Design of a Solar Tracking System Using PIC Microcontroller

by

Rizaudin bin Ismail (1893)

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical and Electronics Engineering)

DECEMBER 2004

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Design of a Solar Tracking System Using PIC Microcontroller

by

Rizaudin bin Ismail (1893)

A project dissertation submitted to the
Electical and Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL AND ELECTRONICS ENGINEERING)

Approved by,

(Mr. Taib bin Ibrahim)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

December 2004

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done unspecified sources or persons.

RIZAUDIN BIN ISMAIL

ABSTRACT

Nowadays, emphasis is being increasingly placed on finding the alternative energy sources since we are running out of fossil-based sources such as petroleum and gas. One of these alternative energy sources is solar energy. This source of energy has unlimited potential as the sun will continue to shine for a very long time. In order to commercialize the solar energy, the efficiency of absorbing the solar energy needs to be improved. The project is concentrated in designing the system which will track the position of the sun direct to the solar panel. So, the solar panel can get better source of sun shine and will improve the efficiency of the output voltage.

To produce such system, further research need to be conducted on the current technology which to enhance the concept of solar tracking system. In this project, the system is equipped with LDR sensors, DC gear motor, and a microcontroller (PIC 16F877) chip. The heart of the system is the controller itself. It will interface the input and the output of the system as well as processing the information gained from the inputs. Knowledge of programming is important and the program will be downloaded to the microcontroller in other to run the system.

For the prototype fabrication, the student is concentrating on the mechanism used and tracking the position of the sun from the sunrise until the sunset. DC motor equipped with gear and perspex as the base of prototype are used as part of solar module mechanism in order to move the model. Testing has been done on the prototype so that the efficiency and reliability can be improved.

ACKNOWLEDGEMENT

Alhamdulillah, all praises be to Him that I have succeeded in completing my Final Year Project for two semesters. Without His blessing, the project is impossible to be completed.

First of all, the author would like to convey his highest gratitude to Mr. Taib bin Ibrahim for his guidance and attention in helping him accomplishing the project. His understanding and quality-based policy urged the author to give the best to achieve the goals of the project.

The author also would like to express deepest thanks to Mr. Zuki bin Yusoff for sharing his idea and giving suggestions to improve the project.

The compliment should also go to all Electrical and Electronics Engineering Laboratory technicians for bundles of information and assistance in completing this project especially to Ms. Siti Hawa. She helps a lot in doing the laboratory works.

Not to forget my family, course mates and friends for their cooperation and willingness in sharing and giving ideas on the project especially to Mr. Afifuddin, Mr. Ahmad Rizal, and Mr. Mohd. Hakimi. The same goes to all the people that have contributed in completing this project.

TABLE OF CONTENTS

CERTIFICATIONS .... **I**

ABSTRACT .... **I II**

ACKNOWLEDGEMENT .... **IV**

TABLE OF CONTENTS .... **V**

LIST OF TABLES .... **VII**

LIST OF FIGURES

LIST OF ABBREVIATION .... **VIII**

CHAPTER 1: INTRODUCTION **1**

1.1 BACKGROUND OF STUDY

1.1.1 Introduction to Solar Energy

1.1.2 Solar Cell and Solar Panel

1.1.3 Types of Solar Collectors 2

1.2 PROBLEM STATEMENT 3

1.3 OBJECTIVES AND SCOPE OF STUDY 4

CHAPTER 2: LITERATURE REVIEW AND THEORY **5**

2.1 HISTORY OF SOLAR TRACKING SYSTEM

2.2 WHAT IS SOLAR TRACKING SYSTEM 6

2.2.1 Electronic Solutions 7

2.2.2 Passive Solutions 8

2.3 THE DESIGN CONCEPT OF SOLAR TRACKING SYSTEM 9

2.3.1 Microcontroller (PIC16F877) 10

2.3.1.1 The Overview of File Registers

2.3.1.2 8-channel to 10-bit Analog-Digital-Converter 11

2.3.1.3 ADC Module in PIC16F877 13

2.3.1.4 Concept of Programming

2.3.2 Motor Controller 14

2.3.3 DC Gear Motor 15

2.3.4 Light Dependent Resistor (LDR) Sensor 16

2.3.4.1 Principle Operation 17

CHAPTER 3: METHODOLOGY/PROJECT WORK....................................18
3.1 PROCEDURE IDENTIFICATION
3.2 HARDWARE...19
3.2.1 Sensor's Circuit
3.2.2 Motor Controller's Circuit...20
3.3 SOFTWARE...21
3.3.1 Block Diagram of Program's Flow
3.3.2 Source Code...22
3.4 PROJECT DEVELOPMENT...24
3.4.1 Main Controller Circuit..25
3.4.2 LDR Sensor Circuit..27
3.4.2 Software (ADCON0 and ADCON1) Configuration............................28
3.4.2.1 ADCON0 Details
3.4.2.2 ADCON1 Details..30

CHAPTER 4: RESULTS AND DISCUSSION..32
4.1 LDR CIRCUIT
4.2 ANALOGUE TO DIGITAL CONVERTER..33
4.2.1 One Analogue Source
4.2.2 Two Analogue Sources..34
4.3 COMPLETE SOURCE CODE...35
4.4 PROTOTYPE FABRICATION..36
4.4 PROTOTYPE TESTING..38

CHAPTER 5: CONCLUSION AND RECOMMENDATION............................40
5.1 CONCLUSION
5.2 RECOMMENDATION..41

REFERENCES...42
APPENDICES..44

LIST OF TABLES

Table 2-1	The Useful Combination of Switches to Drive a Motor.
Table 3-1	Main Registers for Analogue Inputs.
Table 3-2	Clock Select Bits for A/D Conversion.
Table 3-3	Bits Select to Determine the A/D Conversion Input.
Table 3-4	Bits Required in ADCON0.
Table 3-5	Bit Selects to Determine Type of Input at Each Pin.
Table 3-6	Bits Required in ADCON1.
Table 4-1	The Result of the Experiment for One Analogue Input.
Table 4-2	The Result of the Experiment for Two Analogue Inputs.
Table 4-3	LEDs Representation.
Table 4-4	The Result of the First Simulation.
Table 4-5	The Result of the Second Simulation.
Table 4-6	The Result of Prototype Testing.

LIST OF FIGURES

Figure 1-1	Example of Solar Tracking System.
Figure 2-1	An Electric Solar Tracker Including a Sensor that Aims to Minimize the Angle between the Line of the Sun and a Face Perpendicular to the Panel.
Figure 2-2	Passive Solar Tracker using Two Identical Cylindrical Tubes Filled with a Fluid under Partial Pressure.
Figure 2-3	The Design Concept of the Solar Tracking System.
Figure 2-4	The Architecture of PIC16F877/874.
Figure 2-5	PIC16F877 Register Map File.
Figure 2-6	Simplified Block Diagram of the PIC16F877 ADC Module.
Figure 2-7	The Simple Conceptual Schematic of H-bridge Circuit.
Figure 2-8	Parts of an Electric Motor.
Figure 2-9	Example of LDR Sensor.

Figure 2-10 Structure of a Light Dependent Resistor, Showing Cadmium Sulphide Track and an Atom to Illustrate Electrons in the Valence and Conduction Bands.
Figure 3-1 Flow Chart for Basic Procedure Identification.
Figure 3-2 The Circuit for the LDR Sensor.
Figure 3-3 The Relays Circuit.
Figure 3-4 The Motor Controller Circuit Mounted on Veraboard.
Figure 3-5 The Block Diagram of the PIC's Operation.
Figure 3-6 The Circuit Diagram for the PIC.
Figure 3-7 The Pin Connection for Oscillator.
Figure 3-8 The Main Controller Circuit on the Veraboard.
Figure 3-9 The Circuit Diagram for the Microcontroller and LDR Sensor Circuit.
Figure 3-10 The Location of the LDR Sensor Circuits.
Figure 3-11 The Complete LDR Sensor Circuit.
Figure 4-1 The Schematic Diagram of Prototype.
Figure 4-2 Front View of the Prototype.

LIST OF ABBREVIATION

ADC	Analogue-Digital-Converter
FYP	Final Year Project
UTP	Universiti Teknologi Petronas
PIC	Programmable Integrated Circuit
PLC	Programmer Logic Controller
LDR	Light Dependent Resistor
LED	Light Emitting Diode

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF STUDY

1.1.1 Introduction to Solar Energy

Nowadays, renewable energy source plays important role in human's life due to the existing energy resources such as petroleum had been depleted [1]. The new sources of energy have been found which include wind energy, hydropower and solar energy. There are many ways and technology to produce electricity including by using natural resources like petroleum, gas and diesel. But all those resources will be depleted and also effected the environment. So, people starts doing research and development on the alternative energy. The solar energy is the one that receives most attention. It is practical and environmentally friendly in producing electricity. The solar energy also offers an endless supply and most importantly it is free.

1.1.2 Solar Cell and Solar Panel

Solar cells capture the sun's energy and convert it to electricity. Inside a solar panel, each cell contains silicon, an element found in sand that absorbs sunlight. The energy in this absorbed light produces a small electrical current. Metal grids around the solar cells direct the currents into wires that lead to the power controls [2].

The solar array is comprised of one or more solar PV modules (solar panels) which convert sunlight into clean solar electricity. PV is short for Photovoltaic, which means electricity from light. The solar modules need to be mounted facing the sun and avoiding shade for best results [2].

1.1.3 Types of Solar Collectors

There are three types of solar collectors which are flat-plate collector, focusing collector and passive collector [3].

1) *Flat-plate Collector*

   This is the most commonly used type of collectors. There are arrays of solar panels arranged in a simple plane. It can be of nearly any size, and have output that is directly related to a few variables including size, facing and cleanliness. Usually, this kind of collector has an automated machinery to keep it facing the sun. The most common Flat-plate Collector is the one used for hot water in residential area. It is also used for conducting electricity and available in stand still mode and active mode (tracking the sun).

2) *Focusing Collector*

   It is essentially flat-plane collector with optical devices arranged to maximize the radiation falling on the focus of the collector. It is currently used only in a few scattered areas. Solar furnaces are example of this type of collector. Although it can produce far greater amount of energy at a single point than the flat-plane collector, it loses some of the radiation that the flat-plane do not. The focusing collector is ideal for a solar furnace that provides contaminant-free environments for research and industrial use. Solar furnaces are well suited to the destruction of hazardous wastes. Focusing a beam of concentrated light onto hazardous wastes break down numerous toxic chemicals, including dioxin and polychlorobiphenyls (PCBs).

3) *Passive Collector*

   It is completely different from the other two collectors. The passive collector absorbs radiation and converts it to heat naturally, without being designed and built to do so. All objects have this property to some extent, but only some objects (like walls) will be able to produce enough heat to make it worthwhile. Often its natural ability to convert radiation to heat is enhanced in some way or another (by being painted black, for example) and a system for transferring the heat to a different location is generally added. The most basic type is the incidental heat trap. The idea is to allow the maximum amount of light possible inside through a window and allow it to fall on a floor made of stone or another heat holding material. During the day, the area will stay cool as the floor absorbs most of the heat, and at night, the area will stay warm as the stone re-emits the heat it absorbed during the day.

Passive collector is anything that capable to convert radiation into heat naturally including the walls. Since it is not designed to do so, passive collector is the worst in efficiency. The focusing collector instead, produces too much heat and will lose efficiency at high temperature. Due to its characteristic, focusing collector is used in industry that requires very high temperature to operate such as for detoxifying hazardous wastes. The flat-plate collector is the most commonly used solar tracker. It comes with several types and doesn't produce too much heat. It has the best efficiency compared to the other two types of solar collector.

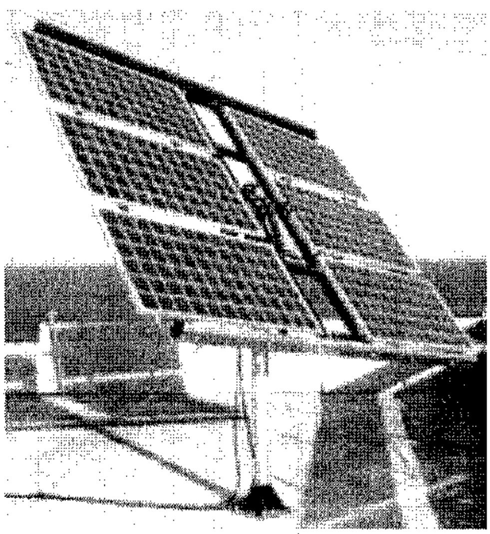


Figure 1-1: Example of Solar Tracking System

1.2 PROBLEM STATEMENT

The solar energy has been started to be commercialized since the need of alternative energy sources. The current solar tracking system is in stand-still mode regardless the position of the sun. The efficiency of the output voltage from the panel is low. To overcome the problem, the active solar tracker has been introduced where the system can follow the sun's track from it's rising to its setting. It will keep the solar panel facing the sun all the times so that the system can get better input from sun shine. Base on the research by Hamburg's SunTechnics GmbH, the energy yield can be increased by around 23% to 30% by using active solar tracker [4]. Thus, the student will design the solar tracker by using PIC microcontroller and DC gear motor to move the model.

1.3 OBJECTIVES AND SCOPE OF STUDY

The objectives of the Solar Tracking System project are to meet to following requirement:-

i. To study the Solar Tracking System.
ii. To design the Solar Tracking System by using PIC microcontroller.
iii. To fabricate the prototype of Solar Tracking System.

The system is relevant to be developed since it has the commercial value in the future. Besides, it gives a better understanding on the system and its sub-components especially the microcontroller part where the student can learn the programming language. Programming language is very important nowadays since the industry is moving toward automation process.

There are several aspects need to be focused on this project. Let divide the project into three components which are:-

i. The study on the system and its sub-components to get further knowledge and understanding.
ii. The development of the system circuitry in other to interface the input and output.
iii. The development of the assembly programming and download it to the microcontroller.

CHAPTER 2

LITERATURE REVIEW AND THEORY

2.1 HISTORY OF SOLAR TRACKING SYSTEM

Solar technology isn't new. Its history spans from the 7^{th} Century B.C. to today. People started out concentrating the sun's heat with glass and mirrors to light fires. Today, there are many type of solar technology varies from solar-powered buildings to solar powered vehicles [5].

In 1767, Swiss scientist Horace de Saussure was credited with building the world's first solar collector, later used by Sir John Herschel to cook food during his South Africa expedition in the 1830s. On September 27, 1816, Robert Stirling applied for a patent for his economizer at the Chancery in Edinburgh, Scotland. By trade, Robert Stirling was actually a minister in the Church of Scotland and he continued to give services until he was eighty-six years old. But, in his spare time, he built heat engines in his home workshop. Lord Kelvin used one of the working models during some of his university classes. This engine was later used in the dish/Stirling system, a solar thermal electric technology that concentrates the sun's thermal energy in order to produce power [5].

In 1839, French scientist Edmond Becquerel discovers the photovoltaic effect while experimenting with an electrolytic cell made up of two metal electrodes placed in an electricity-conducting solution—electricity-generation increased when exposed to light. Later in 1876, William Grylls Adams and Richard Evans Day discover that selenium produces electricity when exposed to light. Although selenium solar cells failed to convert enough sunlight to power electrical equipment, they proved that a solid material could change light into electricity without heat or moving parts [5].

In 1908, William J. Bailley of the Carnegie Steel Company invents a solar collector with copper coils and an insulated box—roughly, it's present design. Passive solar buildings in the United States were in such demand, as a result of scarce energy during the prolonged W.W.II, that Libbey-Owens-Ford Glass Company published a book entitled Your Solar House, which profiled forty-nine of the nation's greatest solar architects [5].

In 1954, Photovoltaic technology is born in the United States when Daryl Chapin, Calvin Fuller, and Gerald Pearson develop the silicon photovoltaic (PV) cell at Bell Labs—the first solar cell capable of converting enough of the sun's energy into power to run everyday electrical equipment. Bell Telephone Laboratories produced a silicon solar cell with 4% efficiency and later achieved 11% efficiency [5].

After the big break by Bell Laboratories, solar power becomes the world's fastest growing renewable energy sources. There are many types of solar technology were developed and the researcher start to concentrate on improving the efficiency of solar tracking system [5].

2.2 WHAT IS SOLAR TRACKING SYSTEM

A solar tracker strives to force sunlight to contact the PV cell normally at all times throughout the day. In other word, solar tracking system is a system that can automatically follow the path of the sun throughout the day. This results in the maximum possible energy generation. Various sensing equipment and actuating devices, both electrical and mechanical, are used in the implementation of the solar tracking system today.

Many solutions are currently on the market. The ideal tracker would allow the PV cell to accurately locate the sun, compensating for both changes in the altitude angle of the sun (throughout the day) and latitudinal offset of the sun (during seasonal changes). The slow movement of the sun requires a damped system that will also respond slowly and avoid an oscillatory movement. Other desirable aspects would include the nocturnal repositioning of the solar tracker to anticipate the alignment of sunrise, opposite to that of the previous day's sunset, reducing lost energy in the morning.

The current technology on market can be categorized as follows [6]:

2.2.1 Electronic Solutions

A common solution is that of a centrally pivoted PV cell being moved, about this pivot, by a motor linked to an electronic sensor. This is probably the simplest electronic method available, with the motor being powered by the cell itself. The panel is positioned out of reach, to avoid human interference, although leaving it more susceptible to wind drag. However, as PV power is reduced, this is not a very elegant solution.

Another type of electric solar tracker is shown in Figure 2-1, and includes a sensor that aims to minimize the angle between the line of the sun and a face perpendicular to the panel. When this angle is reduced to zero the sunlight strikes the panel at 90^{O}. Also, in this design, the central pivot was not horizontal – one bearing was nearer the ground than the other – giving the correct elevation for non-equatorial locations. Again the panel itself produced the power for the motor.

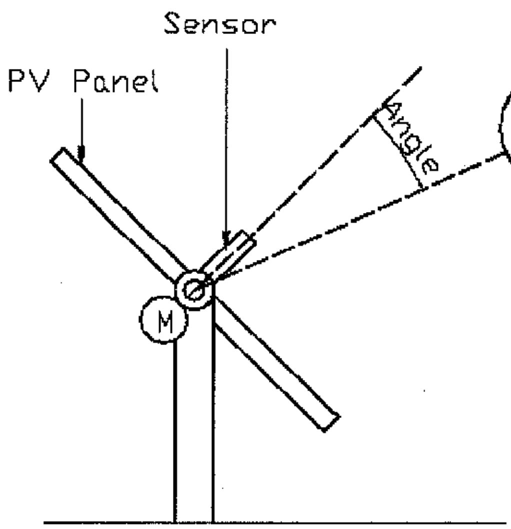


Figure 2-1: An Electric Solar Tracker Including a Sensor that Aims to Minimize the Angle between the Line of the Sun and a Face Perpendicular to the Panel.

2.2.2 Passive Solutions

Figure 2-2 shows how movement is instigated using two identical cylindrical tubes (each at either side of the panel and equal distances from the central pivot) filled with a fluid under partial pressure. Using suitably placed shades, the sun heats the fluid causing evaporation and transfer from one cylinder to the other. This mass imbalance is used to move the solar panel. Damping is used to limit the speed of movement. This simple system can be cheaply made and uses none of the PV cell's power, although it starts every day pointing in the wrong direction, losing sight of the sun as it attempts to reposition itself. Despite this draw back, it is a commonly used method.

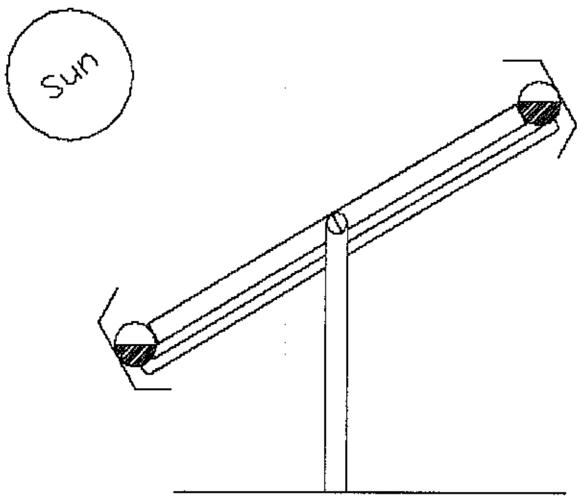


Figure 2-2: Passive Solar Tracker using Two Identical Cylindrical Tubes Filled with a Fluid under Partial Pressure.

Tests have shown that passive trackers have been found to be comparable to electrically based systems in terms of performance, and even though they are less expensive, they have not yet been widely accepted by the consumer. Base on study, the electronic solution will be used in designing the solar tracking system.

2.3 THE DESIGN CONCEPT OF SOLAR TRACKING SYSTEM

To give the brief view of the design concept, a block diagram for the system has been developed. It consists of all the sub-components of the system which are:-

i. Microcontroller (PIC 16F877)
ii. Motor Controller
iii. DC Gear Motor
iv. LDR Sensors

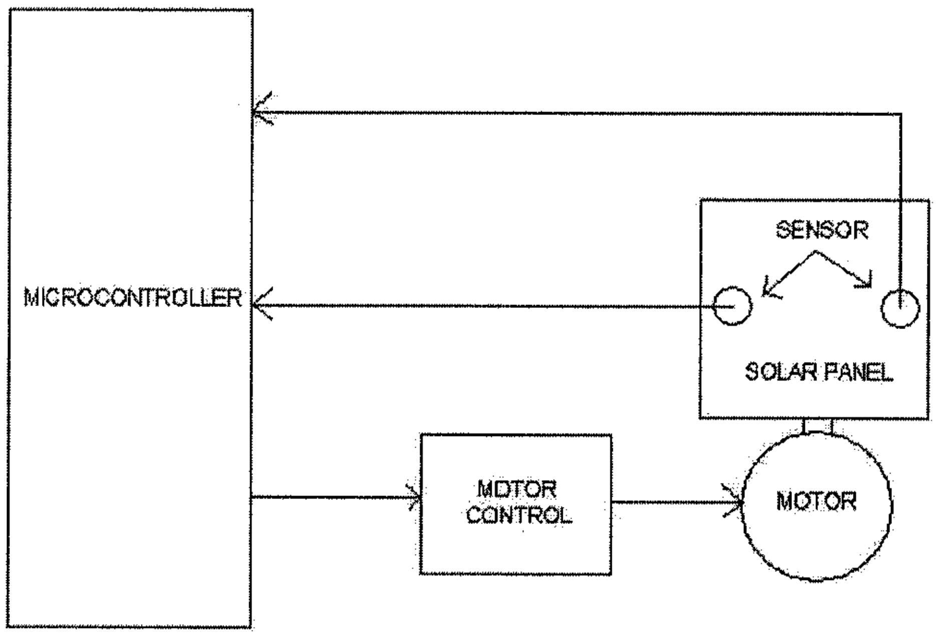


Figure 2-3: The Design Concept of the Solar Tracking System.

The heart of the system is the microcontroller. All the inputs and outputs will be connected to the microcontroller. There will be a motor controller in between the motor and the microcontroller. The microcontroller will give a signal to the motor controller. Then, the motor controller will give instruction to motor whether to rotate clockwise or counterclockwise. The solar panel will be attached to the motor gear system. And two sensors will be attached to solar panel at each edge. Microcontroller will compare the output from both sensors and will give instruction to motor control.

2.3.1 Microcontroller (PIC16F877)

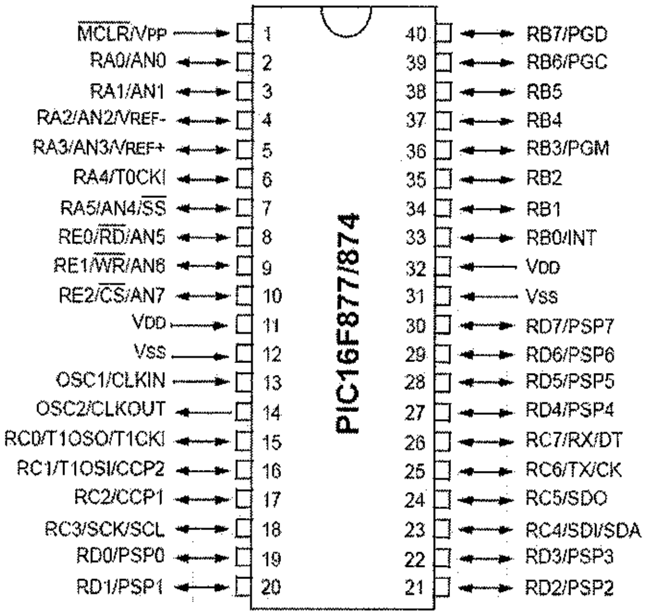


Figure 2-4: The Architecture of PIC16F877/874

PIC16F877 is a high-performance FLASH microcontroller that provides engineers with the highest design flexibility possible. In addition to 8192x14 words of FLASH program memory, 256 data memory bytes, and 368 bytes of user RAM, PIC16F877 also features an **integrated 8-channel 10-bit Analogue-to-Digital converter**. Peripherals include two 8-bit timers, one 16-bit timer, a Watchdog timer, Brown-Out-Reset (BOR), In-Circuit-Serial Programming™, RS-485 type UART for multi-drop data acquisition applications, and I2C™ or SPI™ communications capability for peripheral expansion. Precision timing interfaces are accommodated through two CCP modules and two PWM modules." [7]

2.3.1.1 The Overview of File Registers

The data memory is partitioned into multiple banks which contain the general purpose registers and the special function registers. Bits RP1 and RP0 are the bank select bits, these bits are found in the STATUS register (b6 & b5) [7].

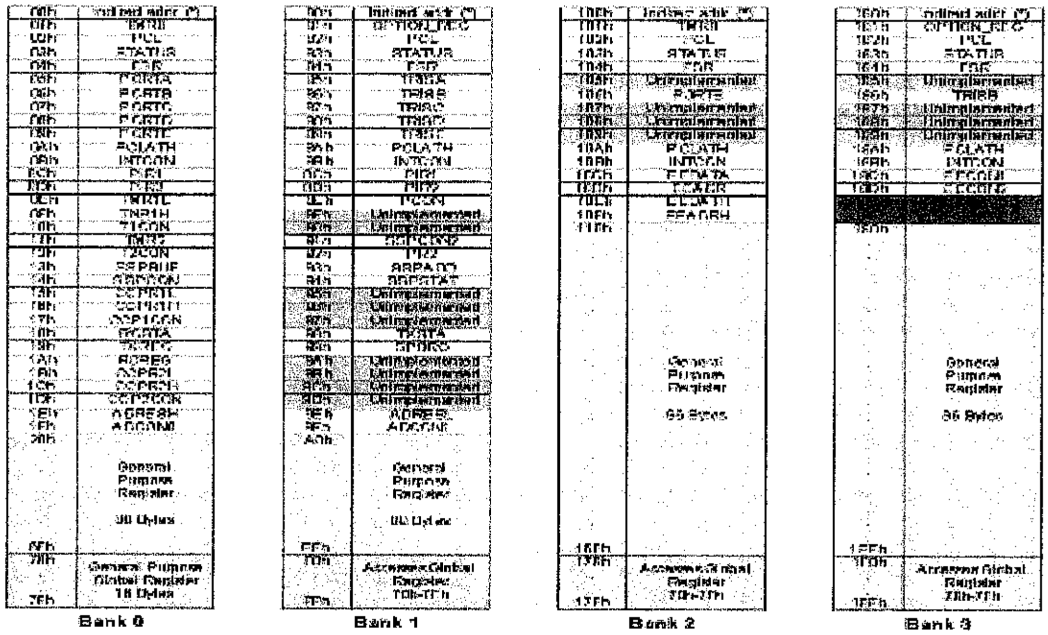


Figure 2-5: PIC16F877 Register Map File.

Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the special function registers (shown in yellow). Above the special function registers are general purpose registers (shown in blue), implemented as static RAM. All implemented banks contain special function registers. Some "high use" special function registers from one bank may be mirrored in another bank for code reduction and quicker access. Also notice that there are 16 general purpose global registers (shown in green) [7].

2.3.1.2 8-channel to 10-bit Analog-Digital-Converter

At first, it appears that the PIC16F877 has 8 built-in ADCs, but this is not the case. Figure 2-6 shows a simplified block diagram of the analogue-to-digital converter module, clearly there is only one 10-bit ADC which can be connected to only one of eight input pins at any one time [7].

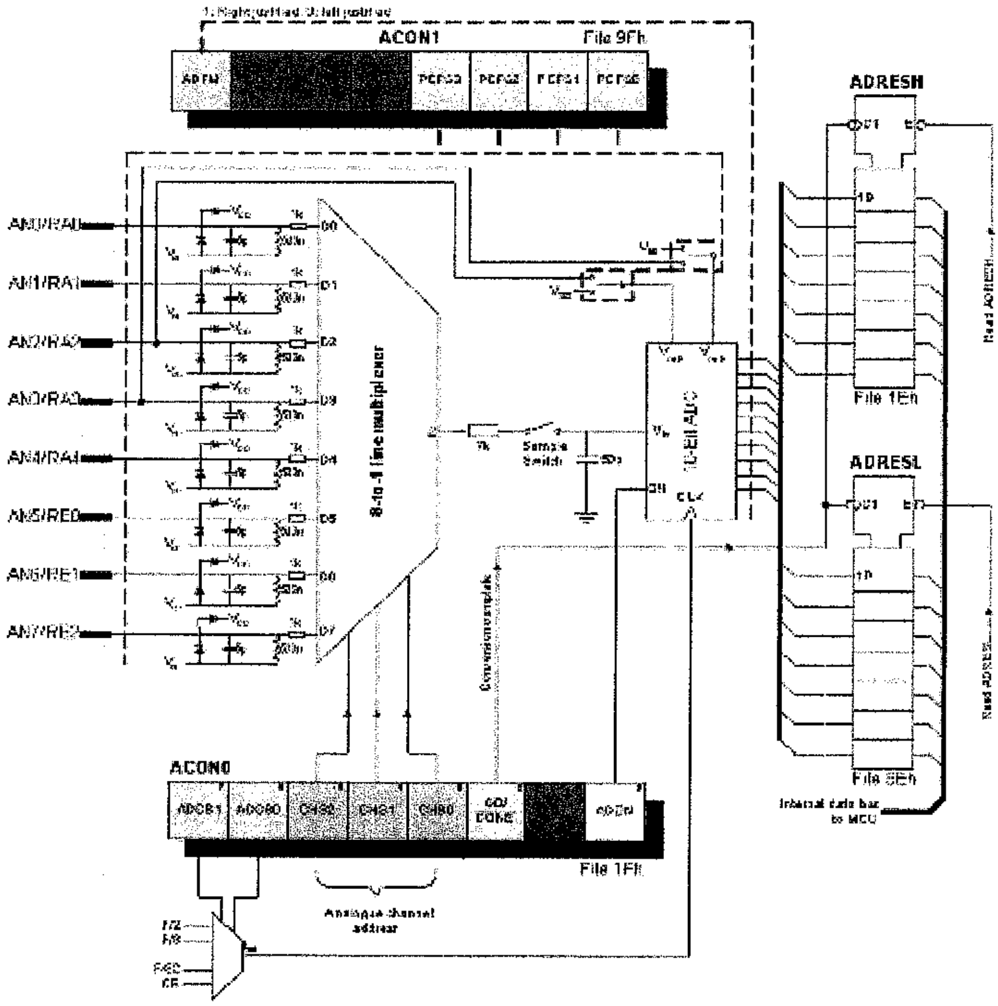


Figure 2-6: Simplified Block Diagram of the PIC16F877 ADC Module.

As shown in Figure 2-6, the input analogue channels AN4.0 are shared with port A, and channels AN7.5 are shard with port E. If less than eight analogue channels are required, then some of the pins can be assigned as digital I/O port lines using PCFG3.0 bits. For example, if PCFG3.0 = 0010 then AN4.0 are configured as analogue inputs, while AN7.5 are digital (port E free), with VDD used as the reference. The ADCON 1 register is pictured in Table 5 to configure the functions of the port pins. These port pins can be configured as analog inputs (RA3 can also be the voltage reference), or as digital I/O [7].

2.3.1.3 ADC Module in PIC16F877

To set up the PC16F877's analogue to digital channel, A/D module which has four registers need to be configured, which are [7]:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register0 (ADCON0)
- A/D Control Register1 (ADCON1)

The details of the above registers will be explained in the methodology section.

2.3.1.4 Concept of Programming

The PIC actually works on 10-bit digital number. The 10-bit A/D result is loaded onto the register pair ADRESH:ADRESL which is a 16-bit wide register pair. The A/D module gives the programmer a choice whether to left or right justify the 10-bit result into the 16-bit register (more details in PIC16F877 manual).

The A/D module has a high reference voltage of 5V and a low reference voltage of 0V, therefore, the analog input must be varied within 0-5V. An example to convert the corresponding voltage to a binary number is as shown below:-

Since the maximum voltage allowed is 5V the corresponding 10-bit binary number would:-

$11\ 1111\ 1111_2$ is equal to 1024_{10}.

This is the base ration for the next voltage value.

e.g.: Input voltage is 3.75 V.

$$3.75\text{ V} \times \frac{1024_{10}}{5\text{V}} = 768_{10}$$

Therefore, 768_{10} is equal to $11\ 0000\ 0000_2$

2.3.2 Motor Controller

H-Bridge circuit is used for rotating the motor clockwise and anti-clockwise. Here is the simple conceptual schematic:

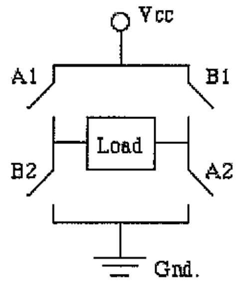


Figure 2-7: The Simple Conceptual Schematic of H-bridge Circuit.

A basic H-Bridge has 4 switches, relays, transistors, or other means of completing a circuit to drive a motor. In the above diagram, the switches are labeled A1, A2, B1, and B2. Since each of the four switches can be either open or closed, there are $2^4 =$ 16 combinations of switch settings. Many are not useful and in fact, several should be avoided since they short out the supply current (e.g., A1 and B2 both closed at the same time). There are four combinations that are useful:

Table 2-1: The Useful Combination of Switches to Drive a Motor.

Closed switches	Polarity	Effect
A1 & A2	forward	motor spins forward
B1 & B2	reverse	motor spins backward
A1 & B1	brake	motor acts as a brake
None	free	motor floats freely

2.3.3 DC Gear Motor

There are several types of electric motors, each with its own unique features and benefits. Basically, most of the electric motors are divided into three categories which are AC, DC, and stepper motor. The difference is in term of the way it generates and moves the magnetic fields. For example, magnetic fields can be generated from permanent magnets or from coils carrying current (electromagnets), and in the latter case, direct connection can deliver current to the coils, through brushes, or by induction. Some motors have permanent magnets on the rotor and coils on the stator, others have permanent magnets on the stator and coils on rotor, and still others have coils and both rotor and stator.

In this project, it was decided to use DC gear motor. DC gear motors operate from a direct current power source. In general, users select brush-type DC motors when low system cost in a priority, and brushless motors to fulfill other requirement (such as maintenance-free operation, high speeds, and explosive environments where sparking could be hazardous). Because varying the voltage and current from the power supply are controlled by speed and torque (twisting force), DC motors work well in complex motion tasks.

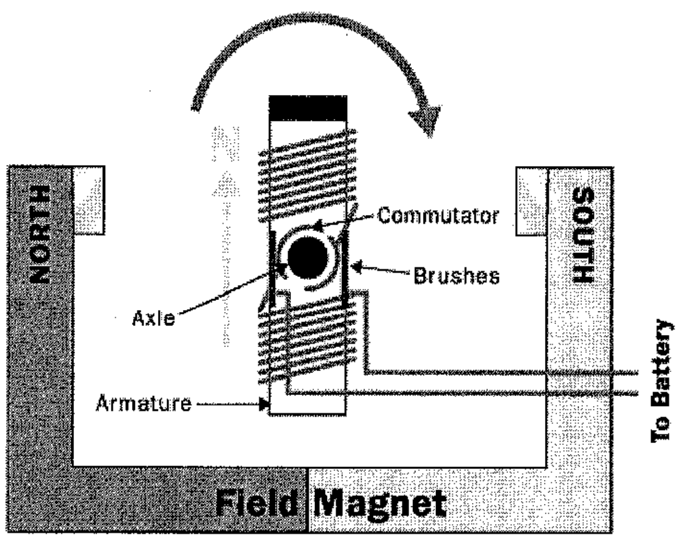


Figure 2-8: Parts of an Electric Motor.

Figure 2-8 is the overall plan of a simple two-pole DC electric motor. A simple motor has six parts as shown in the figure:

- Armature or rotor
- Commutator
- Brushes
- Axle
- Field magnet
- DC power supply

Electric motors fundamentally operate on the principle of magnetism that opposite poles of a magnet attract, and like poles repel. To achieve motion, a motor must do three things that are generating magnetic fields on the moving part of the motor (rotor or armature), generating a magnetic field in the stationary part (stator), and providing some means to keep one of the fields moving. As one field "chases" the other attempting to align, motion results [8].

2.3.4 Light Dependent Resistor (LDR) Sensor

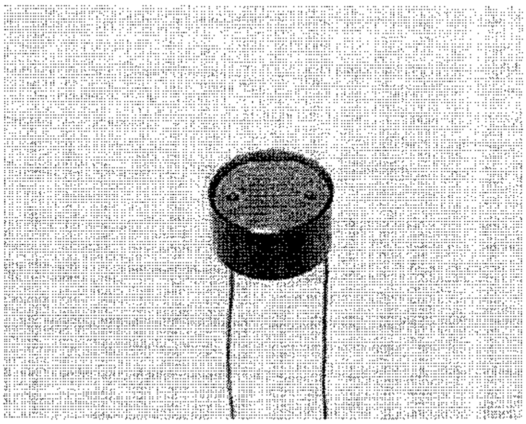


Figure 2-9: Example of LDR sensor.

A LDR will have a resistance that varies according to the amount of visible light that falls on it. The light falling on the brown zigzag lines on the sensor causes the resistance of the device to fall. This is known as a negative co-efficient. There are some LDRs that work in the opposite way i.e. their resistance increases with light (called positive co-efficient) [9].

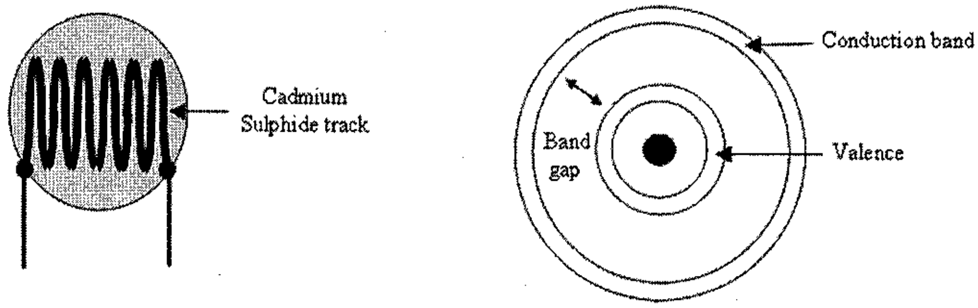


Figure 2-10: Structure of a Light Dependent Resistor, Showing Cadmium Sulphide Track and an Atom to Illustrate Electrons in the Valence and Conduction Bands.

2.3.4.1 Principle Operation

LDR is made from cadmium sulphide. Cadmium Sulphide is a II-VI semiconductor. (It is so called because Cadmium is in group II and Sulphide is in group VI.) It is light sensitive. When the light shining on it is stronger, the resistance of the LDR is smaller.

It is important to note that light has dual properties. On the one hand, light is electromagnetic wave, on the other hand, it can be seen as photons (energetic particles). When light shines on the LDR, the photons break the bonds in the cadmium sulphide and release electrons for the conduction. If the light is of a stronger intensity, then more bonds are broken, thus more electrons are freed for the conduction. So LDR's resistance decreases when stronger light shines on it.

CHAPTER 3

METHODOLOGY/PROJECT WORK

3.1 PROCEDURE IDENTIFICATION

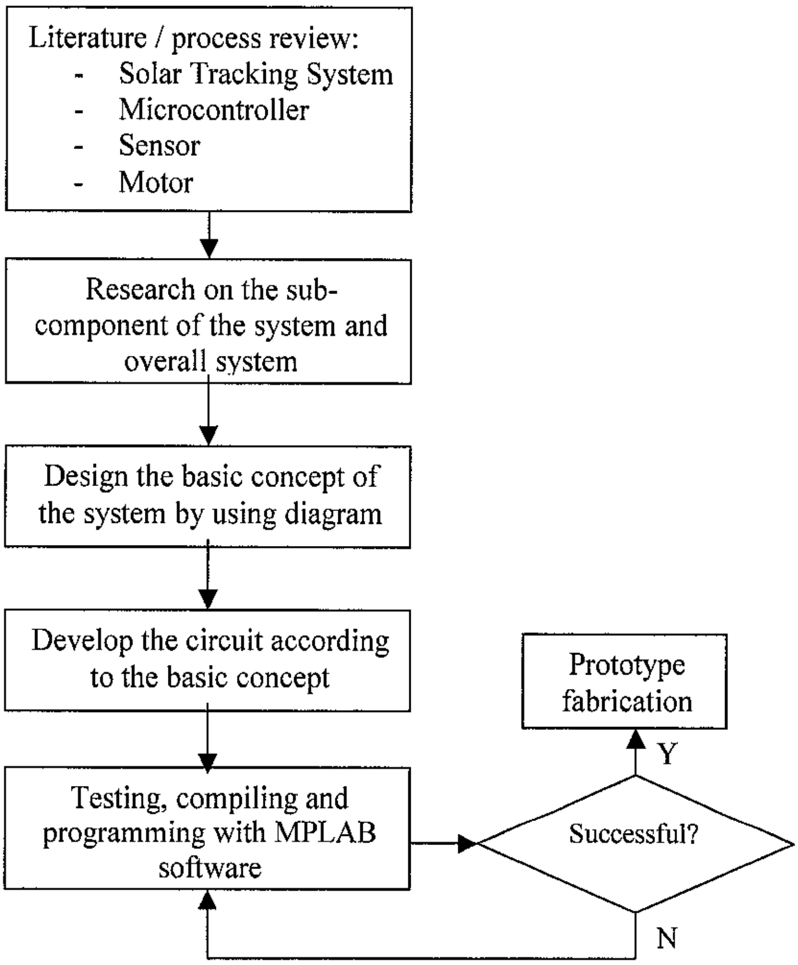


Figure 3-1: Flow Chart for Basic Procedure Identification.

In the initial stages of the project a thorough review of the literature was conducted. It is the initial step to show the student understanding towards the project base on the research and study that have been done. Then, the basic concept of the system will be presented including the block diagram. Starting from this point, a further research will be done and the required circuit design will be developed. After that, the source code of the microcontroller programming will be written and downloaded it to the microcontroller. The crucial part is the circuit testing, simulation and analysis of the system. After completing the circuit analysis, the prototype fabrication will be started.

3.2 HARDWARE

3.2.1 Sensor's Circuit

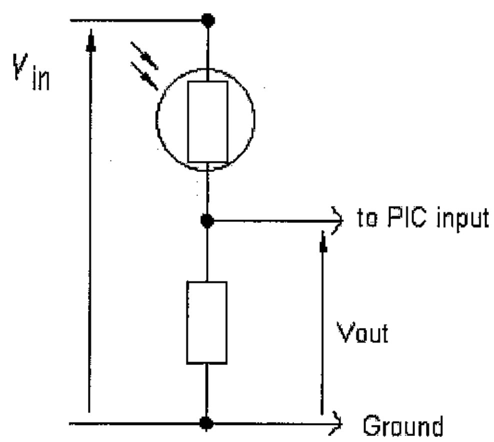


Figure 3-2: The Circuit for the LDR Sensor

The diagram in Figure 3-2 shows the circuit of the LDR sensor. There would be two circuits for the LDR sensor so that it will be two outputs to be connected to the PIC. Each of the LDR sensors is mounted in a "well" (straw) with a narrow slit so that sunlight falls upon it only when the LDR sensor pointed directly to the sun. As discussed earlier, the output voltage from both sensors' circuit will be compared in microcontroller after converting it to digital form. By putting the two sensors in the straw, the output voltage from each of it will be recognized better and then improve the efficiency of the system.

The LDR circuit is supplied with 5 V DC power supply. The output voltage from the bottom resistor will be connected to the microcontroller which is the PIC16F877. The value of the bottom resistor needs to be determined first. Below is the formula used for the calculation.

$$V_{out} = \frac{R_{bottom}}{R_{bottom} + R_{top}} \times V_{in} \quad [10]$$

3.2.2 Motor Controller's Circuit

Base on the H-bridge circuit described in Literature Review section, a circuit is constructed by using two relays (5V 8 pins). The theory is, if 5V input given to one of the relays, the motor will rotate clockwise and vice versa if 5V is given to the other relay. The relays circuit is shown in Figure 3-3.

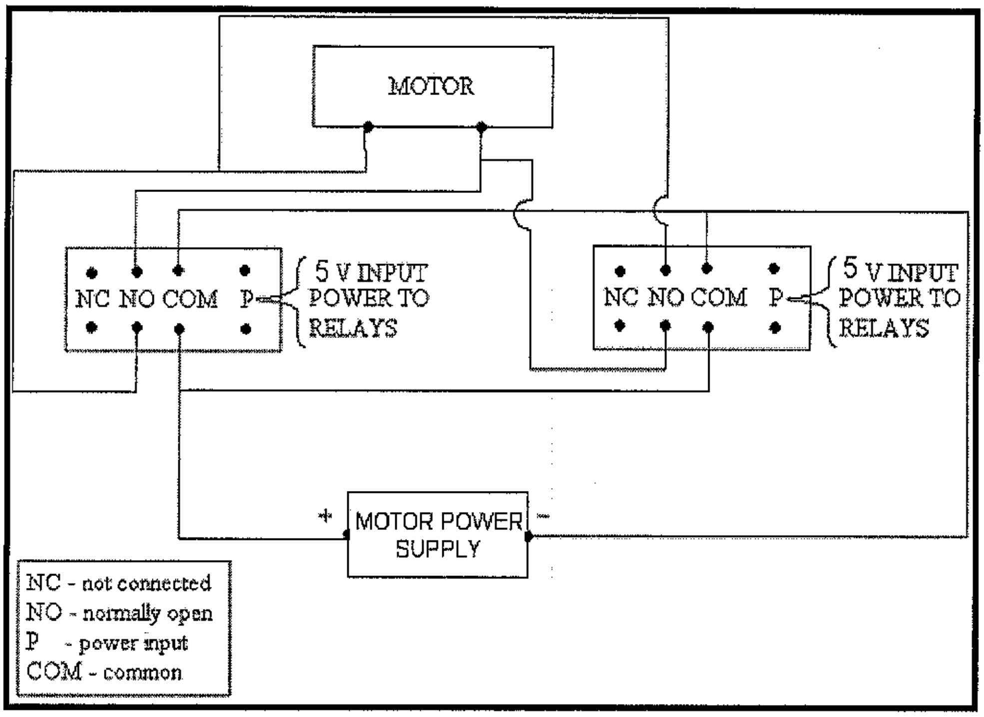


Figure 3-3: The Relays Circuit.

After designing the circuit, the student constructed the actual circuit on the veraboard as shown in Figure 3-4.

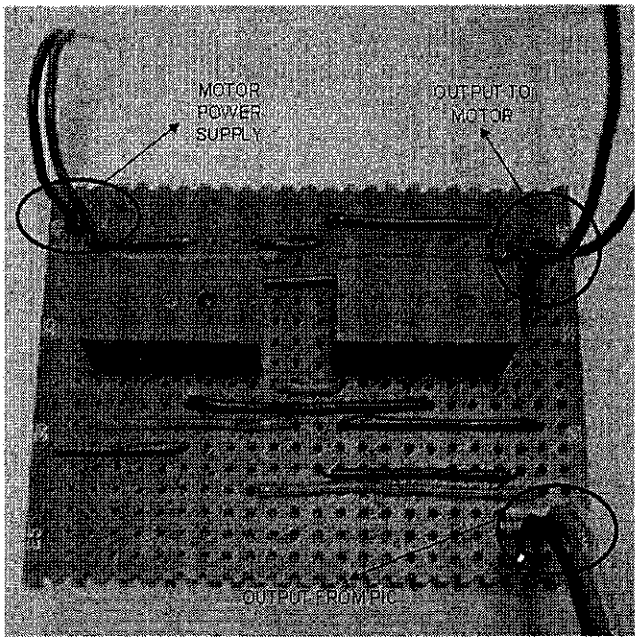


Figure 3-4: The Motor Controller Circuit Mounted on Veraboard.

3.3 SOFTWARE

Microchip MPLAB IDE v6.42 is used in this project for programming part. Microchip MPLAB is the Windows Integrated Development Environment for development systems tools. The program will be written by using this software and then will be downloaded to the PIC.

3.3.1 Block Diagram of Program's Flow

The block diagram of program's flow shows what is happening in the PIC microcontroller. Upon receiving the output from two LDR sensor circuits, the PIC will convert the analog value to digital binary representation. Each of the value will be stored in different variable. Then, the subtraction operation is done on the two binaries. The PIC will check the status registers, Z and C after the operation. Base on the value of the status registers, the PIC will produce an output to motor controller. Motor controller will drive the motor whether to rotate clockwise, anticlockwise or in hold position.

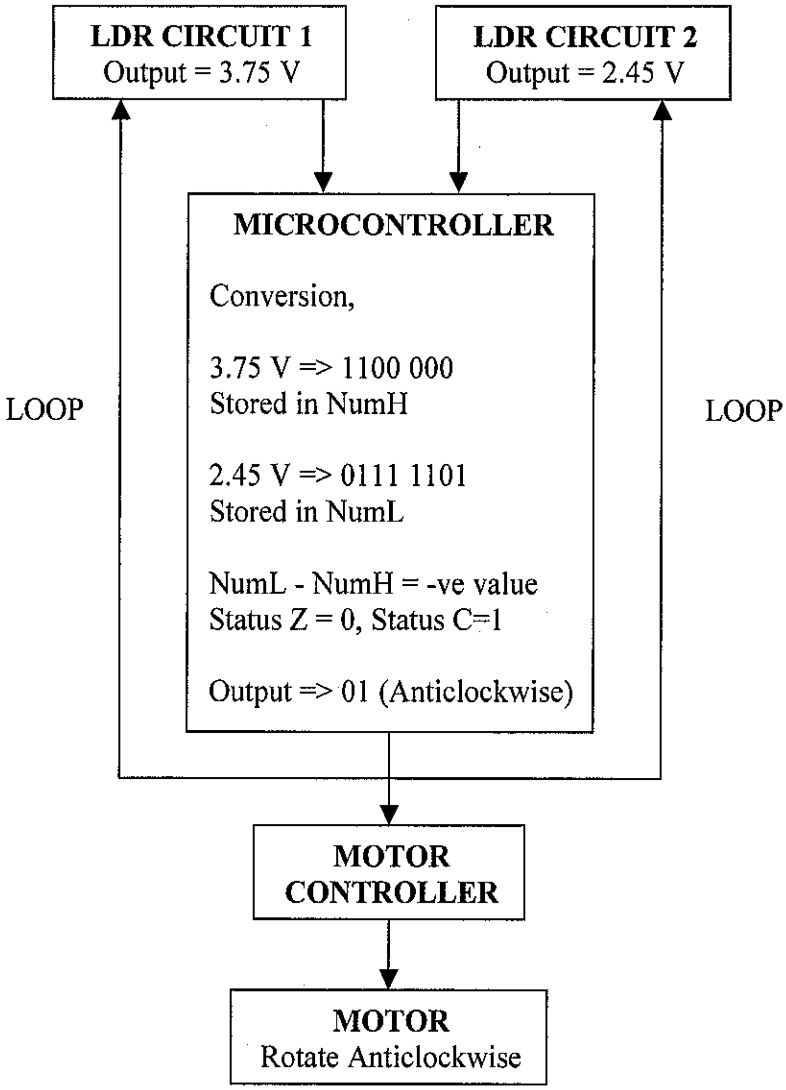


Figure 3-5: The Block Diagram of the PIC's Operation.

3.3.2 Source Code

Below is the complete source code written for the purpose of controlling the solar tracking system.

```
            #include <p16f877.inc>
            cblock          0x20            ; start of general purpose registers
                            NumL            ; register for Input 1
                            NumH            ; register for Input 2
            endc

;start of program
            ORG             0x0000
            GOTO            Initialise
            ORG             0x0004
Initialise  clrf            PORTA
            clrf            PORTC
            BANKSEL         ADCON1          ; disable A2D
            movlw           0x06
            movwf           ADCON1
            BANKSEL         PORTA
```

```
SetPorts        bcf         STATUS, RP0         ; select bank 0
                call        Init_ADC0           ; initialise analogue input
                bsf         STATUS,RP0          ; bank 1
                bcf         STATUS,RP1
                movlw       H'00'
                movwf       TRISC
                bcf         STATUS, RP0

Main
                call        Read_ADC            ; read analogue input
                call        Init_ADC1           ; set for second channel

;read and display second input
                call        Read_ADC2           ; read analogue input
                call        Init_ADC0           ; set for first channel

cclockwise
                movf        NumH, W
                subwf       NumL                ; subtract operation
                btfss       STATUS, C           ; skip next instruction if C = 1*
                goto        clockwise
                btfsc       STATUS, Z           ; skip next instruction if Z = 0*
                goto        hold
                movlw       b'00000001'         ; send signal to motor controller
                movwf       PORTC
                goto        Main

clockwise
                movlw       b'00000010'         ; send signal to motor controller
                movwf       PORTC
                goto        Main

hold
                movlw       b'00000000'         ; send signal to motor controller
                movwf       PORTC
                goto        Main

Init_ADC0                                       ; set for AN0
; Set ADCON0
                movlw       b'01000001'
                movwf       ADCON0
; Set ADCON1
                BANKSEL     ADCON1
                movlw       b'00000000'
                movwf       ADCON1
                BANKSEL     ADCON0
                return


Init_ADC1                                       ; set for AN1
; Set ADCON0
                movlw       b'01001001'
                movwf       ADCON0
; Set ADCON1
                BANKSEL     ADCON1
                movlw       b'00000000'
                movwf       ADCON1
                BANKSEL     ADCON0
                return
Read_ADC
                bsf         ADCON0, GO_DONE             ; initiate conversion
                btfsc       ADCON0, GO_DONE
                goto        $-1                         ; wait for ADC to finish

                movf        ADRESH,W
                movwf       NumH
                return
Read_ADC2
                bsf         ADCON0, GO_DONE             ; initiate conversion
                btfsc       ADCON0, GO_DONE
                goto        $-1                         ; wait for ADC to finish

                movf        ADRESH,W
                movwf       NumL
                return
                end
```

The coding is based on below algorithm,

* There are only 3 possible results:

1) if NumH < NumL: Status,Z = 0. Status,C = 1.
2) if NumH == NumL: Status,Z = 1. Status,C = 1.
3) if NumL < NumH: Status,Z = 0. Status,C = 0.

Z and C are the status register in the PIC16F877. These two registers will be affected after some operations are performed such as ADD and SUBTRACTION operations. In the source code, the value in register NumH is subtracted from the value in register NumL. If the result is negative, the status of the register C will turn to '1' whereas the status register Z will be '0'. If the result is zero, both of the status registers will turn to '1'. And if the result is positive, the value for both of the status registers will be '0'. Base on these status registers value, the program will be directed to clockwise, anticlockwise or hold section in order to produce an output to motor controller circuit.

Actually, the result of the analogue to digital conversion is stored in ADRESH:ADRESL register in 10-bit format. But to make it simple and efficient, the results are stored in NumH (AN0) and NumL (AN1) registers in 8-bit form.

3.4 PROJECT DEVELOPMENT

The development of the Solar Tracking System includes the construction of main controller circuit, LDR sensor circuit, and also the motor controller circuit. Then, all the circuits will be integrated together to fabricate the prototype.

3.4.1 Main Controller Circuit

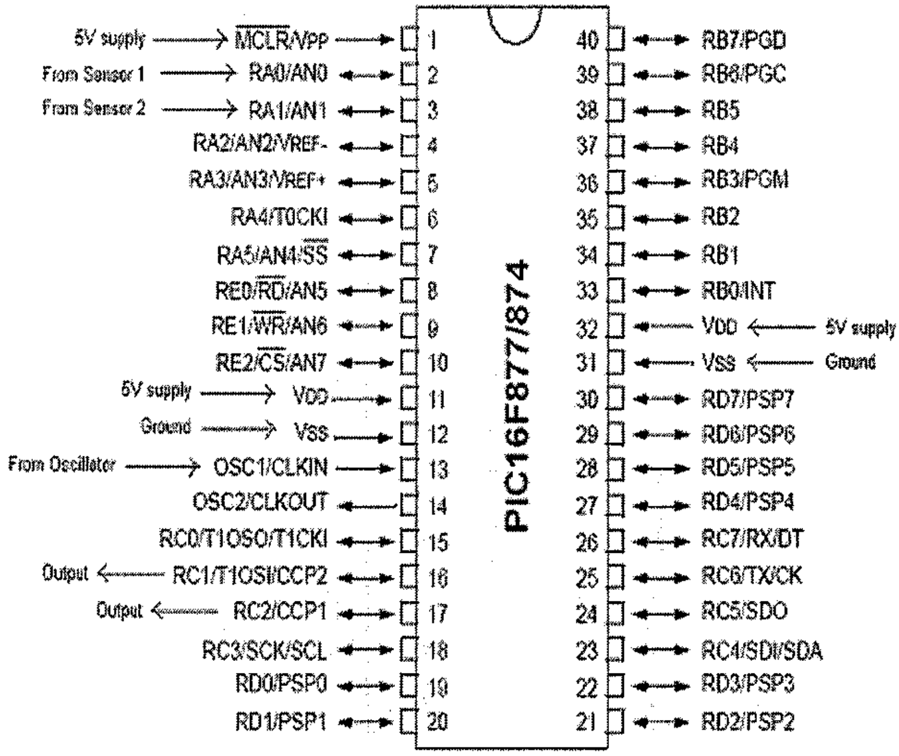


Figure 3-6: The Circuit Diagram for the PIC.

Crystal Oscillator (Clock) will supply the circuit with 4 MHz clock. Figure 3-7 shows the pin connection of the clock.

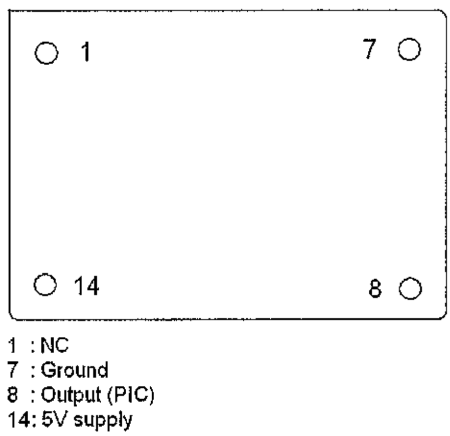


Figure 3-7: The Pin Connection for Oscillator.

After designing and constructing the circuit on breadboard, the student transfers the circuit to veraboard. It is shown in Figure 3-8.

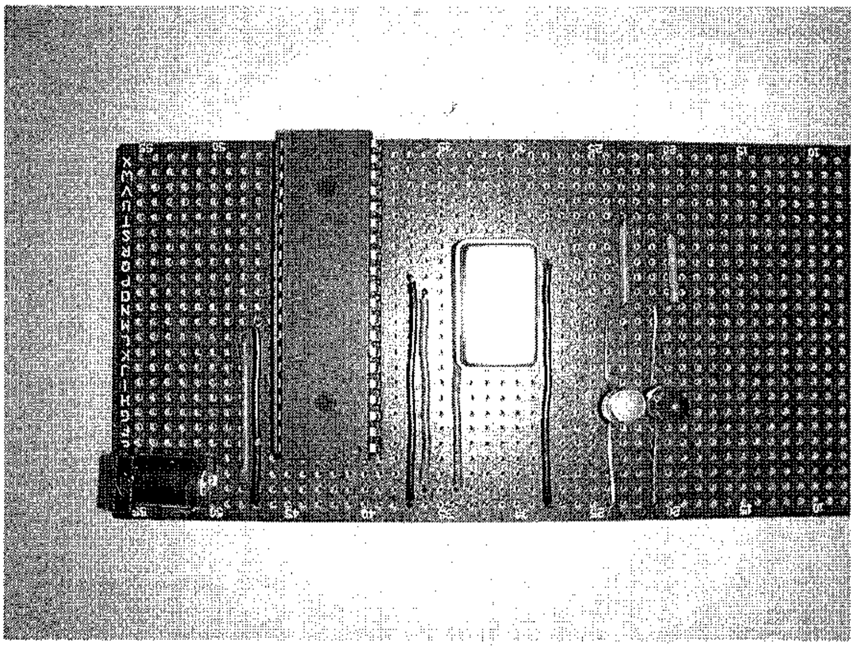


Figure 3-8: The Main Controller Circuit on the Veraboard.

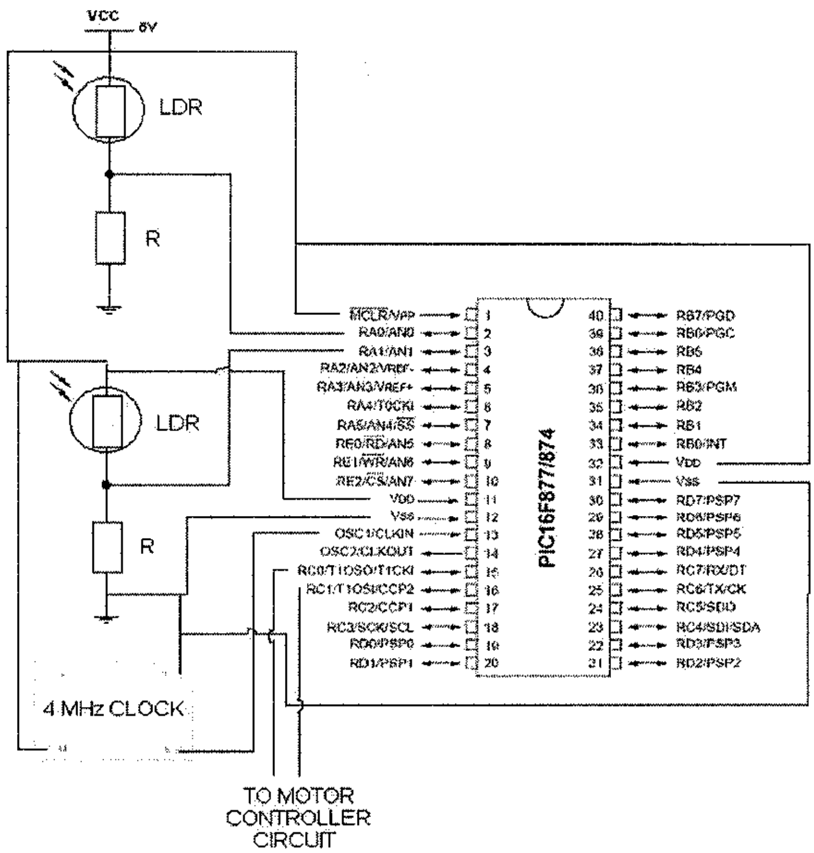


Figure 3-9: The Circuit Diagram for the Microcontroller and LDR Sensor Circuit.

3.4.2 LDR Sensor Circuit

There will be two LDR sensors attached to the system. Each of the LDR sensor circuits is attached to the edge of the solar panel base as shown in Figure 3-10. As stated earlier, the LDR sensors are put in a straw so that there will be a difference in amount of light received by each sensor due to the orientation of the panel.

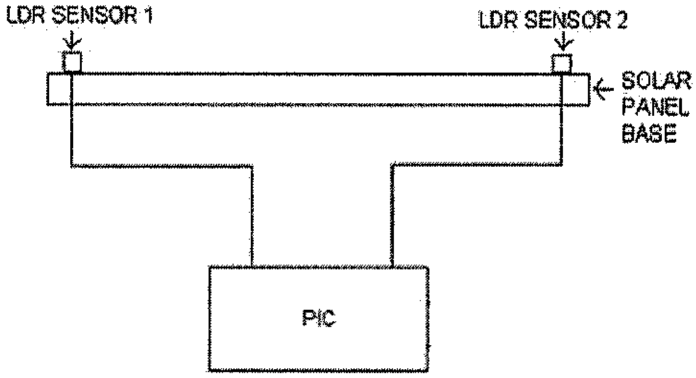


Figure 3-10: The Location of the LDR Sensor Circuits.

Figure 3-11 shows the complete circuit of LDR sensor. This circuit will be integrated with the main controller circuit.

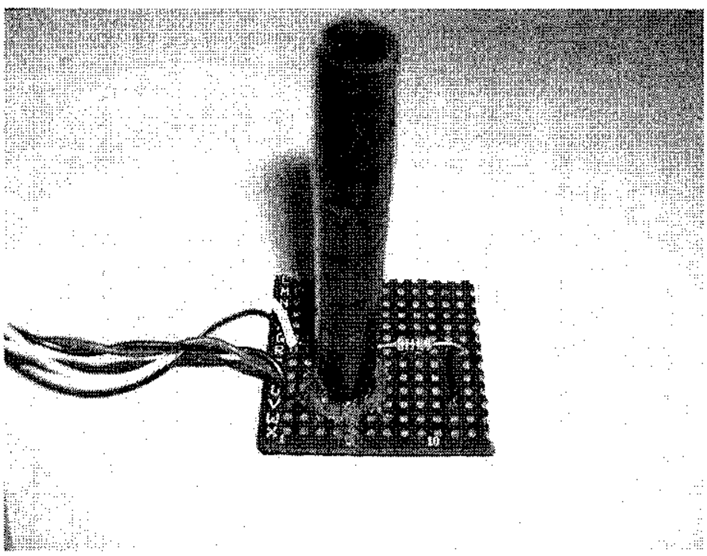


Figure 3-11: The Complete LDR Sensor Circuit.

3.4.2 Software (ADCON0 and ADCON1) Configuration

The programming part requires a further understanding on the registers used in the PIC especially for A/D converter. The configuration of ADCON0 and ADCON1 registers will be explained in detail below.

There are four main registers associated with using the analogue inputs, these are listed in this table:

Table 3-1: Main Registers for Analogue Inputs.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADRESH	A2D Result Register - High Byte							
ADRESL	A2D Result Register - Low Byte							
ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	-	ADON
ADCON1	ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0

ADRESH and ADRESL are fairly self explanatory, they are the registers that return the result of the analogue to digital conversion, the only slightly tricky thing about them is that they are in different memory banks.

3.4.2.1 ADCON0 Details

ADCON0 is split into four separate parts, the first part consists of the highest two bits ADCS1 and ADCS0, and sets the clock frequency used for the analogue to digital conversion, this is divided down from the system clock (or can use an internal RC oscillator), as the project is using a 4MHz clock, Fosc/8 will be used (as given in the table below).

Table 3-2: Clock Select Bits for A/D Conversion.

ADCS1	ADCS0	A/D Conversion Clock Select bits	Max. Clock Freq.
0	0	Fosc/2	1.25MHz
0	1	Fosc/8	5MHz
1	0	FOsc/32	20MHz
1	1	Frc (Internal A2D RC Osc.)	Typ. 4uS

The second part of ADCON0 consists of the next three bits, CHS2, CHS1 and CHS0, these are the channel select bits, and set which input pin is routed to the analogue to digital converter. Eights inputs are available (AN0-AN7) for A/D conversion on the 16F877.

Table 3-3: Bits Select to Determine the A/D Conversion Input.

CHS2	CHS1	CHS0	Channel	Pin
0	0	0	Channel0	RA0/AN0
0	0	1	Channel1	RA1/AN1
0	1	0	Channel2	RA2/AN2
0	1	1	Channel3	RA3/AN3
1	0	0	Channel4	RA5/AN4
1	0	1	Channel5	RE0/AN5
1	1	0	Channel6	RE1/AN6
1	1	1	Channel7	RE2/AN7

The third part is a single bit (bit 2), GO/DONE, this bit has two functions, firstly by setting the bit it initiates the start of analogue to digital conversion, secondly the bit is cleared when the conversion is complete - this bit will be checked to wait for the conversion to finish.

The fourth part is another single bit (bit 0), ADON, this simply turns the A2D On or Off, with the bit set it's On, with the bit cleared it's Off - thus saving the power it consumes.

So for our application the data required in ADCON0 is binary '01000001' to read from AN0, and binary '01001001' to read from AN1. Bit 1 isn't used, and can be either '0' or '1' - in this project; it will be set to '0'. To initiate a conversion, Bit 2 is set to high by a "BCF ADCON0, GO_DONE" line. Likewise, a "BTFSC ADCON0, GO_DONE" line will be used to check for the end of conversion.

Table 3-4: Bits Required in ADCON0.

Input	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AN0	0	1	0	0	0	0	-	1
AN1	0	1	0	0	1	0	-	1

3.4.2.2 ADCON1 Details

ADCON1 is really a little more complicated, although it's only split into two sections. The first section is a single bit, ADFM; this is the Result Format Selection Bit, and selects if the output is Right Justified (bit set) or Left Justified (bit cleared). The advantage of this is that it makes it very easy to use as an 8 bit converter (instead of ten bit) - by clearing this bit, and reading just ADRESH, we get an 8 bit result, ignoring the two least significant bits in ADRESL. For this project, the ADFM will be set as 0.

PCFG3-0 are probably the most complicated part of setting the A2D section, they set a lot of different options, and also limit which pins can be analogue, and which can be digital:

Table 3-5: Bit Selects to Determine Type of Input at Each Pin.

PCFG3-PCFG0	AN7 RE2	AN6 RE1	AN5 RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	Vref+	Vref-
0000	A	A	A	A	A	A	A	A	Vdd	Vss
0001	A	A	A	A	Vref+	A	A	A	RA3	Vss
0010	D	D	D	A	A	A	A	A	Vdd	Vss
0011	D	D	D	A	Vref+	A	A	A	RA3	Vss
0100	D	D	D	D	A	D	A	A	Vdd	Vss
0101	D	D	D	D	Vref+	D	A	A	RA3	Vss
0110	D	D	D	D	D	D	D	D	Vdd	Vss
0111	D	D	D	D	D	D	D	D	Vdd	Vss
1000	A	A	A	A	Vref+	Vref-	A	A	RA3	RA2
1001	D	D	A	A	A	A	A	A	Vdd	Vss
1010	D	D	A	A	Vref+	A	A	A	RA3	Vss
1011	D	D	A	A	Vref+	Vref-	A	A	RA3	RA2
1100	D	D	D	A	Vref+	Vref-	A	A	RA3	RA2
1101	D	D	D	D	Vref+	Vref-	A	A	RA3	RA2
1110	D	D	D	D	D	D	D	A	Vdd	Vss
1111	D	D	D	D	Vref+	Vref-	D	A	RA3	RA2

As mentioned above, this part looks rather complicated - but when it is split down, it starts to make more sense. There are actually four different options being set here:

1. Setting a pin to be an analogue input.
2. Setting a pin to be a digital input.

3. Setting the positive reference for the converter (Vref+).
4. Setting the negative reference for the converter (Vref-).

For a start, the initial setting for ADCON1 register should be decided. First the project is only using analogue inputs AN0 and AN1. As an assumption, all the inputs are set to be analogue which from the table eliminates 13 of the possibilities (0010, 0011, 0100, 0101, 0110, 0111, 1001, 1010, 1011, 1100, 1101, 1110 and 1111). Secondly we are using a VRef- of Vss (Ground) and Vref+ of Vdd (power supply), so that eliminates another two (0001 and 1000) alternatives which leaves the only option which fits all our requirements is '0000', so this is the value we will need to write to PCFG3:PCFG0.

So now the ADCON1 will be set to binary '00000000', with 0's in the places of the unused bits.

Table 3-6: Bits Required in ADCON1.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCON1	0	-	-	-	0	0	0	0

So, the bits select for the ADCON0 and ADCON1 is already determined. The next step is to check the current setting in the source code and its integrity with the hardware or circuit.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 LDR CIRCUIT

A test has been done on the LDR sensor. The minimum resistance for that sensor is 120 ohm (during sunshine) and the maximum resistance is 230k ohm (at night). From this value, the value of the bottom resistor will be determined.

$$V_{out} = \frac{R_{bottom}}{R_{bottom} + R_{top}} \times V_{in}$$

To determine the value of the bottom resistor, the student assumes that the maximum output voltage will be 4 V. The input voltage will be set to 5 V.

4 V	= [R / (R + 120 ohm)] X 5
R / (R + 120 ohm)	= 4/5
R	= (4R/5) + 96 ohm
R	= **480 ohm**

For circuit simplicity, 550 ohm resistor was decided to be used instead of 480 ohm. The maximum output voltage will be,

Vout	= [550 ohm / (550 ohm + 120 ohm)] X 5
	= **4.1045 V**

And the minimum output voltage will be,

Vout = [550 ohm / (550 ohm + 230k ohm)] X 5

= **0.0120 V**

4.2 ANALOGUE TO DIGITAL CONVERTER

4.2.1 One Analogue Source

An experiment has been conducted on the source code for Analogue to Digital Converter. The result is shown in Table 4-1. The experiment is done by connecting power supply to pin RA0. The power source will be varied from 0 V to 4.5 V. The output will be a series of LED connected to PORTC to represent the binary value.

Table 4-1: The Result of the Experiment for One Analogue Input.

Input Voltage, V	**Output, (Binary representation)**	**Output (Calculation), (Binary representation)**
0	0000 0000	0000 0000
0.2	0000 0111	0000 1010
1.0	0011 0011	0011 0011
1.5	0100 1111	0100 1100
2.0	0110 0001	0110 0110
2.5	0111 1111	1000 0000
3.0	1001 1111	1001 1001
3.5	1011 1111	1011 0011
4.0	1100 1111	1100 1100
4.5	1110 0011	1110 0110

4.2.2 Two Analogue Sources

The second experiment is done by using another source code, which is the modified version to receive two analogue inputs. The hardware connection is just like the previous experiment except RA1 is used as the second analogue input. Below is the result.

Table 4-2: The Result of the Experiment for Two Analogue Inputs.

Input Voltage, V	Output for RA0, (Binary)	Output for RA1, (Binary)	Output (Calculation), (Binary)
0	0000 0000	0000 0000	0000 0000
0.5	0001 1001	0001 0100	0011 0001
1.0	0011 0000	0011 0000	0011 0011
1.5	0100 1100	0100 1000	0100 1100
2.0	0110 0100	0110 0100	0110 0110
2.5	1000 0000	1000 0000	1000 0000
3.0	1001 1000	1001 1000	1001 1001
3.5	1011 0110	1011 0100	1011 0011
4.0	1101 0001	1101 0000	1100 1100
4.5	1110 1100	1110 1100	1110 0110

From the above result, it is confirmed that the source code is efficient. Even though there is slight different between the experimental result and calculation value, it is due to input voltage precision. For the two inputs conversion, the output is not really stable. But as long as the result is accurate enough, the source code will be used for the next debugging process step. The next step is to modify the source code to compare the two binary values and produce the required output for motor controller circuit.

4.3 COMPLETE SOURCE CODE

From the previous experimented source codes, the student did a modification by adding a subtraction operation to compare those two digital values and produce an output to motor controller circuit. Below is the table showing the output of the LED and what is represented by the combination.

Table 4-3: LEDs Representation.

NO	LEDs	Represent
1	00	Motor in hold position
2	01	Motor rotates counter clockwise
3	10	Motor rotates clockwise

*0 – LED off, 1 – LED on

A couple of simulations have been done by the student to confirm the efficiency of the program. The simulations are conducted by keeping one input to zero value and increasing the other input up to 5 V. The tables show the result of the simulations.

Table 4-4: The Result of the First Simulation.

RA0 (V)	RA1 (V)	OUTPUT (LEDs)
0	0.0	00
0	0.5	10
0	1.0	10
0	1.5	10
0	2.0	10
0	2.5	10
0	3.0	10
0	3.5	10
0	4.0	10
0	4.5	10
0	5.0	10

First simulation is conducted by keeping the input at RA0 to zero and keep increasing the input at RA1 up to 5 V. The output of LEDs are showing 10 if RA1 > RA0, which means the motor is rotating clockwise. The result is shown in Table 4-4.

Table 4-5: The Result of the Second Simulation.

RA0	RA1	OUTPUT (LEDs)
0.0	0	01
0.5	0	01
1.0	0	01
1.5	0	01
2.0	0	01
2.5	0	01
3.0	0	01
3.5	0	01
4.0	0	01
4.5	0	01
5.0	0	01

Second simulation is conducted by keeping the input at RA1 to zero and keep increasing the input at RA0 up to 5 V. The output of LEDs are showing 01 if RA0 > RA1, which means the motor is rotating anticlockwise. The result is shown in Table 4-5.

4.4 PROTOTYPE FABRICATION

After completing the design of the solar tracking system, the next step is to fabricate the prototype according to design. This step is the most crucial step and consumed much time. The main material used for the prototype fabrication is perspex as the base of the model. Steel rod is used as the shaft to support the solar panel base. Figure 4-1 shows the schematic diagram of the prototype. The motor and gear are not shown in the diagram.

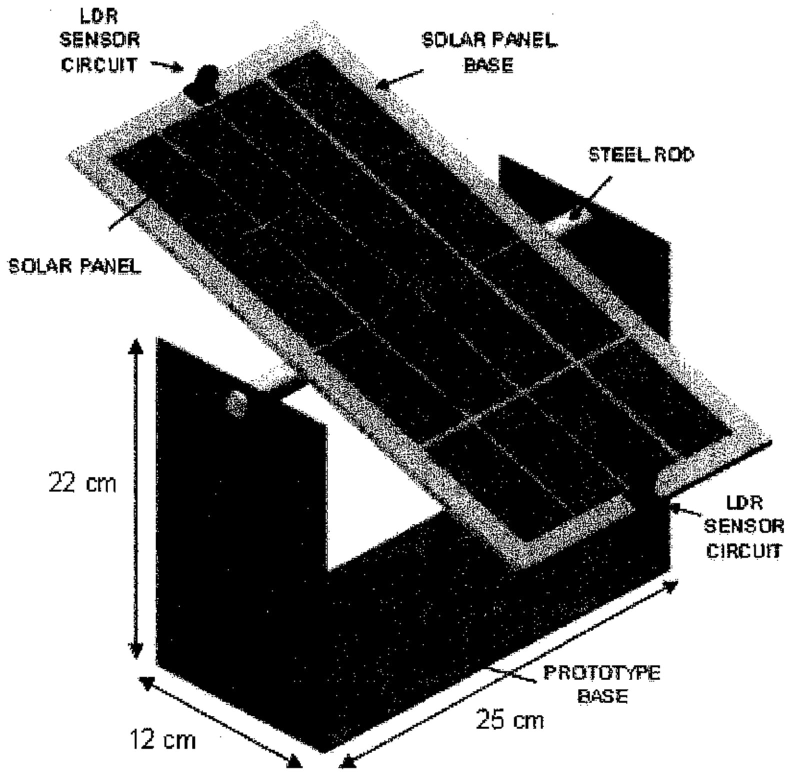


Figure 4-1: The Schematic Diagram of the Prototype.

The solar panel base is constructed independently before attaching it to the steel rod. The dimension of the solar panel base is 35 cm x 18 cm. Two solar panels are screwed on the solar panel base. The two LDR sensor circuits are attached on the both edges of solar panel base.

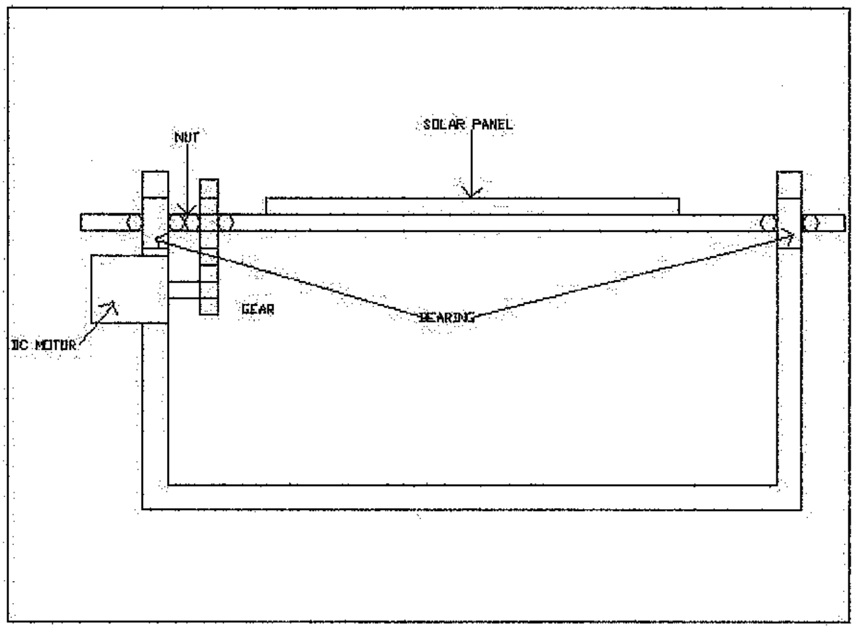


Figure 4-2: Front View of the Prototype.

The base of the prototype is mainly made from perspex. Three pieces of perspex are used to make the U-shape by glue it altogether. Before that, the perspex is drilled to make a hole for the bearings to take place. Two bearings were fabricated to the two holes to ensure the smooth movement of the model. One more hole is drilled to be attached with the motor. The 80 teeth gear is attached to the steel rod and glued. The 12 teeth gear is attached to the DC motor shaft. Then, the steel rod is attached to the bearing whereas DC motor is screwed on the base of the model.

The last step is to attach all the circuits to the prototype including the microcontroller circuit, motor controller circuit and LDR sensor circuits. After completing the fabrication of the prototype, prototype testing is conducted.

4.4 PROTOTYPE TESTING

A testing has been conducted on the prototype in other to verify and evaluate the solar tracking system. The testing is conducted manually without using the motor so that the output voltage of each LDR sensor circuit can be measured. Spotlight is used to replace the sunshine. The angle of the light is applied randomly. The rotation of motor is represented by the LED. The Table 4-6 shows the result of the testing.

Table 4-6: The Result of Prototype Testing.

OUTPUT VOLTAGE LDR Circuit 1 (V)	**OUTPUT VOLTAGE LDR Circuit 2 (V)**	**OUTPUT (LEDs)**
1.21	0.05	10
0.03	0.05	01
1.02	0.03	10
0.15	0.03	10
0.04	0.06	01
1.54	0.90	10
1.24	0.10	10
0.13	1.13	01
0.09	0.03	10
0.03	0.15	01
0.09	1.15	01

The result shows that the prototype meets the required efficiency. The output voltage varies from 0 – 2 V approximately, a little bit different from the theoretical value which is from 0 – 4 V. This is because the testing is conducted by using spotlight and not the actual sunshine. Furthermore, The LDR sensor is mounted in the straw, thus very limited light can reach it.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

There are several ways that solar tracker can be implemented. In this project, the student has implemented the 1-axis active solar tracking system. That means the solar tracker can track the position of the sun from the east till the west. The new approached has been introduced which PIC microcontroller, LDR sensors and DC gear motor are used to track the sun.

Basically, this project is concentrating on the microcontroller part which is used to control the movement of the model. In other word, the most important thing is the programming part. Since the student starts the programming part with zero knowledge, a lot of time was spent to develop the source code. So, other improvements cannot be implemented.

After completing the Final Year Project, it can be concluded that the design concepts of the project were successfully completed. The fabrication of the prototype according to design was also completed. The microcontroller which is used to control the movement of the model was also successfully programmed.

The Solar Tracking System project is a very interesting project. This project will give the student a lot of experiences in problem solving and troubleshooting skills. Most importantly, the student gets the experience on how to conduct an engineering project. Besides, the student gets the information about the solar tracking system, something that he never heard before. The student will also familiarize himself with the programming language (assembly language) which is very important nowadays since the industry moves toward automation.

5.2 RECOMMENDATION

For the project enhancement, the student would recommend the integration with the computer interface. That means user can monitor the angle of the solar panel base, the output voltage of the solar panel, and even can do some adjustment to the solar tracker angle from the computer interface. There are two most popular medium that can be used to integrate with the computer which are through serial port or parallel port.

The project also can be improved by implementing the 2-axis system. This will require the usage of two motors and also more than two LDR sensor circuits. By introducing the 2-axis system, the efficiency of the solar tracker will be greatly improved.

All the improvements require the changes to the programming part. It will be more complex and quite challenging. The knowledge in programming language is a requirement in other to enhance the solar tracking system project.

Other techniques for solar tracking system by using PLC controller and fuzzy logic can be implemented. Some of the system combines two techniques in order to get better result.

REFERENCES

[1] http://www.solarenergy.com/info_history.html

[2] http://www.oksolar.com/panels

[3] http://www.ccs.neu.edu/home/feneric/solar.html

[4] http://www.solarserver.de/solarmagazin/anlageapril2000-e.html

[5] http://www.eere.energy.gov/solar

[6] Darren Eastwood, 2002, *Appropriate Technology: Solar Tracker*, Journal.

[7] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1335&dDocName=en010241

[8] http://www.howstuffworks.com/motor.htm

[9] http://www.mstracey.btinternet.co.uk/technical/Theory/theorysensors.htm

[10] Thomas L. Floyd, 2002, *Electronic Devices-Sixth Edition*, Prentice Hall.

[11] http://picbasic.com/resources/samples.htm

[12] http://www.doctronics.co.uk/ldr_sensors.htm

[13] http://www.oshonsoft.com/picgetstarted.html

[14] http://www.solar-trackers.com/p1-2.htm

[15] http://www.carterscott.com.au/prod10.htm

[16] http://www.electro-tech-online.com/viewtopic.php?t=6111

[17] http://www.powerlight.com/products/ground_mounted.shtml

[18] http://www.winpicprog.co.uk/

[19] http://www.all-science-fair-projects.com/science_fair_projects_encyclopedia/Light-dependent_resistor

[20] http://www.mstracey.btinternet.co.uk/technical/Theory/theorysensors.htm

[21] http://www.technologystudent.com/elec1/ldr1.htm

[22] http://library.thinkquest.org/26776/ldr.htm

[23] http://www.oksolar.com/panels/

[24] http://www.ata.org.au/basics/bassolar.htm

[25] http://www.oksolar.com/technical/pv_history.html

[26] http://www.picotech.com/experiments/solar/solar.html

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

[28] http://www.physics.unlv.edu/~bill/ecg497/spring02_proj/naod.htm

[29] http://www.mcmanis.com/chuck/robotics/projects/pic-16bit.htm

[30] http://www.solarserver.de/solarmagazin/anlageapril2000-e.html

[31] http://www.kipr.org/curriculum/photodiod.html

[32] http://www.lorentzpumps.com/etatrack.htm

[33] http://forums.basicmicro.net/ShowPost.aspx?PostID=138

[34] John A. Wood, 2000, *The Solar System-Second Edition*, Prentice Hall.

[35] R.H Bube, *Photovoltaic Materials*, Imperial College Press, USA.

[36] PIC16F87x Data Sheets 28/40-Pin 8-Bit CMOS FLASH Microcontrollers, Microchips Technology Inc., 2001.

APPENDICES

APPENDIX A
Gantt Chart

Gantt Chart for Final Year Project
(for first semester)

No	Detail/Week	1	2	3	4	5	6	7	Break	8	9	10	11	12	13	14	15	16	17	18	19	Remarks
1	Selection of topic																					
	- Topic proposal																					
	- Topic awarded																					
2	Preliminary Research Work																					
	- Problem Definition																					
	- Fundametal of Solar Tracking Concept																					
3	Submission of Preliminary Report																					
4	Further Research																					
	- DC motor characteristics																					
	- LDR sensor																					
	- Microcontroller/PLC																					
5	Methodologies and Planning Documentation																					
6	Project Work																					
	- Overall System Overview																					
	- Circuit Design																					
	- Circuit Testing, Simulation and Analysis																					
7	Submission of Progress Report																					
8	Finalised all the findings																					
9	Revision on the current project status																					
10	Submission of Interim Report																					
11	Oral Presentation																					

Gantt Chart for Final Year Project
(for second semester)

No	Detail/Week	1	2	3	4	5	6	Break	7	8	9	10	11	12	13	SW	EW				19	20
1	Briefing by FYP committee																					
	- Short briefing by FYP committee																					
	- Topic assigned to FYP 1 students																					
2	Laboratory Work																					
	- Finish the programming part																					
	- Construct the H-Bridge circuit																					
	- Choose suitable motor																					
	- Construct the LDR sensors circuit																					
	- Fabricate the prototype																					
3	Submission of Progress Report																					
4	Planning Documentation																					
5	Submission of Draft Report																					
6	Submission of Final Report (soft cover)																					
7	Revision on the current project status																					
8	Oral Presentation																					
9	Submission of Final Report (hard cover)	DUE ON FRIDAY (24TH DECEMBER 2004)																				

APPENDIX B
PIC 16F877/874 Specification Sheet

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

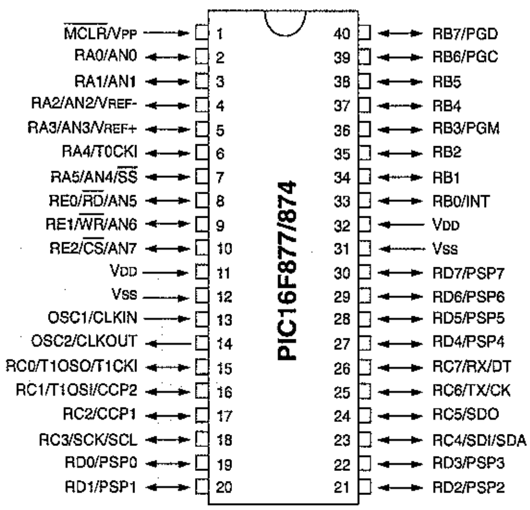
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

PDIP



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^{2}C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

1.0 DEVICE OVERVIEW

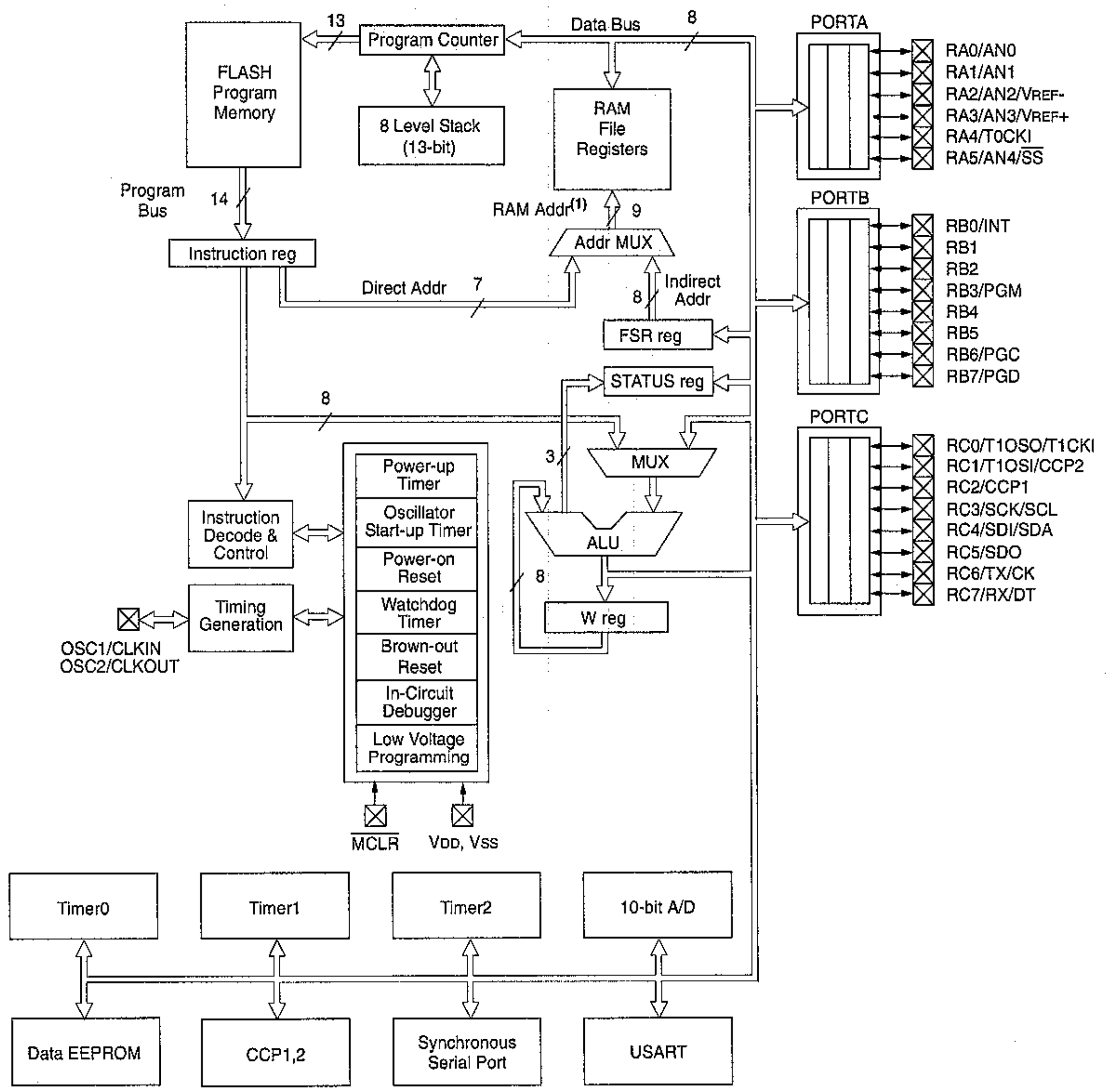
This document contains device specific information. Additional information may be found in the PICmicro™ Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

There are four devices (PIC16F873, PIC16F874, PIC16F876 and PIC16F877) covered by this data sheet. The PIC16F876/873 devices come in 28-pin packages and the PIC16F877/874 devices come in 40-pin packages. The Parallel Slave Port is not implemented on the 28-pin devices.

The following device block diagrams are sorted by pin number; 28-pin for Figure 1-1 and 40-pin for Figure 1-2. The 28-pin and 40-pin pinouts are listed in Table 1-1 and Table 1-2, respectively.

FIGURE 1-1: PIC16F873 AND PIC16F876 BLOCK DIAGRAM

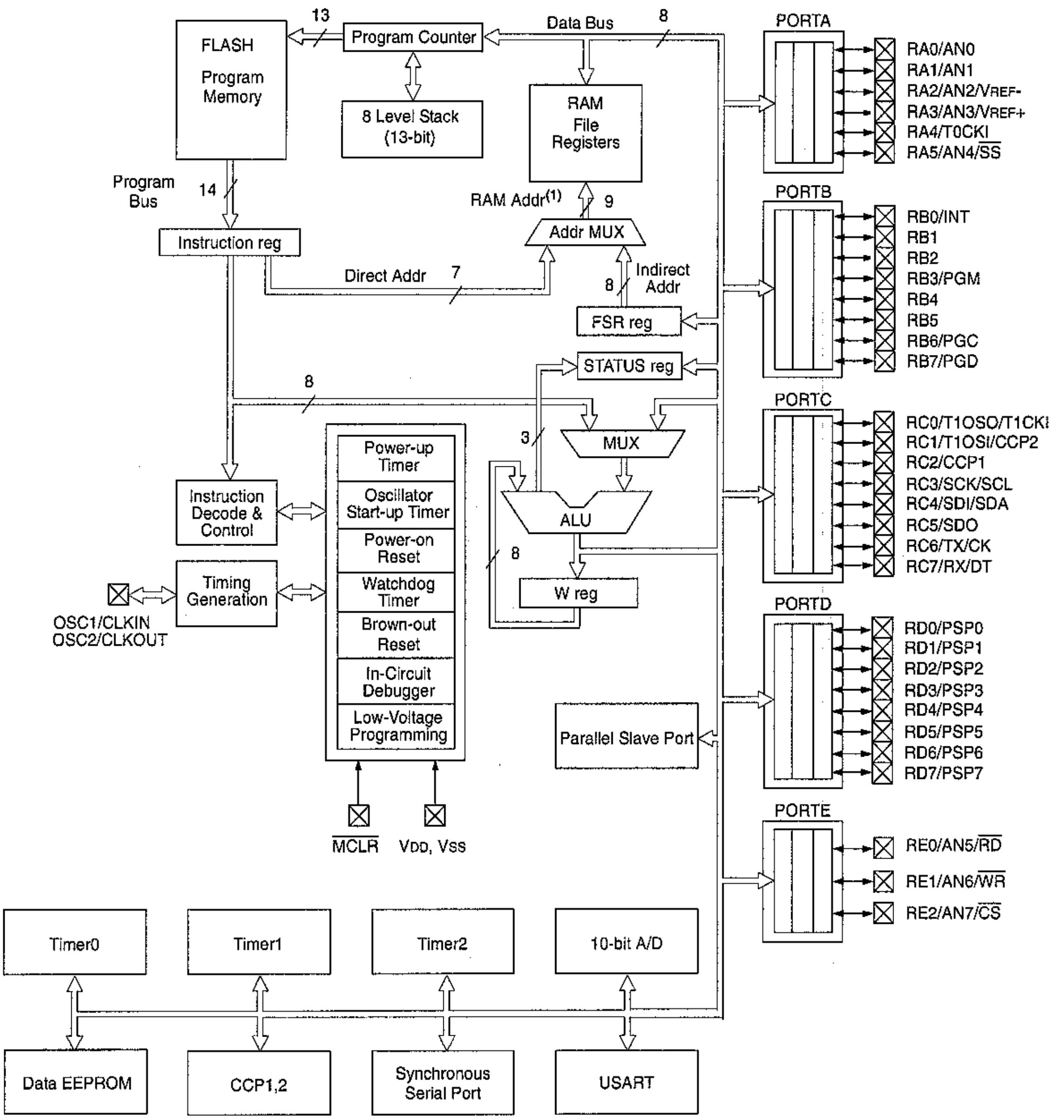
Device	Program FLASH	Data Memory	Data EEPROM
PIC16F873	4K	192 Bytes	128 Bytes
PIC16F876	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

TABLE 1-1: PIC16F873 AND PIC16F876 PINOUT DESCRIPTION

Pin Name	DIP Pin#	SOIC Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	9	9	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	10	10	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{\text{MCLR}}$/VPP	1	1	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
					PORTA is a bi-directional I/O port.
RA0/AN0	2	2	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	3	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	4	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	5	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	6	I/O	ST	RA4 can also be the clock input to the Timer0 module. Output is open drain type.
RA5/$\overline{\text{SS}}$/AN4	7	7	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	21	21	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	22	22	I/O	TTL	
RB2	23	23	I/O	TTL	
RB3/PGM	24	24	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	25	25	I/O	TTL	Interrupt-on-change pin.
RB5	26	26	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	27	27	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	28	28	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
					PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	11	11	I/O	ST	RC0 can also be the Timer1 oscillator output or Timer1 clock input.
RC1/T1OSI/CCP2	12	12	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	13	13	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	14	14	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^{2}C modes.
RC4/SDI/SDA	15	15	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^{2}C mode).
RC5/SDO	16	16	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	17	17	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	18	18	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
VSS	8, 19	8, 19	P	—	Ground reference for logic and I/O pins.
VDD	20	20	P	—	Positive supply for logic and I/O pins.

Legend: I = input   O = output   I/O = input/output   P = power
— = Not used   TTL = TTL input   ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{\text{MCLR}}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type.
RA5/$\overline{\text{SS}}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/ output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{\text{RD}}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{\text{WR}}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{\text{CS}}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
VSS	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

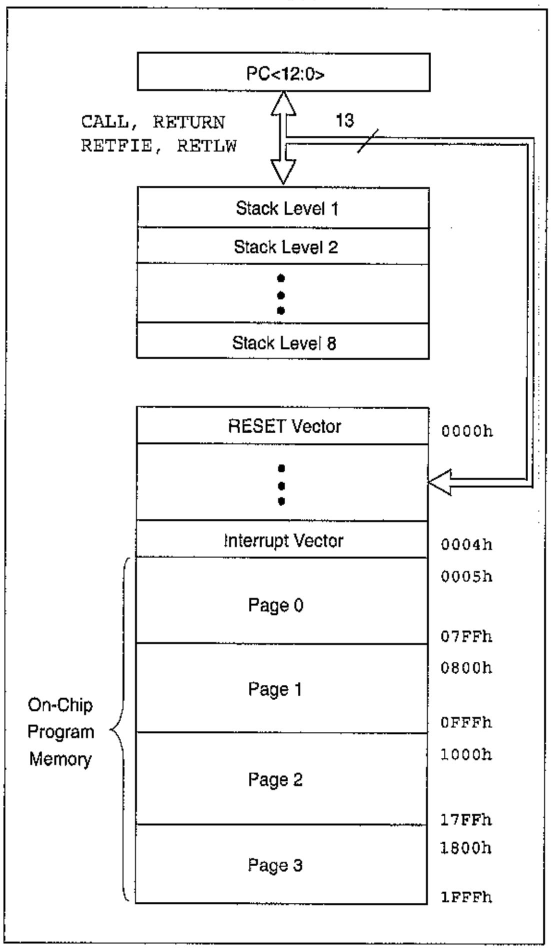
Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

2.0 MEMORY ORGANIZATION

There are three memory blocks in each of the PIC16F87X MCUs. The Program Memory and Data Memory have separate buses so that concurrent access can occur and is detailed in this section. The EEPROM data memory block is detailed in Section 4.0.

Additional information on device memory may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

FIGURE 2-1: PIC16F877/876 PROGRAM MEMORY MAP AND STACK

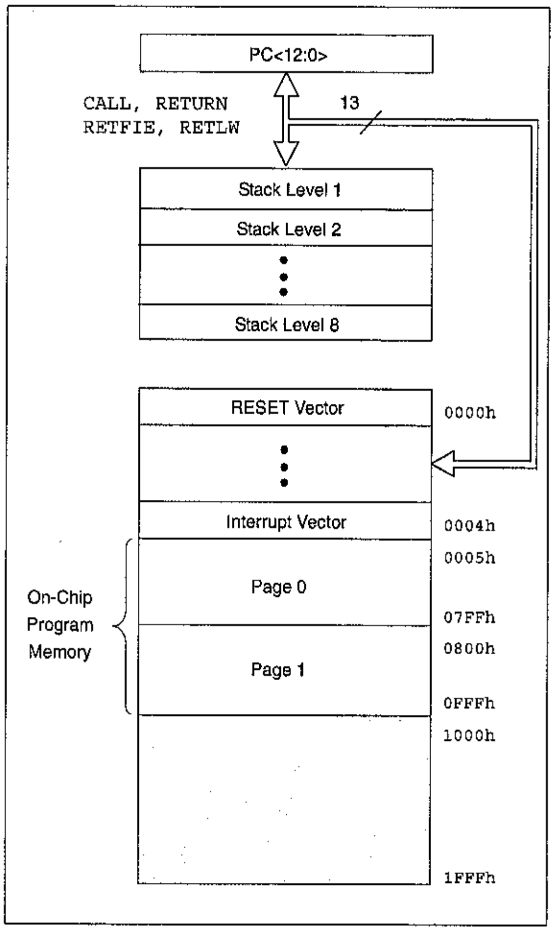


2.1 Program Memory Organization

The PIC16F87X devices have a 13-bit program counter capable of addressing an 8K x 14 program memory space. The PIC16F877/876 devices have 8K x 14 words of FLASH program memory, and the PIC16F873/874 devices have 4K x 14. Accessing a location above the physically implemented address will cause a wraparound.

The RESET vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 2-2: PIC16F874/873 PROGRAM MEMORY MAP AND STACK



2.2 Data Memory Organization

The data memory is partitioned into multiple banks which contain the General Purpose Registers and the Special Function Registers. Bits RP1 (STATUS<6>) and RP0 (STATUS<5>) are the bank select bits.

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain Special Function Registers. Some frequently used Special Function Registers from one bank may be mirrored in another bank for code reduction and quicker access.

Note:	EEPROM Data Memory description can be found in Section 4.0 of this data sheet.

2.2.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly, or indirectly through the File Select Register (FSR).

FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h	General Purpose Register 16 Bytes	110h–11Fh	General Purpose Register 16 Bytes	190h–19Fh
TMR2	11h	SSPCON2	91h				
T2CON	12h	PR2	92h				
SSPBUF	13h	SSPADD	93h				
SSPCON	14h	SSPSTAT	94h				
CCPR1L	15h		95h				
CCPR1H	16h		96h				
CCP1CON	17h		97h				
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah		9Ah				
CCPR2L	1Bh		9Bh				
CCPR2H	1Ch		9Ch				
CCP2CON	1Dh		9Dh				
ADRESH	1Eh	ADRESL	9Eh				
ADCON0	1Fh	ADCON1	9Fh				
General Purpose Register 96 Bytes	20h–7Fh	General Purpose Register 80 Bytes	A0h–EFh	General Purpose Register 80 Bytes	120h–16Fh	General Purpose Register 80 Bytes	1A0h–1EFh
		accesses 70h-7Fh	F0h–FFh	accesses 70h-7Fh	170h–17Fh	accesses 70h - 7Fh	1F0h–1FFh

Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
2: These registers are reserved, maintain these registers clear.

FIGURE 2-4: PIC16F874/873 REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h				
T2CON	12h	PR2	92h				
SSPBUF	13h	SSPADD	93h				
SSPCON	14h	SSPSTAT	94h				
CCPR1L	15h		95h				
CCPR1H	16h		96h				
CCP1CON	17h		97h				
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah		9Ah				
CCPR2L	1Bh		9Bh				
CCPR2H	1Ch		9Ch				
CCP2CON	1Dh		9Dh				
ADRESH	1Eh	ADRESL	9Eh				
ADCON0	1Fh	ADCON1	9Fh				
General Purpose Register 96 Bytes	20h	General Purpose Register 96 Bytes	A0h	accesses 20h-7Fh	120h	accesses A0h - FFh	1A0h
					16Fh		1EFh
					170h		1F0h
	7Fh		FFh		17Fh		1FFh

▒ Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F873.

2: These registers are reserved, maintain these registers clear.

APPENDIX C
Specification Sheet for ADC Module

11.0 ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has five inputs for the 28-pin devices and eight for the other devices.

The analog input charges a sample and hold capacitor. The output of the sample and hold capacitor is the input into the converter. The converter then generates a digital result of this analog level via successive approximation. The A/D conversion of the analog input signal results in a corresponding 10-bit digital number. The A/D module has high and low voltage reference input that is software selectable to some combination of VDD, Vss, RA2, or RA3.

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 clock must be derived from the A/D's internal RC oscillator.

The A/D module has four registers. These registers are:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register0 (ADCON0)
- A/D Control Register1 (ADCON1)

The ADCON0 register, shown in Register 11-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 11-2, configures the functions of the port pins. The port pins can be configured as analog inputs (RA3 can also be the voltage reference), or as digital I/O.

Additional information on using the A/D module can be found in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

REGISTER 11-1: ADCON0 REGISTER (ADDRESS: 1Fh)

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

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits
00 = Fosc/2
01 = Fosc/8
10 = Fosc/32
11 = FRC (clock derived from the internal A/D module RC oscillator)

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits
000 = channel 0, (RA0/AN0)
001 = channel 1, (RA1/AN1)
010 = channel 2, (RA2/AN2)
011 = channel 3, (RA3/AN3)
100 = channel 4, (RA5/AN4)
101 = channel 5, (RE0/AN5)(1)
110 = channel 6, (RE1/AN6)(1)
111 = channel 7, (RE2/AN7)(1)

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

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

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

Note 1: These channels are not available on PIC16F873/876 devices.

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

REGISTER 11-2: ADCON1 REGISTER (ADDRESS 9Fh)

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

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

bit 6-4 **Unimplemented:** Read as '0'

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

PCFG3: PCFG0	AN7[(1)] RE2	AN6[(1)] RE1	AN5[(1)] RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs[(2)]
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Analog input D = Digital I/O

Note 1: These channels are not available on PIC16F873/876 devices.
2: This column indicates the number of analog channels available as A/D inputs and the number of analog channels used as voltage reference inputs.

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

The ADRESH:ADRESL registers contain the 10-bit result of the A/D conversion. When the A/D conversion is complete, the result is loaded into this A/D result register pair, the GO/$\overline{\text{DONE}}$ bit (ADCON0<2>) is cleared and the A/D interrupt flag bit ADIF is set. The block diagram of the A/D module is shown in Figure 11-1.

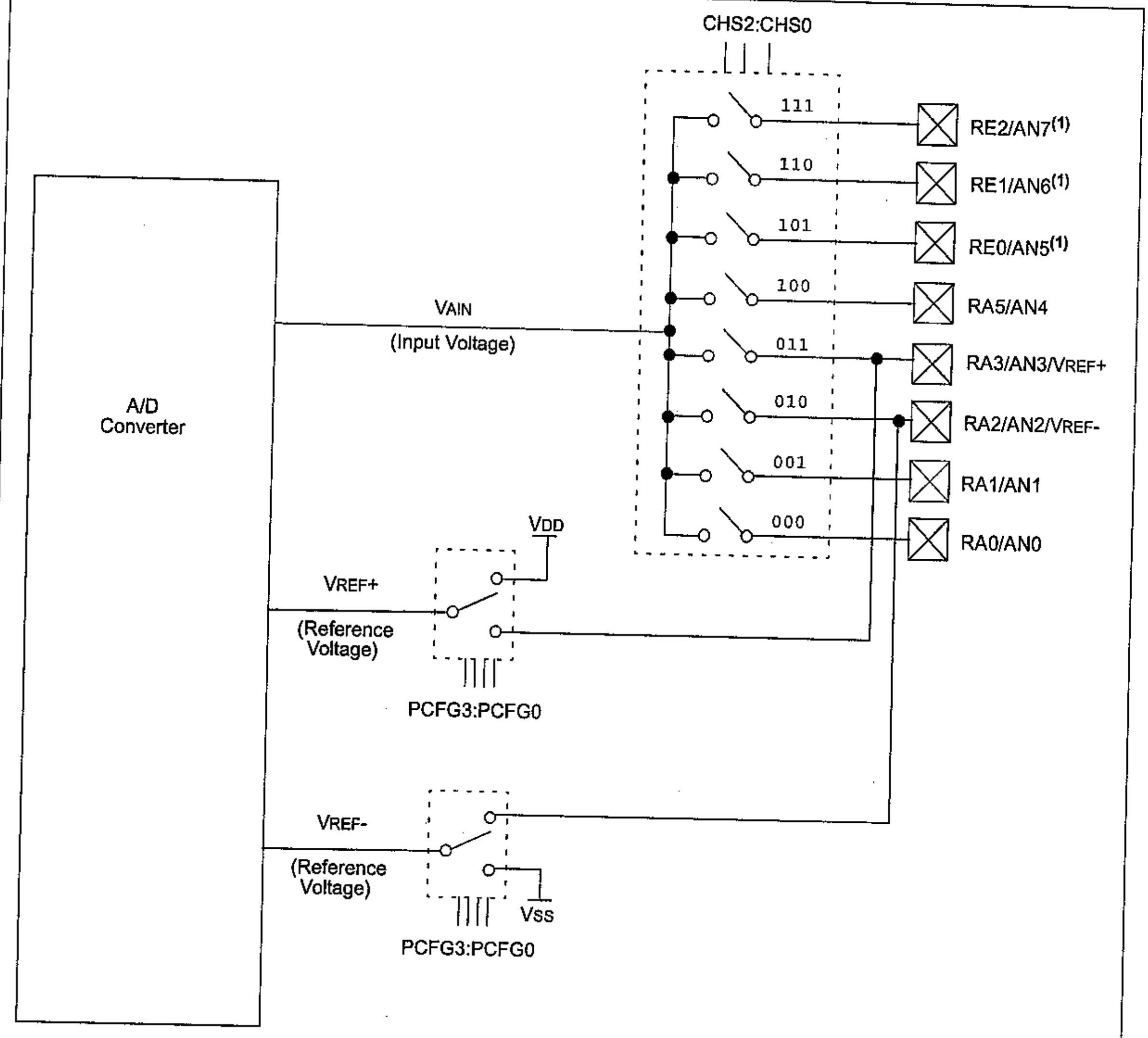
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as inputs.

To determine sample time, see Section 11.1. After this acquisition time has elapsed, the A/D conversion can be started.

These steps should be followed for doing an A/D Conversion:

1. Configure the A/D module:
   - Configure analog pins/voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D conversion clock (ADCON0)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set PEIE bit
   - Set GIE bit
3. Wait the required acquisition time.
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared (with interrupts enabled); OR
   - Waiting for the A/D interrupt
6. Read A/D result register pair (ADRESH:ADRESL), clear bit ADIF if required.
7. For the next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2TAD is required before the next acquisition starts.

FIGURE 11-1: A/D BLOCK DIAGRAM



Note 1: Not available on PIC16F873/876 devices.

11.1 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 11-2. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), see Figure 11-2. **The maximum recommended impedance for analog sources is 10 kΩ.** As the impedance is decreased, the acquisition time may be decreased.

After the analog input channel is selected (changed), this acquisition must be done before the conversion can be started.

To calculate the minimum acquisition time, Equation 11-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

To calculate the minimum acquisition time, TACQ, see the PICmicro™ Mid-Range Reference Manual (DS33023).

EQUATION 11-1: ACQUISITION TIME

$$
\begin{aligned}
T_{ACQ} &= \text{Amplifier Settling Time} + \\
&\quad \text{Hold Capacitor Charging Time} + \\
&\quad \text{Temperature Coefficient} \\
&= T_{AMP} + T_C + T_{COFF} \\
&= 2\mu s + T_C + [(\text{Temperature} - 25°C)(0.05\mu s/°C)] \\
T_C &= C_{HOLD}\,(R_{IC} + R_{SS} + R_S)\ \ln(1/2047) \\
&= -120pF\,(1k\Omega + 7k\Omega + 10k\Omega)\ \ln(0.0004885) \\
&= 16.47\mu s \\
T_{ACQ} &= 2\mu s + 16.47\mu s + [(50°C - 25°C)(0.05\mu s/°C)] \\
&= 19.72\mu s
\end{aligned}
$$

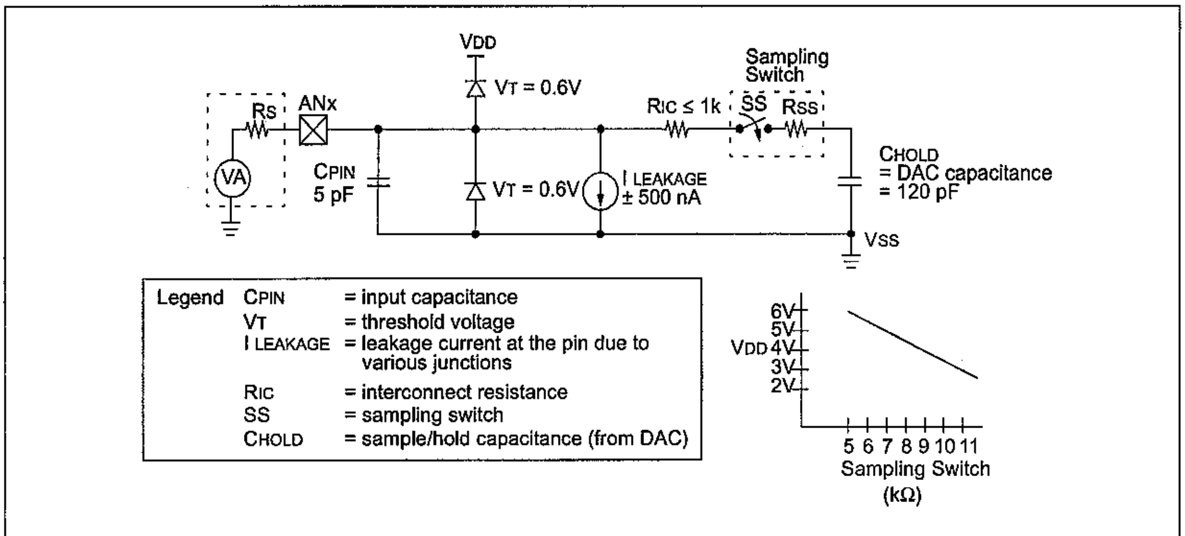
Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

4: After a conversion has completed, a 2.0TAD delay must complete before acquisition can begin again. During this time, the holding capacitor is not connected to the selected A/D input channel.

FIGURE 11-2: ANALOG INPUT MODEL



11.2 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires a minimum 12TAD per 10-bit conversion. The source of the A/D conversion clock is software selected. The four possible options for TAD are:

- 2Tosc
- 8Tosc
- 32Tosc
- Internal A/D module RC oscillator (2-6 μs)

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of 1.6 μs.

Table 11-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

TABLE 11-1: TAD vs. MAXIMUM DEVICE OPERATING FREQUENCIES (STANDARD DEVICES (C))

AD Clock Source (TAD)		Maximum Device Frequency
Operation	**ADCS1:ADCS0**	**Max.**
2Tosc	`00`	1.25 MHz
8Tosc	`01`	5 MHz
32Tosc	`10`	20 MHz
RC[1, 2, 3]	`11`	**(Note 1)**

Note 1: The RC source has a typical TAD time of 4 μs, but can vary between 2-6 μs.
2: When the device frequencies are greater than 1 MHz, the RC A/D conversion clock source is only recommended for SLEEP operation.
3: For extended voltage devices (LC), please refer to the Electrical Characteristics (Sections 15.1 and 15.2).

11.3 Configuring Analog Port Pins

The ADCON1 and TRIS registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the CHS2:CHS0 bits and the TRIS bits.

> **Note 1:** When reading the port register, any pin configured as an analog input channel will read as cleared (a low level). Pins configured as digital inputs will convert an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.
>
> **2:** Analog levels on any pin that is defined as a digital input (including the AN7:AN0 pins), may cause the input buffer to consume current that is out of the device specifications.

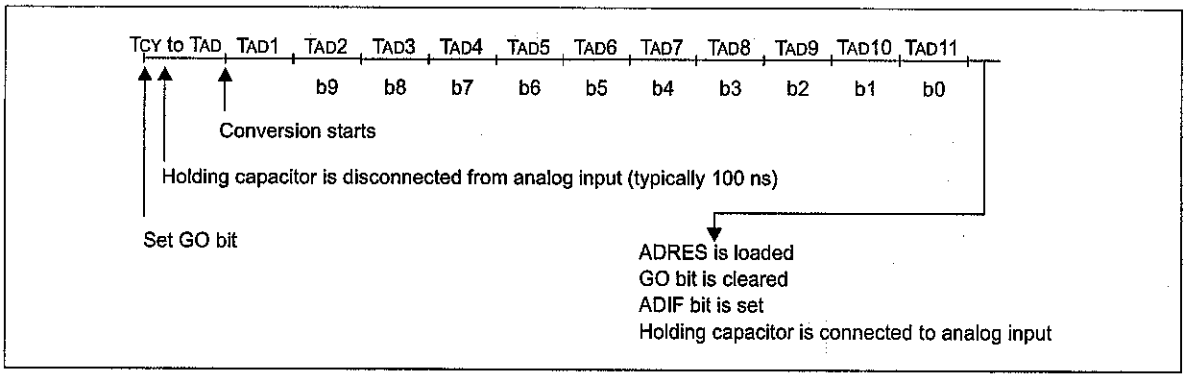
11.4 A/D Conversions

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The A/D result register pair will NOT be updated with the partially completed A/D conversion sample. That is, the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers). After the A/D conversion is aborted, a 2TAD wait is required before the next acquisition is started. After this 2TAD wait, acquisition on the selected channel is automatically started. The GO/$\overline{\text{DONE}}$ bit can then be set to start the conversion.

In Figure 11-3, after the GO bit is set, the first time segment has a minimum of TCY and a maximum of TAD.

> **Note:** The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

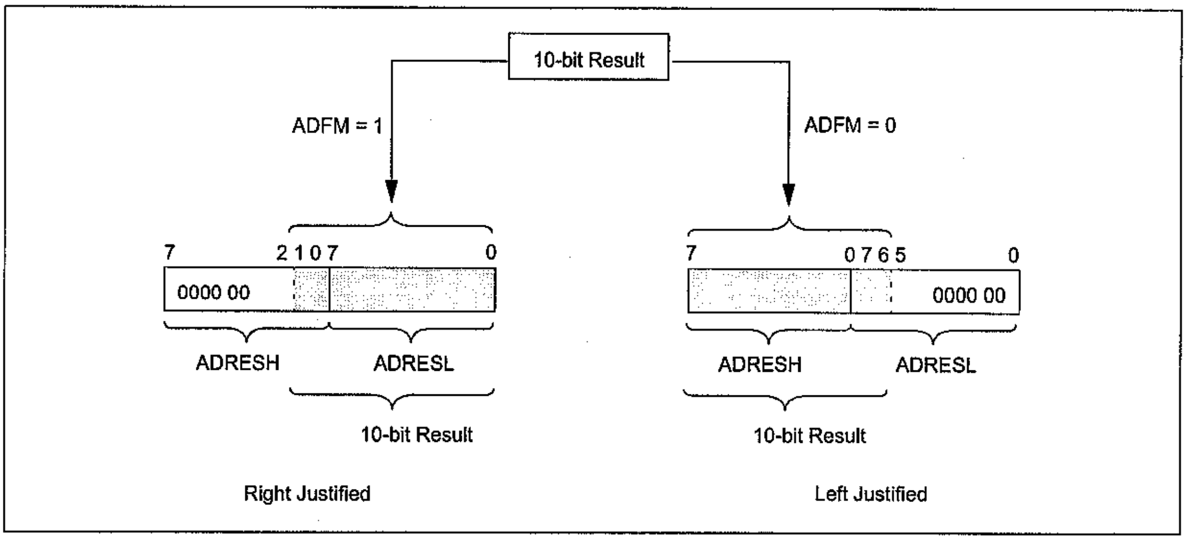
FIGURE 11-3: A/D CONVERSION TAD CYCLES



11.4.1 A/D RESULT REGISTERS

The ADRESH:ADRESL register pair is the location where the 10-bit A/D result is loaded at the completion of the A/D conversion. This register pair is 16-bits wide. The A/D module gives the flexibility to left or right justify the 10-bit result in the 16-bit result register. The A/D Format Select bit (ADFM) controls this justification. Figure 11-4 shows the operation of the A/D result justification. The extra bits are loaded with '0's'. When an A/D result will not overwrite these locations (A/D disable), these registers may be used as two general purpose 8-bit registers.

FIGURE 11-4: A/D RESULT JUSTIFICATION



APPENDIX D
Sample Source Code

```
'877 a/d test routine

.T CIRCUIT:
 11 & 32 = 5VDC
 12 & 31 = 0VDC
 1 = 5VDC
 2 = 0 - 5 VDC analog input (center tap on 20k variable resistor)
 13 & 14 = 4MHz crystal with 18pF capacitors to 0VDC
 15,16,17,18,23,24,25,26 PORTC outputs each to 330 ohm resistor
    in series with LED to 0VDC
*******************************************************

    __config _LVP_OFF & _XT_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF & _BODEN_OFF &
UG_OFF
    list p=16f877

    #include <p16f877.inc>

    ; Start at the reset vector
    org     0x000
    goto    Start
    org     0x004
rrupt
    retfie
't
    bsf     STATUS,RP0              ;bank 1
    bcf     STATUS,RP1
    movlw   H'00'
    movwf   TRISC                   ;portc [7-0] outputs
    clrf    ADCON1                  ;left justified, all inputs a/d
    bcf     STATUS,RP0              ;bank 0
    movlw   B'01000001'             ;Fosc/8 [7-6], A/D ch0 [5-3], a/d on [0]
    movwf   ADCON0

    call    ad_portc
    goto    Main


ortc
                                    ;wait for acquision time (20uS)
                                    ;(non-critical for this test)

    bsf     ADCON0,GO               ;Start A/D conversion

    btfsc   ADCON0,GO               ;Wait for conversion to complete
    goto    Wait

    movf    ADRESH,W                ;Write A/D result to PORTC
    movwf   PORTC                   ;LEDs
    return

    end
```

```
ding two analogue inputs, and display them on the LCD
el Goodwin 2004
vice 16F876

    LIST     p=16F876, W=2, X=ON, R=DEC        ;tell assembler what chip we are
g
    include "P16F876.inc"                      ;include the defaults for the chip
    ERRORLEVEL      0,      -302               ;suppress bank selection messages
    __CONFIG    0x393A                         ;sets the configuration settings
illator type etc.)


            cblock  0x20                       ;start of general purpose registers
                    count
                    count1
                    counta
                    countb
                    LoX
                    Bit_Cntr
                    Timer_H
                    Flags
                    Flags2
                    tmp1
                    tmp2
                    tmp3
                    NumL
                    NumH
                    TenK
                    Thou
                    Hund
                    Tens
                    Ones
                    templcd
                    templcd2
                    Point                      ;position of decimal point
            endc

.PORT       Equ     PORTB
.TRIS       Equ     TRISB
.RS         Equ     0x04                       ;LCD handshake lines
.RW         Equ     0x06
.E          Equ     0x07

'Z          Equ     0x00                       ;set to display leading zeros

.rt of program

            ORG     0x0000
            NOP
            BCF     PCLATH, 3
            BCF     PCLATH, 4
            GOTO    Initialise

            ORG     0x0004
            RETURN

            ORG     0x0005


:ialise     clrf    count
            clrf    PORTA
            clrf    PORTB
            clrf    PORTC
```

```
                clrf     Flags
                BANKSEL ADCON1                          ;disable A2D
                movlw    0x06
                movwf    ADCON1
                BANKSEL PORTA
                                                        ;variables for decimal numbers
lay
                bsf      Flags, LEADZ                   ;show leading zeros
                movlw    0x00                           ;set decimal point position to zero
e)
                movwf    Point


orts            bsf      STATUS,         RP0            ;select bank 1
                movlw    0x00                           ;make all LCD pins outputs
                movwf    LCD_TRIS
                bcf      STATUS,         RP0            ;select bank 0
                call     LCD_Init                       ;setup LCD module
                call     LCD_CurOff                     ;turn cursor off
                call     Init_ADC0                      ;initialise analogue input


                call     LCD_Line1                      ;set to first line
                call     String1
                call     Read_ADC                       ;read analogue input
                call     LCD_Decimal                    ;and display on LCD (in decimal)
                movlw    ' '
                call     LCD_Char                       ;display a space
                movf     NumH, W
                call     LCD_HEX                        ;and display in hexadecimal
                movf     NumL, W
                call     LCD_HEX

                call     Init_ADC1                      ;set for second channel
                call     Delay50                        ;delay to give 10 readings per
nd

d and display second input

                call     LCD_Line2
                call     String2
                call     Read_ADC                       ;read analogue input
                call     LCD_Decimal                    ;and display on LCD (in decimal)
                movlw    ' '
                call     LCD_Char                       ;display a space
                movf     NumH, W
                call     LCD_HEX                        ;and display in hexadecimal
                movf     NumL, W
                call     LCD_HEX

                call     Init_ADC0                      ;set for first channel
                call     Delay50                        ;delay to give 10 readings per
nd

                goto     Main                           ;loop for ever

_ADC0                                                   ;set for AN0
t ADCON0
                movlw    b'10000001'
                movwf    ADCON0
t ADCON1
                BANKSEL ADCON1
```

```
            movlw   b'10000101'
            movwf   ADCON1
            BANKSEL ADCON0
            return

_ADC1                                           ;set for AD1
t ADCON0
            movlw   b'10001001'
            movwf   ADCON0
t ADCON1
            BANKSEL ADCON1
            movlw   b'10000101'
            movwf   ADCON1
            BANKSEL ADCON0
            return

_ADC
            bsf     ADCON0, GO_DONE             ;initiate conversion
            btfsc   ADCON0, GO_DONE
            goto    $-1                         ;wait for ADC to finish

            movf    ADRESH,W
            andlw   0x03
            movwf   NumH
            BANKSEL ADRESL
            movf    ADRESL,W
            BANKSEL ADRESH
            movwf   NumL                        ;return result in NumL and NumH
            return


LES
Table       addwf   PCL         , f
            retlw   0x30
            retlw   0x31
            retlw   0x32
            retlw   0x33
            retlw   0x34
            retlw   0x35
            retlw   0x36
            retlw   0x37
            retlw   0x38
            retlw   0x39
            retlw   0x41
            retlw   0x42
            retlw   0x43
            retlw   0x44
            retlw   0x45
            retlw   0x46

t           addwf   PCL, f
            retlw   'C'
            retlw   'H'
            retlw   '0'
            retlw   ' '
            retlw   0x00

t           addwf   PCL, f
            retlw   'C'
            retlw   'H'
            retlw   '1'
            retlw   ' '
            retlw   0x00
```

```
 of tables

ng1             clrf    count                      ;set counter register to zero
1               movf    count, w                   ;put counter value in W
                call    Xtext                      ;get a character from the text table
                xorlw   0x00                       ;is it a zero?
                btfsc   STATUS, Z
                retlw   0x00                       ;return when finished
                call    LCD_Char
                incf    count, f
                goto    Mess1

ng2             clrf    count                      ;set counter register to zero
2               movf    count, w                   ;put counter value in W
                call    Ytext                      ;get a character from the text table
                xorlw   0x00                       ;is it a zero?
                btfsc   STATUS, Z
                retlw   0x00                       ;return when finished
                call    LCD_Char
                incf    count, f
                goto    Mess2

 routines

tialise LCD
Init            call    LCD_Busy                   ;wait for LCD to settle

                movlw   0x20                       ;Set 4 bit mode
                call    LCD_Cmd

                movlw   0x28                       ;Set display shift
                call    LCD_Cmd

                movlw   0x06                       ;Set display character mode
                call    LCD_Cmd

                movlw   0x0c                       ;Set display on/off and cursor
iand
                call    LCD_Cmd                    ;Set cursor off

                call    LCD_Clr                    ;clear display

                retlw   0x00

ımmand set routine
_Cmd            movwf   templcd
                swapf   templcd,        w          ;send upper nibble
                andlw   0x0f                       ;clear upper 4 bits of W
                movwf   LCD_PORT
                bcf     LCD_PORT, LCD_RS           ;RS line to 1
                call    Pulse_e                    ;Pulse the E line high

                movf    templcd,        w          ;send lower nibble
                andlw   0x0f                       ;clear upper 4 bits of W
                movwf   LCD_PORT
                bcf     LCD_PORT, LCD_RS           ;RS line to 1
                call    Pulse_e                    ;Pulse the E line high
                call    LCD_Busy
                retlw   0x00

_CharD          addlw   0x30                       ;add 0x30 to convert to ASCII
_Char           movwf   templcd
```

```
            swapf   templcd,        w       ;send upper nibble
            andlw   0x0f                    ;clear upper 4 bits of W
            movwf   LCD_PORT
            bsf     LCD_PORT, LCD_RS        ;RS line to 1
            call    Pulse_e                 ;Pulse the E line high

            movf    templcd,        w       ;send lower nibble
            andlw   0x0f                    ;clear upper 4 bits of W
            movwf   LCD_PORT
            bsf     LCD_PORT, LCD_RS        ;RS line to 1
            call    Pulse_e                 ;Pulse the E line high
            call    LCD_Busy
            retlw   0x00

Line1       movlw   0x80                    ;move to 1st row, first column
            call    LCD_Cmd
            retlw   0x00

Line2       movlw   0xc0                    ;move to 2nd row, first column
            call    LCD_Cmd
            retlw   0x00

Line1W      addlw   0x80                    ;move to 1st row, column W
            call    LCD_Cmd
            retlw   0x00

Line2W      addlw   0xc0                    ;move to 2nd row, column W
            call    LCD_Cmd
            retlw   0x00

CurOn       movlw   0x0d                    ;Set display on/off and cursor
and
            call    LCD_Cmd
            retlw   0x00

CurOff      movlw   0x0c                    ;Set display on/off and cursor
and
            call    LCD_Cmd
            retlw   0x00

Clr         movlw   0x01                    ;Clear display
            call    LCD_Cmd
            retlw   0x00

HEX         movwf   tmp1
            swapf   tmp1,   w
            andlw   0x0f
            call    HEX_Table
            call    LCD_Char
            movf    tmp1, w
            andlw   0x0f
            call    HEX_Table
            call    LCD_Char
            retlw   0x00

e_e         bsf     LCD_PORT, LCD_E
            nop
            bcf     LCD_PORT, LCD_E
            retlw   0x00

Busy
            bsf     STATUS, RP0             ;set bank 1
            movlw   0x0f                    ;set Port for input
```

```
        movwf   LCD_TRIS
        bcf     STATUS, RP0             ;set bank 0
        bcf     LCD_PORT, LCD_RS        ;set LCD for command mode
        bsf     LCD_PORT, LCD_RW        ;setup to read busy flag
        bsf     LCD_PORT, LCD_E
        swapf   LCD_PORT, w             ;read upper nibble (busy flag)
        bcf     LCD_PORT, LCD_E
        movwf   templcd2
        bsf     LCD_PORT, LCD_E         ;dummy read of lower nibble
        bcf     LCD_PORT, LCD_E
        btfsc   templcd2, 7             ;check busy flag, high = busy
        goto    LCD_Busy                ;if busy check again
        bcf     LCD_PORT, LCD_RW
        bsf     STATUS, RP0             ;set bank 1
        movlw   0x00                    ;set Port for output
        movwf   LCD_TRIS
        bcf     STATUS, RP0             ;set bank 0
        return

Decimal
        call    Convert
        btfsc   Flags, LEADZ
        goto    LCD_TENK
        movf    TenK, w
        btfss   STATUS, Z
        goto    LCD_TENK
        movf    Thou, w
        btfss   STATUS, Z
        goto    LCD_THOU
        movf    Hund, w
        btfss   STATUS, Z
        goto    LCD_HUND
        movf    Tens, w
        btfss   STATUS, Z
        goto    LCD_TENS
        goto    LCD_ONES
TENK
        movlw   0x05                    ;test if decimal point 5
        subwf   Point, w
        btfss   STATUS    , Z
        goto    NO_DP5
        movlw   '.'
        call    LCD_Char                ;display decimal point
P5      movf    TenK, w
        call    LCD_CharD
        movlw   0x04                    ;test if decimal point 4
        subwf   Point, w
        btfss   STATUS    , Z
        goto    LCD_THOU
        movlw   '.'
        call    LCD_Char                ;display decimal point
THOU
        movf    Thou, w
        call    LCD_CharD
        movlw   0x03                    ;test if decimal point 3
        subwf   Point, w
        btfss   STATUS    , Z
        goto    LCD_HUND
        movlw   '.'
        call    LCD_Char                ;display decimal point
HUND
        movf    Hund, w
        call    LCD_CharD
```

```
            movlw   0x02                    ;test if decimal point 2
            subwf   Point, w
            btfss   STATUS    , Z
            goto    LCD_TENS
            movlw   '.'
            call    LCD_Char                ;display decimal point
TENS
            movf    Tens, w
            call    LCD_CharD
            movlw   0x01                    ;test if decimal point 1
            subwf   Point, w
            btfss   STATUS    , Z
            goto    LCD_ONES
            movlw   '.'
            call    LCD_Char                ;display decimal point
ONES
            movf     Ones, w
            call     LCD_CharD
            return

 of LCD routines


ay routines

y255        movlw   0xff            ;delay 255 mS
            goto    d0
y100        movlw   d'100'          ;delay 100mS
            goto    d0
y50         movlw   d'50'           ;delay 50mS
            goto    d0
y20         movlw   d'20'           ;delay 20mS
            goto    d0
y5          movlw   0x05            ;delay 5.000 ms (20 MHz clock)
            movwf   count1
            movlw   0xE7
            movwf   counta
            movlw   0x04
            movwf   countb
y_0         decfsz  counta, f
            goto    $+2
            decfsz  countb, f
            goto    Delay_0

            decfsz  count1   ,f
            goto    d1
            retlw   0x00

| of Delay routines

s routine downloaded from http://www.piclist.com
'ert:                          ; Takes number in NumH:NumL
                               ; Returns decimal in
                               ; TenK:Thou:Hund:Tens:Ones
    swapf   NumH, w
    iorlw   B'11110000'
    movwf   Thou
    addwf   Thou,f
    addlw   0XE2
    movwf   Hund
    addlw   0X32
    movwf   Ones
```

```
movf    NumH,w
andlw   0X0F
addwf   Hund,f
addwf   Hund,f
addwf   Ones,f
addlw   0XE9
movwf   Tens
addwf   Tens,f
addwf   Tens,f

swapf   NumL,w
andlw   0X0F
addwf   Tens,f
addwf   Ones,f

rlf     Tens,f
rlf     Ones,f
comf    Ones,f
rlf     Ones,f

movf    NumL,w
andlw   0X0F
addwf   Ones,f
rlf     Thou,f

movlw   0X07
movwf   TenK

        ; At this point, the original number is
        ; equal to
        ; TenK*10000+Thou*1000+Hund*100+Tens*10+Ones
        ; if those entities are regarded as two's
        ; complement binary.  To be precise, all of
        ; them are negative except TenK.  Now the number
        ; needs to be normalized, but this can all be
        ; done with simple byte arithmetic.

movlw   0X0A                              ; Ten

addwf   Ones,f
decf    Tens,f
btfss   3,0
goto   Lb1

addwf   Tens,f
decf    Hund,f
btfss   3,0
goto   Lb2

addwf   Hund,f
decf    Thou,f
btfss   3,0
goto   Lb3

addwf   Thou,f
decf    TenK,f
btfss   3,0
goto   Lb4

retlw   0x00
```

```
end
```