setting the clock select bits, SCS1:SCS0 (OSCCON<1:0>), to '01', the device switches to SEC_RUN mode; both the CPU and peripherals are clocked from the Timer1 oscillator. If the IDLEN bit (OSCCON<7>) is cleared and a SLEEP instruction is executed, the device enters SEC_IDLE mode. Additional details are available in **Section 3.0 "Power Managed Modes"**.

Whenever the Timer1 oscillator is providing the clock source, the Timer1 system clock status flag, T1RUN (T1CON<6>), is set. This can be used to determine the controller's current clocking mode. It can also indicate the clock source being currently used by the Fail-Safe Clock Monitor. If the Clock Monitor is enabled and the Timer1 oscillator fails while providing the clock, polling the T1RUN bit will indicate whether the clock is being provided by the Timer1 oscillator or another source.

12.3.2 LOW-POWER TIMER1 OPTION

The Timer1 oscillator can operate at two distinct levels of power consumption based on device configuration. When the LPT1OSC configuration bit is set, the Timer1 oscillator operates in a low-power mode. When LPT1OSC is not set, Timer1 operates at a higher power level. Power consumption for a particular mode is relatively constant, regardless of the device's operating mode. The default Timer1 configuration is the higher power mode.

As the low-power Timer1 mode tends to be more sensitive to interference, high noise environments may cause some oscillator instability. The low-power option is, therefore, best suited for low noise applications where power conservation is an important design consideration.

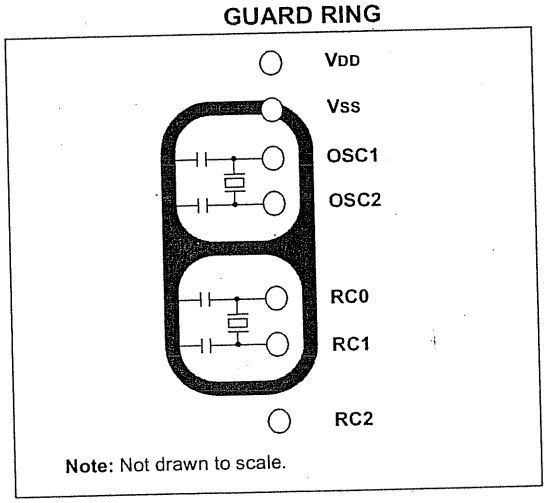
12.3.3 TIMER1 OSCILLATOR LAYOUT CONSIDERATIONS

The Timer1 oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity.

The oscillator circuit, shown in Figure 12-3, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than VSS or VDD.

If a high-speed circuit must be located near the oscillator (such as the CCP1 pin in Output Compare or PWM mode, or the primary oscillator using the OSC2 pin), a grounded guard ring around the oscillator circuit, as shown in Figure 12-4, may be helpful when used on a single-sided PCB or in addition to a ground plane.

FIGURE 12-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING



Note: Not drawn to scale.

12.4 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled or disabled by setting or clearing the Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

12.5 Resetting Timer1 Using the CCP Special Event Trigger

If either of the CCP modules is configured to use Timer1 and generate a Special Event Trigger in Compare mode (CCP1M3:CCP1M0 or CCP2M3:CCP2M0 = 1011), this signal will reset Timer1. The trigger from CCP2 will also start an A/D conversion if the A/D module is enabled (see **Section 15.3.4 "Special Event Trigger"** for more information).

The module must be configured as either a timer or a synchronous counter to take advantage of this feature. When used this way, the CCPRH:CCPRL register pair effectively becomes a period register for Timer1.

If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a special Event Trigger, the write operation will take precedence.

Note:	The Special Event Triggers from the CCP2 module will not set the TMR1IF interrupt flag bit (PIR1<0>).

12.6 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 12.3 "Timer1 Oscillator"** above) gives users the option to include RTC functionality to their applications. This is accomplished with an inexpensive watch crystal to provide an accurate time base and several lines of application code to calculate the time. When operating in Sleep mode and using a battery or supercapacitor as a power source, it can completely eliminate the need for a separate RTC device and battery backup.

The application code routine, `RTCisr`, shown in Example 12-1, demonstrates a simple method to increment a counter at one-second intervals using an Interrupt Service Routine. Incrementing the TMR1 register pair to overflow triggers the interrupt and calls the routine, which increments the seconds counter by one; additional counters for minutes and hours are incremented as the previous counter overflow.

Since the register pair is 16 bits wide, counting up to overflow the register directly from a 32.768 kHz clock would take 2 seconds. To force the overflow at the required one-second intervals, it is necessary to preload it; the simplest method is to set the MSb of TMR1H with a `BSF` instruction. Note that the TMR1L register is never preloaded or altered; doing so may introduce cumulative error over many cycles.

For this method to be accurate, Timer1 must operate in Asynchronous mode and the Timer1 overflow interrupt must be enabled (PIE1<0> = 1), as shown in the routine, `RTCinit`. The Timer1 oscillator must also be enabled and running at all times.

EXAMPLE 12-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE

```
RTCinit
        MOVLW   80h             ; Preload TMR1 register pair
        MOVWF   TMR1H           ; for 1 second overflow
        CLRF    TMR1L
        MOVLW   b'00001111'     ; Configure for external clock,
        MOVWF   T1CON           ; Asynchronous operation, external oscillator
        CLRF    secs            ; Initialize timekeeping registers
        CLRF    mins            ;
        MOVLW   .12
        MOVWF   hours
        BSF     PIE1, TMR1IE    ; Enable Timer1 interrupt
        RETURN
RTCisr
        BSF     TMR1H, 7        ; Preload for 1 sec overflow
        BCF     PIR1, TMR1IF    ; Clear interrupt flag
        INCF    secs, F         ; Increment seconds
        MOVLW   .59             ; 60 seconds elapsed?
        CPFSGT  secs
        RETURN                  ; No, done
        CLRF    secs            ; Clear seconds
        INCF    mins, F         ; Increment minutes
        MOVLW   .59             ; 60 minutes elapsed?
        CPFSGT  mins
        RETURN                  ; No, done
        CLRF    mins            ; clear minutes
        INCF    hours, F        ; Increment hours
        MOVLW   .23             ; 24 hours elapsed?
        CPFSGT  hours
        RETURN                  ; No, done
        CLRF    hours           ; Reset hours
        RETURN                  ; Done
```

TABLE 12-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
TMR1L	Timer1 Register, Low Byte								50
TMR1H	Timer1 Register, High Byte								50
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	50

Legend: Shaded cells are not used by the Timer1 module.

Note 1: These bits are unimplemented on 28-pin devices; always maintain these bits clear.

14.0 TIMER3 MODULE

The Timer3 module timer/counter incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR3H and TMR3L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Module Reset on CCP Special Event Trigger

A simplified block diagram of the Timer3 module is shown in Figure 14-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 14-2.

The Timer3 module is controlled through the T3CON register (Register 14-1). It also selects the clock source options for the CCP modules (see **Section 15.1.1 "CCP Modules and Timer Resources"** for more information).

REGISTER 14-1: T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6,3 **T3CCP2:T3CCP1:** Timer3 and Timer1 to CCPx Enable bits
1x = Timer3 is the capture/compare clock source for the CCP modules
01 = Timer3 is the capture/compare clock source for CCP2;
Timer1 is the capture/compare clock source for CCP1
00 = Timer1 is the capture/compare clock source for the CCP modules

bit 5-4 **T3CKPS1:T3CKPS0:** Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the device clock comes from Timer1/Timer3.)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T13CKI (on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

13.0 TIMER2 MODULE

The Timer2 module timer incorporates the following features:

- 8-bit timer and period registers (TMR2 and PR2, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4 and 1:16)
- Software programmable postscaler (1:1 through 1:16)
- Interrupt on TMR2-to-PR2 match
- Optional use as the shift clock for the MSSP module

The module is controlled through the T2CON register (Register 13-1), which enables or disables the timer and configures the prescaler and postscaler. Timer2 can be shut off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

A simplified block diagram of the module is shown in Figure 13-1.

13.1 Timer2 Operation

In normal operation, TMR2 is incremented from 00h on each clock (FOSC/4). A 4-bit counter/prescaler on the clock input gives direct input, divide-by-4 and divide-by-16 prescale options; these are selected by the prescaler control bits, T2CKPS1:T2CKPS0 (T2CON<1:0>). The value of TMR2 is compared to that of the period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/postscaler (see **Section 13.2 "Timer2 Interrupt"**).

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, while the PR2 register initializes at FFh. Both the prescaler and postscaler counters are cleared on the following events:

- a write to the TMR2 register
- a write to the T2CON register
- any device Reset (Power-on Reset, $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

REGISTER 13-1: 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
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

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

bit 6-3 **T2OUTPS3:T2OUTPS0**: 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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

13.2 Timer2 Interrupt

Timer2 also can generate an optional device interrupt. The Timer2 output signal (TMR2-to-PR2 match) provides the input for the 4-bit output counter/postscaler. This counter generates the TMR2 match interrupt flag which is latched in TMR2IF (PIR1<1>). The interrupt is enabled by setting the TMR2 Match Interrupt Enable bit, TMR2IE (PIE1<1>).

A range of 16 postscale options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, T2OUTPS3:T2OUTPS0 (T2CON<6:3>).

13.3 Timer2 Output

The unscaled output of TMR2 is available primarily to the CCP modules, where it is used as a time base for operations in PWM mode.

Timer2 can be optionally used as the shift clock source for the MSSP module operating in SPI mode. Additional information is provided in **Section 17.0 "Master Synchronous Serial Port (MSSP) Module"**.

FIGURE 13-1: TIMER2 BLOCK DIAGRAM

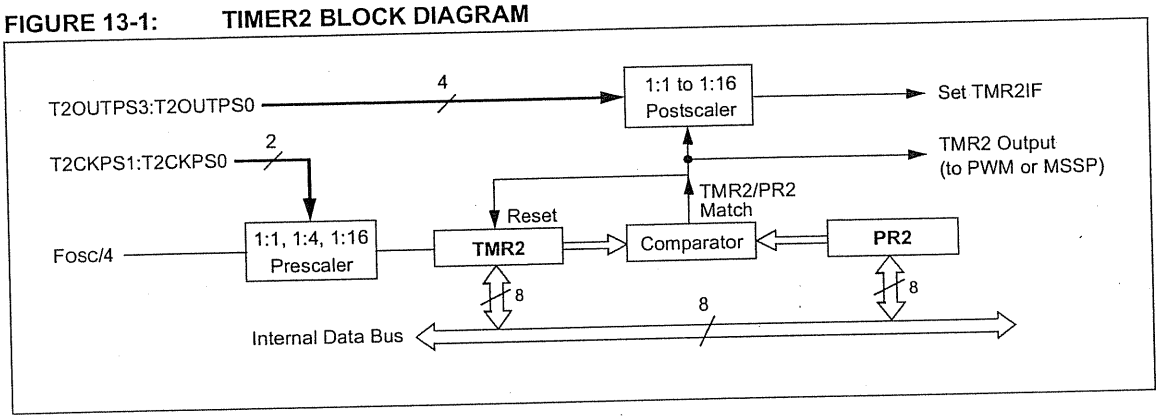


TABLE 13-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
TMR2	Timer2 Register								50
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	50
PR2	Timer2 Period Register								50

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

Note 1: These bits are unimplemented on 28-pin devices; always maintain these bits clear.