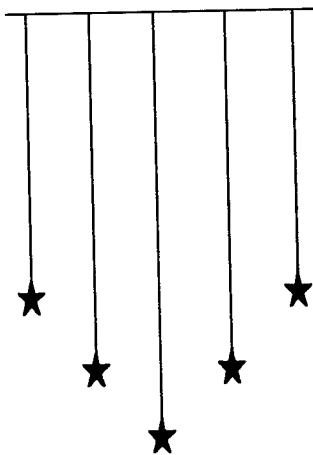
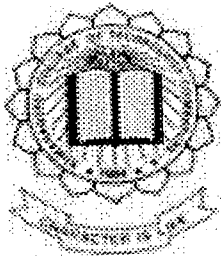


# AUTOMATED ENERGY RECORDING



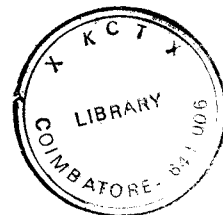
2002 - 2003

PROJECT REPORT

P-914

Submitted by

**Arunn. C**  
**Priya**  
**Prashant**  
**Manivannan**



Guided by

**Mrs.D.Somasundareswari M.**  
Senior Lecturer, Dept. of E

In partial fulfillment of the requirements for  
award of the degree of Bachelor of Engineering,  
Electrical and Electronics branch of Bharathiar University, Coimbatore

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING  
KUMARAGURU COLLEGE OF TECHNOLOGY  
COIMBATORE - 641 006



DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING  
KUMARAGURU COLLEGE OF TECHNOLOGY  
COIMBATORE – 641 006



ISO 9001:2000

## CERTIFICATE

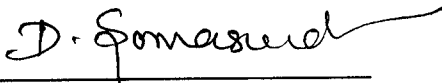
*This is to certify that the Project Report entitled*

**“AUTOMATED ENERGY RECORDING”**

*has been submitted by*

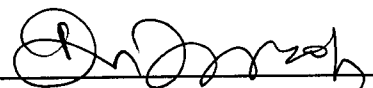
ARUNN C.K	99EEE09
PRIYA P	99EEE38
PRASHANT B	99EEE37
MANIVANNAN M	99EEE26

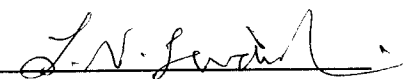
*in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Electrical and Electronics branch of Bharathiar University, Coimbatore during the academic year 2002 - 2003*

  
\_\_\_\_\_  
*Project Guide*

  
\_\_\_\_\_  
*Head of Department*

*Certified that the candidate with University Register No. \_\_\_\_\_  
was examined in project work Viva – Voce Examination on \_\_\_\_\_*

  
\_\_\_\_\_  
*Internal Examiner*

  
\_\_\_\_\_  
*External Examiner*

## **ACKNOWLEDGEMENT**

We wish to express our deep sense of gratitude and indebtedness to our project guide Mrs.D.Somasundareswari M.E.,MISTE,AMIE,MSSI, Senior Lecturer, Department of Electrical and Electronics Engineering, for her invaluable guidance and constructive suggestions, the keen and constant inspiration and interest evinced by her at all stages right from the inception to the completion.

We are highly grateful to Mr.T.M.Kameswaran B.E., M.Sc.,(Engg), Ph.D., MISTE, Sr.M.I.E.E.E., F.I.E, Head of the Department, Electrical and Electronics Engineering, for his enthusiasm and encouragement which has been instrumental in the success of the project.

We express our heartfelt gratitude to our principal Dr.K.K.Padmanabhan, B.Sc.,(Engg), M.Tech., Ph.D., F.I.E., for providing the necessary facilities for the successful completion of the project.

We are also indebted to the staff members of Electrical and Electronics Department for their cooperation. We are thankful to our student friends for their encouragement and help during the course of the project.

## **SYNOPSIS**

In the present scenario, each and every analog system present in the world is getting digitized and automated. The need for automation comes in due to precision and accuracy of the results obtained.

Taking the above factor into consideration, in our project we have implemented the digitalization and automation of the analog energy reading obtained for a single phase supply. In order to reduce the difficulties faced in manual recording of energy readings it is necessary to automate the energy recording. This automation is implemented using PIC 16F877 microcontroller and the energy units recorded are transferred to the microcontroller temporarily and then sent to the personal computer. The main computer has the control of supplying power to the load. By energizing or de-energizing the relay connected to the load, the power supplied to the load can be controlled. The relay obtains the signals from the personal computer through the microcontroller.

# CONTENTS

**CERTIFICATE**

**ACKNOWLEDGEMENT**

**SYNOPSIS**

**CONTENTS**

## **CHAPTER I**

**PAGE NO.**

### **INTRODUCTION**

1.1 ANALOG ENERGY METER	1
1.2 PRINCIPLE OF OPERATION	1
1.3 DISADVANTAGES	2
1.4 NEED FOR AUTOMATION	2

## **CHAPTER II**

### **HARDWARE**

2.1 SCHEMATIC DIAGRAM	4
2.2 POWER SUPPLY UNIT	5
2.2.1 RECTIFIER	8
2.2.2 FILTER	8
2.2.3 REGULATORS	9
2.3 VOLTAGE & CURRENT MEASUREMENT	
2.3.1 SPECIFICATIONS	10
2.3.2 PRECISION RECTIFIER	10
2.4 POWER FACTOR MEASUREMENT	12
2.5 RELAY	13
2.5.1 RELAY CIRCUIT	13
2.6 LCD DISPLAY	15

## **CHAPTER III**

### **MICROCONTROLLER UNIT**

3.1 PIC 16F877 ARCHITECTURE	16
3.2 MEMORY ORGANIZATION	
3.2.1 PROGRAM MEMORY ORGANIZATION	18
3.2.2 DATA MEMORY ORGANIZATION	19
3.3 I/O PORTS	21
3.4 TIMERS	
3.4.1 TIMER 0 MODULE	23
3.4.2 TIMER 1 MODULE	23
3.4.3 TIMER 2 MODULE	24
3.5 ANALOG-TO-DIGITAL CONVERTER	25

## **CHAPTER IV**

### **DATA TRANSMISSION**

4.1 SERIAL DATA TRANSMISSION	27
4.2 ELECTRICAL SPECIFICATION	28
4.3 RS232-C STANDARD	31
4.4 RS232-C PINOUTS	32

## **CHAPTER IV**

### **SOFTWARE**

5.1 PIC PROGRAM	34
5.2 C PROGRAM	47

## **CHAPTER V**

CONCLUSION	50
FURTHER DEVELOPMENTS	51
BIBILIOGRAPHY	52
APPENDIX	53

# **CHAPTER I**

## **INTRODUCTION**

### **1.1 ANALOG ENERGY METER**

An Energy Meter is a device that is used to measure the energy consumed by the electrical equipments connected to the power supply. The commonly used energy meter for single phase supply is the Single Phase Induction Type Energy Meter.

### **1.2 PRINCIPLE OF OPERATION**

The principle of operation of analog equipments mainly depends on the magnetic fields produced. In analog energy meters the eddy currents produced due to magnetic flux plays an important role in energy measurement. The Single Phase Energy Meter consists of two coils namely, the Current coil and the Pressure coil. The Current coil is made up of few turns of thick wire and is connected in series with the load. The pressure coil is made up of many turns of thin wires and is connected in parallel with the supply. An aluminium disc mounted on a spindle is placed between two coils. The two fluxes produced by the two coils interact with aluminium disc and produces eddy currents in the disc and hence the disc

starts rotating. The speed of disc is proportional to the energy consumed by the load.

### **1.3 DISADVANTAGES**

The main disadvantage in using analog energy meter is that it basically depends upon the magnetic field for energy measurement. Certain disadvantages present in analog meters are

- ❖ The life time of analog meters are very less
- ❖ The readings obtained are not precise and accurate
- ❖ The readings registered can be easily tampered
- ❖ Presence of mechanical moving parts

### **1.4 NEED FOR AUTOMATION**

Due to the above mentioned difficulties it is necessary to switch over to digital measurement. In case of digital measurement the readings obtained are precise and accurate. Also the digital meters don't suffer the disadvantages of magnetic field which leads to less accurate measurement. Since no mechanical parts is involved the life time expectancy is also very high.



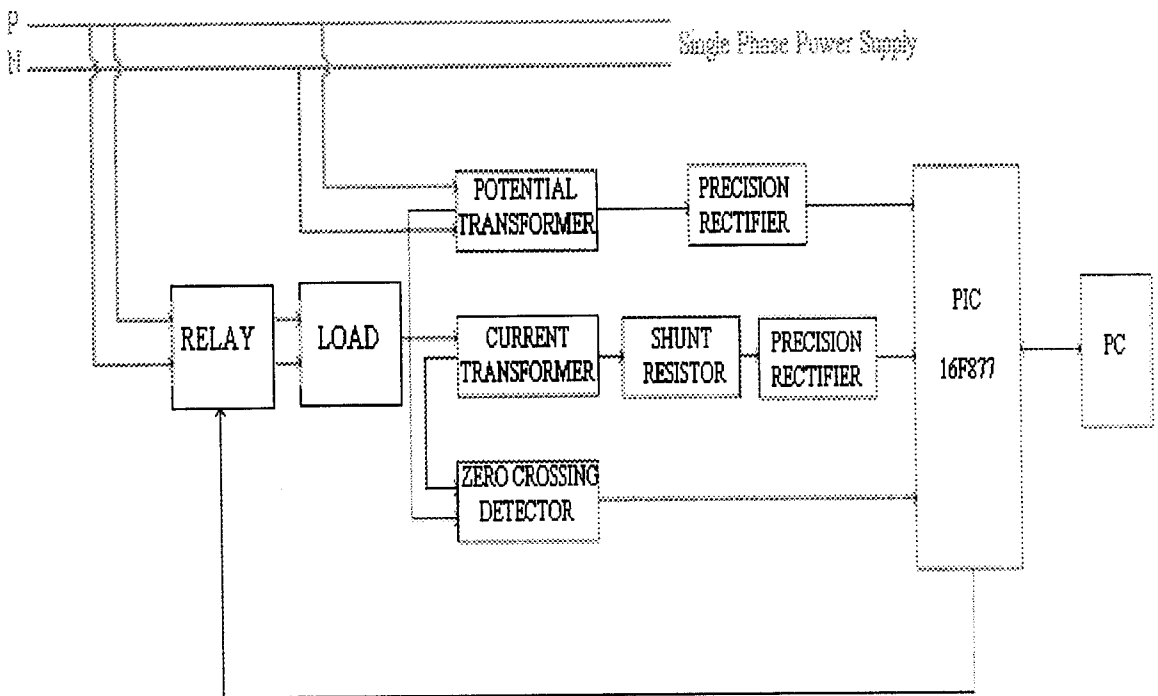
In the present scenario the world has become a networked structure and every unit is being automated. Any bit of data available can be sent to any place in the world. Hence in our project, the energy reading obtained is sent to the computer by using a microcontroller. This leads to reduction in cost and man labour.

## CHAPTER II

### HARDWARE

#### 2.1 SCHEMATIC DIAGRAM

The schematic diagram of the Automated Energy Meter is shown below



The Voltage and Current readings are obtained from the Potential and Current Transformers respectively and AC to DC conversion of the readings are done in the Precision Rectifier. The DC voltage obtained is in the range of 0-5V and the obtained readings are sent to the PIC microcontroller. The phase angle difference between the current and

voltage signals is obtained using the Zero Crossing Detector. The resultant value ranging from 0-5V gives the Power Factor and the Power Factor reading is also sent to the PIC microcontroller. Calculation of the total power consumed is done by using the formula:

$$P = V * I * \text{COS } \Phi$$

From this, energy is calculated and the resultant value is shown on the LCD display. Also the data is sent to the Personal Computer by serial data transmission using RS232 interface. The Tariff rate for the energy consumed is calculated in the computer and displayed on the monitor. The relay is used to control the Power supply to the load connected. The relay is controlled by the Personal Computer.

## 2.2 POWER SUPPLY UNIT

Since all electronic circuits work only with low D.C. voltage we need a power supply unit to provide the appropriate voltage supply. The Power Supply unit Comprises of two circuits, one providing a +5V supply and the other providing +12V and -12V supply. The Circuit Diagrams for the same are shown in Fig 1(a) and Fig 1(b). This unit consists of transformers, rectifiers, filters and regulators. The input A.C. voltage, typically 230V rms is connected to a transformer to obtain the desired step-

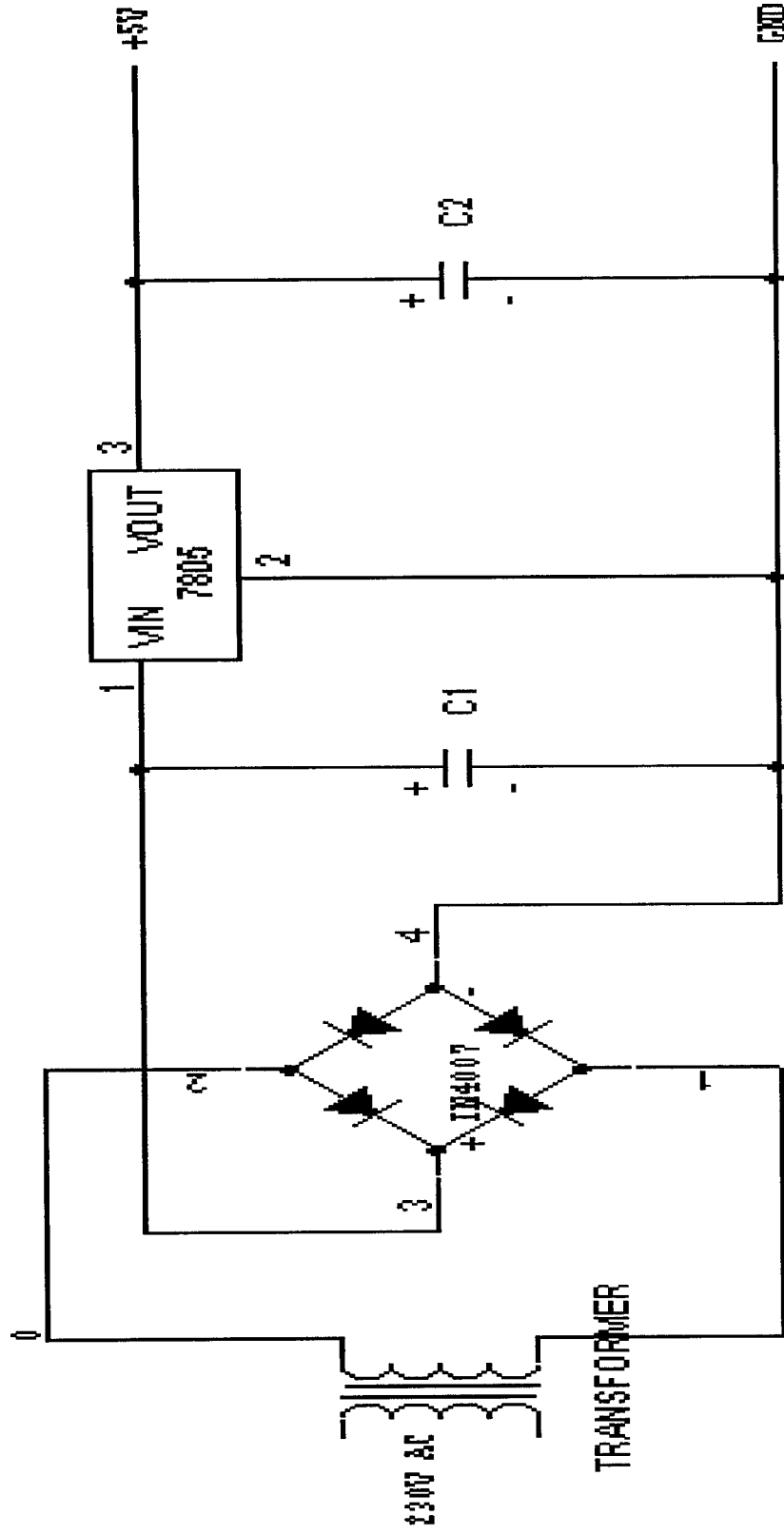


Fig 1(a) +5V Supply

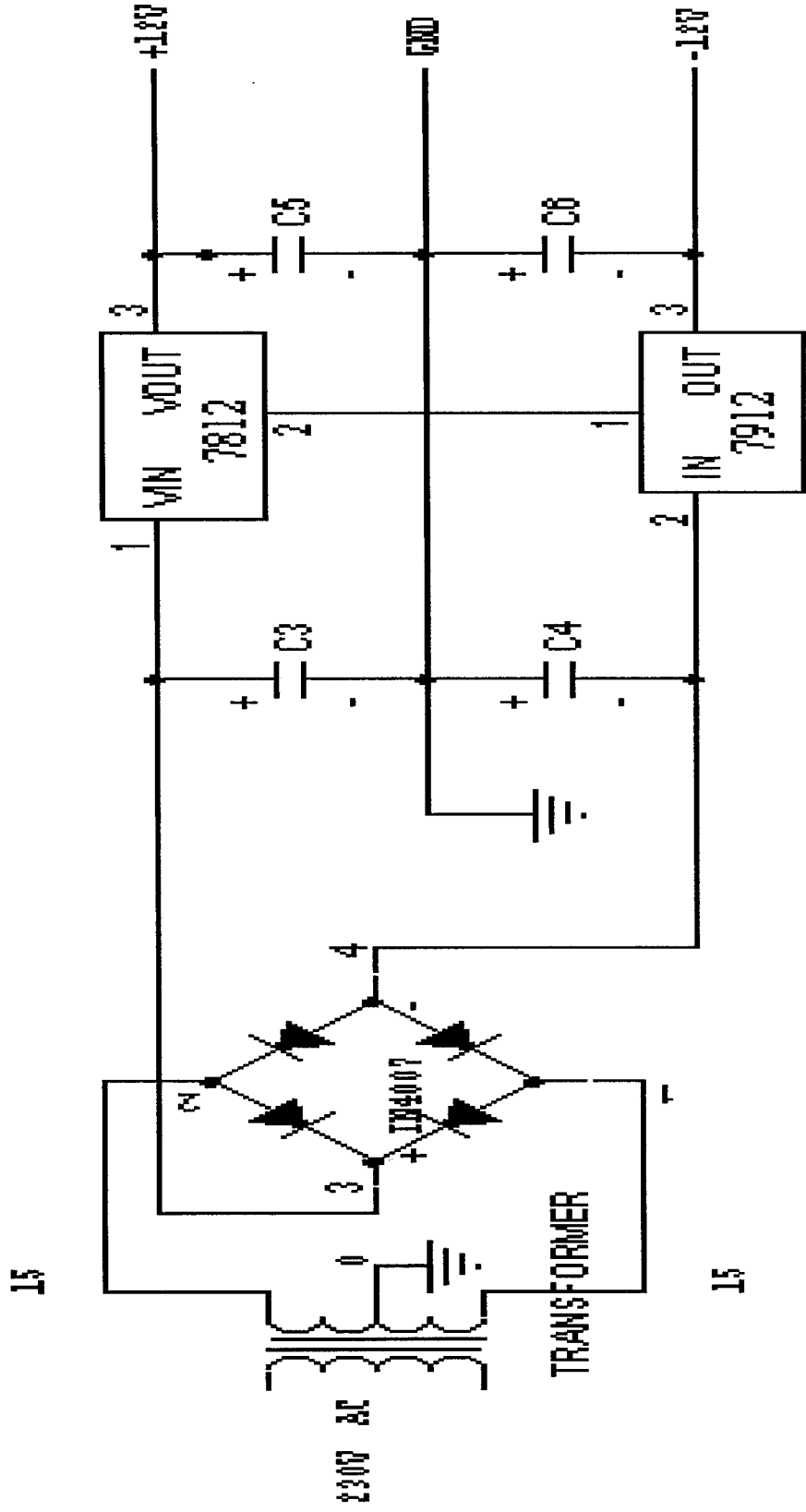


Fig 1(b) +12V and -12V supply

down AC voltage. A diode rectifier then provides a full-wave rectified voltage that is initially filtered by a simple capacitor filter to produce a DC voltage. This resulting DC voltage usually has some ripple or AC voltage variations. A regulator circuit can use this DC input to provide DC voltage that not only has much less ripple voltage but also remains the same DC value even if the DC voltage varies, or when the load connected to the output DC voltage changes.

### **2.2.1 RECTIFIER**

The DC level obtained from a sinusoidal input can be improved 100% using a process called full-wave rectification. It uses 4 diodes in a bridge configuration. From the basic bridge configuration we see that two diodes (say D2 & D3) are conducting while the other two diodes (D1 & D4) are in “off” state during the period  $t = 0$  to  $T/2$ . Accordingly for the negative half cycle of the input the conducting diodes are D1 & D4. Thus the polarity across the load is the same.

### **2.2.2 FILTER**

The filter circuit used here is the capacitor filter circuit where a capacitor is connected at the rectifier output, and a DC is

obtained across it. The filtered waveform is essentially a DC voltage with negligible ripples, which is ultimately fed to the load.

### **2.2.3 REGULATORS**

The voltage regulator is a device, which maintains the output voltage constant irrespective of the change in supply variations, load variation and temperature changes. The fixed voltage regulator is a three terminal device which has an unregulated dc input voltage,  $V_i$ , applied to one input terminal, a regulated output dc voltage,  $V_o$ , from a second terminal, with the third terminal connected to ground. For a selected regulator, IC device specifications list a voltage range over which the input voltage can vary to maintain a regulated output voltage over a range of load current. The specifications also list the amount of output voltage change resulting from a change in load current or in input voltage.

Here we use two fixed voltage regulators namely LM 7812, LM 7805 and LM7912. The IC 7812 is a +12V regulator, IC 7912 is a -12V regulator and IC 7805 is a +5V regulator. The series 78 regulators provide fixed regulated voltages from 5 to 24 V. Similarly the series 79 regulators provide regulated voltages from -5 to -24 V.

## **2.3 VOLTAGE & CURRENT MEASUREMENT**

### **2.3.1 SPECIFICATIONS**

#### **a) Potential Transformer**

Primary : 230V AC

Secondary : 6V AC

#### **b) Current Transformer**

Primary : 15 A

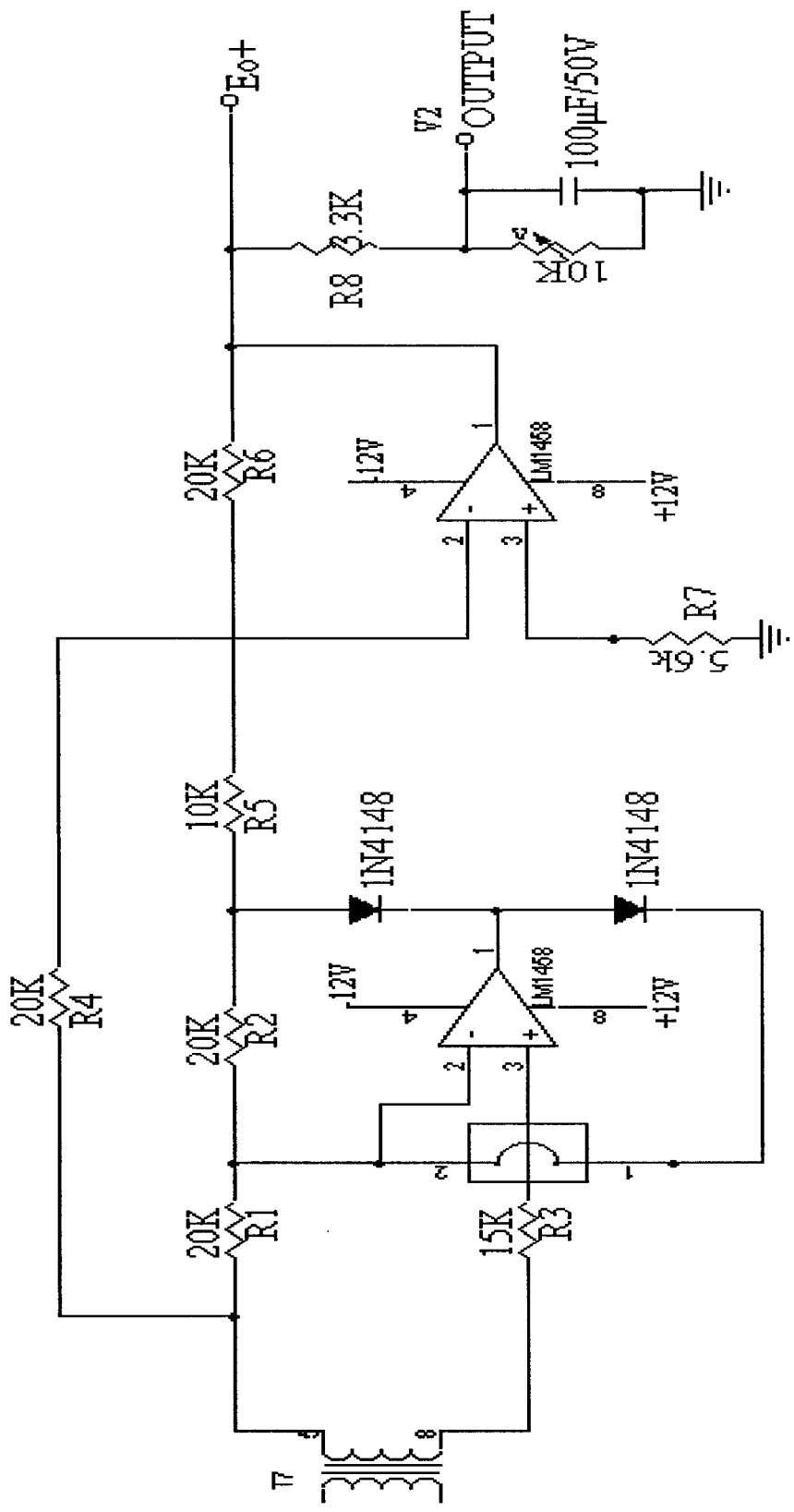
Secondary : 5 A



### **2.3.2 PRECISION RECTIFIER**

Precision rectifier is a full wave rectifier device which is used to rectify the input AC voltage. Normal diode rectification circuits output only RMS value of the output voltage. Since Peak value of the output voltage is required we go in precision rectifier. By providing a shunt resistor of 56 ohms, the value of current consumed can be obtained as a corresponding voltage signal. The circuit diagram for the precision rectifier is shown in Fig. For current measurement we provide a 56 ohm resistor between points 5 and 8 shown in the circuit diagram.





## 2.4 POWER FACTOR MEASUREMENT

The difference in phase angle between the voltage and current can be calculated by using the zero-crossing detector. The circuit diagram of the Zero Crossing Detector is shown in Fig 4

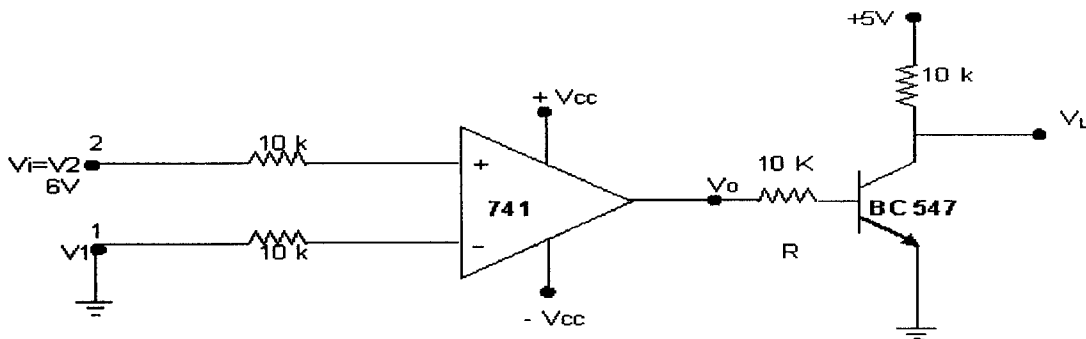


Fig 4 Zero Crossing Detector

The sinusoidal waveforms of voltage and current is given to the zero-crossing detector, the output will be the corresponding square wave. These signals are then given to an NAND Gate with one of the input inverted (say for corresponding voltage signal of current).

The NAND gate will give the output when both the inputs are high. When one of the inputs to the Gate is inverted, then the gate will be conducting for the moment when the inverted input is low and the other input is high. We can calculate this time period. From that we can easily calculate the power factor using the pre-calculated table.

Say for example if the conduction period of the NAND gate is 700  $\mu$ sec then the power factor will be 0.9.

## **2.5 RELAY**

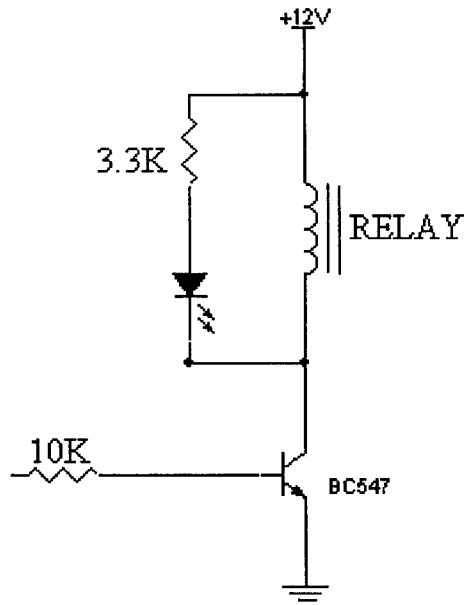
A relay is a switch worked by an electromagnet. It is useful if we want a small current in one circuit to control another circuit containing a device such as a lamp or electric motor which requires a large current, or if we wish several different switch contacts to be operated simultaneously.

When the controlling current flows through the coil, the soft iron core is magnetized and attracts the L-shaped soft iron armature. This rocks on its pivot and opens, closes or changes over, the electrical contacts in the circuit being controlled as it closes the contacts. The current needed to operate a relay is called the pull-in current and the dropout current in the coil when the relay just stops working.

### **2.5.1 RELAY CIRCUIT**

In our project we use relays to cut the power supply going to the load. The relay is controlled by the computer via Microcontroller. The signals sent from the computer are sensed by the

Microcontroller and it sends necessary signals to activate the relay. The activation of relay is done using the relay circuit shown in Fig 5.



In this circuit transistor BC547 is used as a switch. The control signal is given to the base terminal of the transistor. The collector is attached to the relay coil. There are two types of relays.

- ❖ Normally closed
- ❖ Normally opened

We are using normally closed type relay. When the controller output from the PC is low, the transistor will be in the OFF state, so the relay is not energized. When the controller output from the PC is high the transistor will be in the ON state, so the relay is energized and the power supply is cut-off from the load. When the relay is de-energized the power supply to the load is resumed. So according to the controller output the power supply to the load is varied.

## 2.6 LCD DISPLAY

In our project, the readings obtained are shown on an LCD display. An LCD consists of two glass panels, with the liquid crystal material sandwiched between them. The inner surface of the glass plates are coated with transparent electrodes which define the character, symbols or patterns to be displayed. Polymeric layers are present in between the electrodes and the liquid crystal, which makes the liquid crystal molecules to maintain a defined orientation angle.

Polarisers are pasted outside the two glass panels. These polarisers would rotate the light rays passing through them to a definite angle, in a particular direction. When the LCD is in the off state, light rays are rotated by the two polarisers and the liquid crystal, such that the light rays come out of the LCD without any orientation, and hence the LCD appears transparent.

When sufficient voltage is applied to the electrodes, the liquid crystal molecules would be aligned in a specific direction. The light rays passing through the LCD would be rotated by the polarisers, which would result in activating / highlighting the desired characters.

## **CHAPTER III**

### **MICROCONTROLLER UNIT**

#### **3.1 PIC 16F877 ARCHITECTURE**

The Microcontroller that we have used is PIC 16F877. The PIC 16 series comprises of PIC16F873, PIC16F874, PIC16F876 and PIC16F877. Out of this PIC 16F873 and PIC16F874 are 28-pin packages. In our Project we have used the PIC16F877 for its outstanding features like

- High performance RISC CPU
- Only 35 single word instructions
- 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
- Interrupt capability (up to 14 sources)
- Power saving SLEEP mode
- Low power, high speed CMOS FLASH/EEPROM technology
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA

The block diagram of PIC16F877 is shown in Fig 6

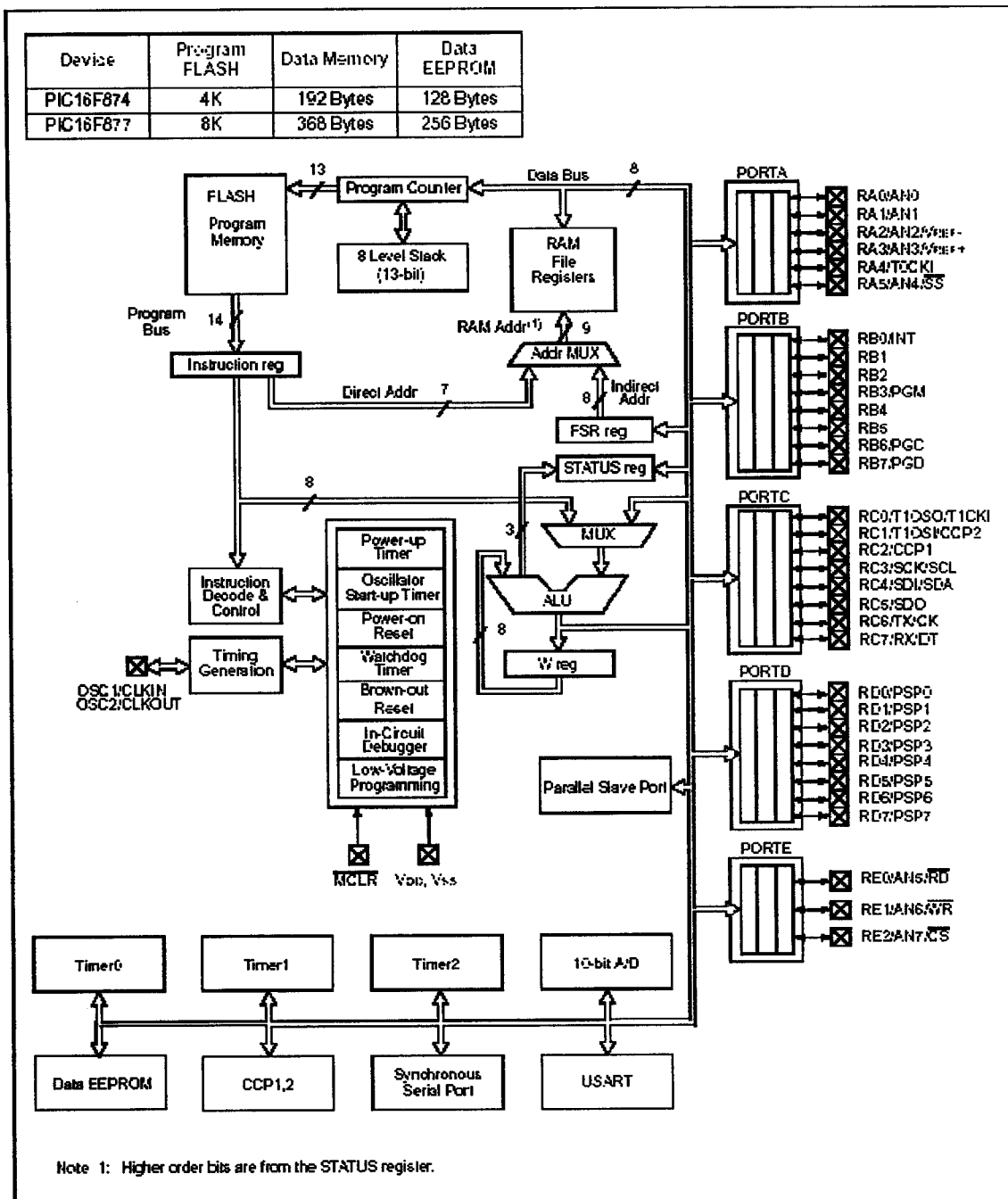


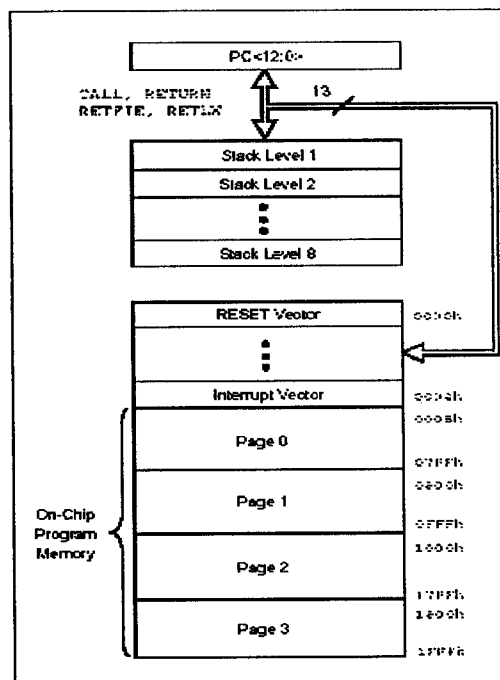
Fig 6 PIC 16F877 Block Diagram

## 3.2 MEMORY ORGANIZATION

There are three memory blocks in the PIC16F877 Microcontroller. The Program Memory and Data Memory have separate buses so that concurrent access can occur in both.

### 3.2.1 PROGRAM MEMORY ORGANIZATION

The PIC16F87X devices have a 13-bit program counter capable of addressing an 8K x 14 program memory space. The PIC16F877 device has 8K x 14 words of FLASH program memory. 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. The program memory map and stack organization of 16F877 is shown in Fig 7.





### 3.2.2 DATA MEMORY ORGANIZATION

The data memory is partitioned into multiple banks which contain the General Purpose Registers and the Special Function Registers. Bits RP1 and RP0 are the bank select bits.

RP1	RP0	BANK
0	0	0
0	1	1
1	0	2
1	1	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.

The General Purpose Register file can be accessed either directly, or indirectly through the File Select Register (FSR). The Special Function Registers are registers used by the CPU and peripheral

modules for controlling the desired operation of the device. These registers are implemented as static RAM. The Special Function Registers can be classified into two sets: core (CPU) and peripheral. The registers present in Special Function Registers are explained in brief

The **STATUS** register contains the arithmetic status of the ALU, the RESET status and the bank select bits for data memory.

The **OPTION\_REG** Register is a readable and writable register, which contains various control bits to configure the TMR0 prescaler/WDT postscaler, the External INT Interrupt, TMR0 and the weak pull-ups on PORTB.

The **INTCON** Register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow, RB Port change and External RB0/INT pin interrupts.

The **PIE1** register contains the individual enable bits for the peripheral interrupts.

The **PIR1** register contains the individual flag bits for the peripheral interrupts.

The **PIE2** register contains the individual enable bits for the CCP2 peripheral interrupt, the SSP bus collision interrupt, and the EEPROM write operation interrupt.

The **PIR2** register contains the flag bits for the CCP2 interrupt, the SSP bus collision interrupt and the EEPROM write operation interrupt.

The **Power Control (PCON)** Register contains flag bits to allow differentiation between a Power-on Reset (POR), a Brown-out Reset (BOR), a Watchdog Reset (WDT), and an external MCLR Reset.

### 3.3 I/O PORTS

There are totally 6 ports present in PIC 16F877.

They are Port A, Port B, Port C, Port D, Port E and Parallel Slave Port.

**PORTA** is a 6-bit wide, bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

**PORTB** is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISB bit (= 0) will make the

corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

**PORTC** is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

**PORTD** is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. PORTD can be configured as an 8-bit wide microprocessor port (parallel slave port) by setting control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

**PORTE** has three pins (RE0/RD/AN5, RE1/WR/AN6, and RE2/CS/AN7) which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. The PORTE pins become the I/O control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set.

PORTD operates as an 8-bit wide **Parallel Slave Port** or microprocessor port, when control bit PSPMODE (TRISE<4>) is set. In Slave mode, it is asynchronously readable and writable by the external world through RD control input pin RE0/RD and WR control input pin RE1/WR.

## **3.4 TIMERS**

### **3.4.1 TIMER 0 MODULE**

The Timer0 module timer/counter has the following features:

- ❖ 8-bit timer/counter
- ❖ Readable and writable
- ❖ 8-bit software programmable prescaler
- ❖ Internal or external clock select
- ❖ Interrupt on overflow from FFh to 00h
- ❖ Edge select for external clock

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit T0IF (INTCON<2>). The interrupt can be masked by clearing bit T0IE (INTCON<5>). Bit T0IF must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP, since the timer is shut-off during SLEEP.

### **3.4.2 TIMER 1 MODULE**

The Timer1 module is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L), which are readable

and writable. The TMR1 Register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR1 Interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing TMR1 interrupt enable bit TMR1IE (PIE1<0>). Timer1 can operate in one of two modes:

- ❖ As a timer

- ❖ As a counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>). In Timer mode, Timer1 increments every instruction cycle. In Counter mode, it increments on every rising edge of the external clock input. Timer1 can be enabled/disabled by setting/clearing control bit TMR1ON (T1CON<0>).

### **3.4.3 TIMER 2 MODULE**

Timer2 is an 8-bit timer with a prescaler and a postscaler. It can be used as the PWM time-base for the PWM mode of the CCP module(s). The TMR2 register is readable and writable, and is cleared on any device RESET.

The Timer2 module has an 8-bit period register, PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon RESET.

### **3.5 ANALOG-TO-DIGITAL CONVERTER**

The Analog-to-Digital (A/D) Converter module has eight inputs for the 40-pin device. 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 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 ADCON0 register controls the operation of the A/D module. The ADCON1 register configures the functions of the port pins. The port pins can be configured as analog inputs or as digital I/O.



## **CHAPTER IV**

### **DATA TRANSMISSION**

#### **4.1 SERIAL DATA TRANSMISSION**

The need to send and receive data from one location to another is of vital importance. The most cost-effective way to meet this demand is to send the data as a serial stream of bits in order to reduce the cost (and bulk) of multiple conductor cable. Optical fiber bundles, which are physically small, can be used for the parallel data transmission. However, the cost incurred for the fibers, the terminations, and the optical interface are so high.

So the optimum way of transmitting data is by serial data transmission. Special integrated circuits dedicated solely to serial data transmission and reception appeared commercially in the early 1970s. These chips, commonly called universal asynchronous receiver transmitters or USARTS, perform all the serial data transmission and reception timing tasks. The most popular data communication scheme still in use today is the transmission of serial 8-bit ASCII-coded characters at predefined bit rates of 300 to 19,200 bits per second.

The serial port transmits a '1' as -3 to -25 volts and a '0' as +3 to +25 volts whereas a parallel port transmits a '0' as 0v and a '1' as 5v. Therefore the serial port can have a maximum swing of 50V compared to the parallel port which has a maximum swing of 5 Volts. Therefore cable loss is not going to be as much of a problem for serial cables than they are for parallel.

It requires less number of wires than parallel transmission. If the device needs to be mounted a far distance away from the computer then 3 core cable (Null Modem Configuration) is cheaper than running 19 or 25 core cable. Many of the Microcontrollers have in built SCI (Serial Communications Interfaces) which can be used to talk to the outside world. Serial Communication reduces the pin count of these Microcontrollers. Only two pins are commonly used, Transmit Data (TXD) and Receive Data (RXD) compared with at least 8 pins if you use a 8 bit Parallel method.

## **4.2 ELECTRICAL SPECIFICATION**

The electrical Specifications of Serial Port are given as follows

- ❖ A "Space" (logic 0) will be between +3 and +25 Volts.

- ❖ A "Mark" (Logic 1) will be between -3 and -25 Volts.
- ❖ The region between +3 and -3 volts is undefined.
- ❖ An open circuit voltage should never exceed 25 volts. (In Reference to GND)
- ❖ A short circuit current should not exceed 500mA. The driver should be able to handle this without damage.

<b>Name</b>	<b>Address</b>	<b>IRQ</b>
COM 1	3F8	4
COM 2	2F8	3
COM 3	3E8	4
COM 4	2E8	3

### ***Standard Port Addresses***

Above is the standard port addresses. These should work for most P.C s. The base addresses for the COM ports can be read from the BIOS Data Area.

<b>Start Address</b>	<b>Function</b>
0000:0400	COM1's Base Address
0000:0402	COM2's Base Address
0000:0404	COM3's Base Address
0000:0406	COM4's Base Address

### ***COM Port Addresses in the BIOS Data Area***

There are two basic types of serial communications, Synchronous and Asynchronous. Asynchronous means "no synchronization", and thus does not require sending and receiving idle characters. However, the beginning and end of each byte of data must be identified by start and stop bits. The start bit indicates when the data byte is about to begin and the stop bit signals when it ends. The requirement to send these additional two bits cause asynchronous communications to be slightly slower than synchronous however it has the advantage that the processor does not have to deal with the additional idle characters.

An asynchronous line that is idle is identified with a value of 1, (also called a mark state). By using this value to indicate that no data is currently being sent, the devices are able to distinguish between an idle state and a disconnected line. When a character is about to be transmitted, a start bit is sent. A start bit has a value of 0, (also called a space state). Thus, when the line switches from a value of 1 to a value of 0, the receiver is alerted that a data character is about to come down the line.

An example serial data transmission through Asynchronous mode is shown in Fig 8. Here a data packet corresponding to ASCII character 'A' is sent through the wires. Here we have one start bit, seven data bits, one parity bit and two stop bits.

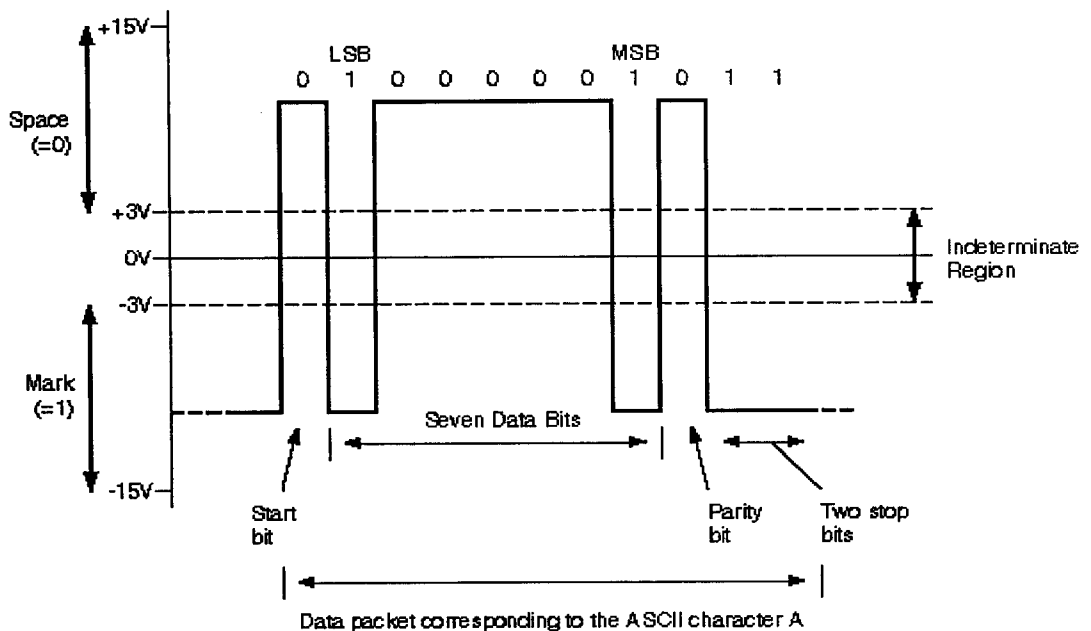


Fig 8 Serial Data Transmission

### 4.3 RS232-C STANDARD

The serial port to which the mouse connects is to the EIA RS232-C standard. This standard defines the output voltage for a 'logic-high' to be -3 to -15 volts. A 'logic-low' is to be in the range +3 to +15 volts. Common values are  $\pm 5V$  and  $\pm 12V$ . Because a logical true is represented by a negative voltage, the start bits are at the high voltage level, the and stop bits are at the low voltage level, and data bits that are logically true are transmitted at the negative voltage level as well. However, all of the handshake lines coming out of or into the PC are complimented which means that setting, for example, RTS to true in software results in the

compliment of RTS which appears at the output port being at the positive voltage level.

#### 4.4 RS232-C PINOUTS

Two connectors are normally used for RS232-C, 25-pin D-type and 9-pin D-type, the 9-pin variety being most common. The DTE end (computer end) should have the pins (male) and the DCE end (mouse end) should have the sockets into which the pins fit (female). The pinout diagram for the 9-pin D-type connector is shown in Fig 9

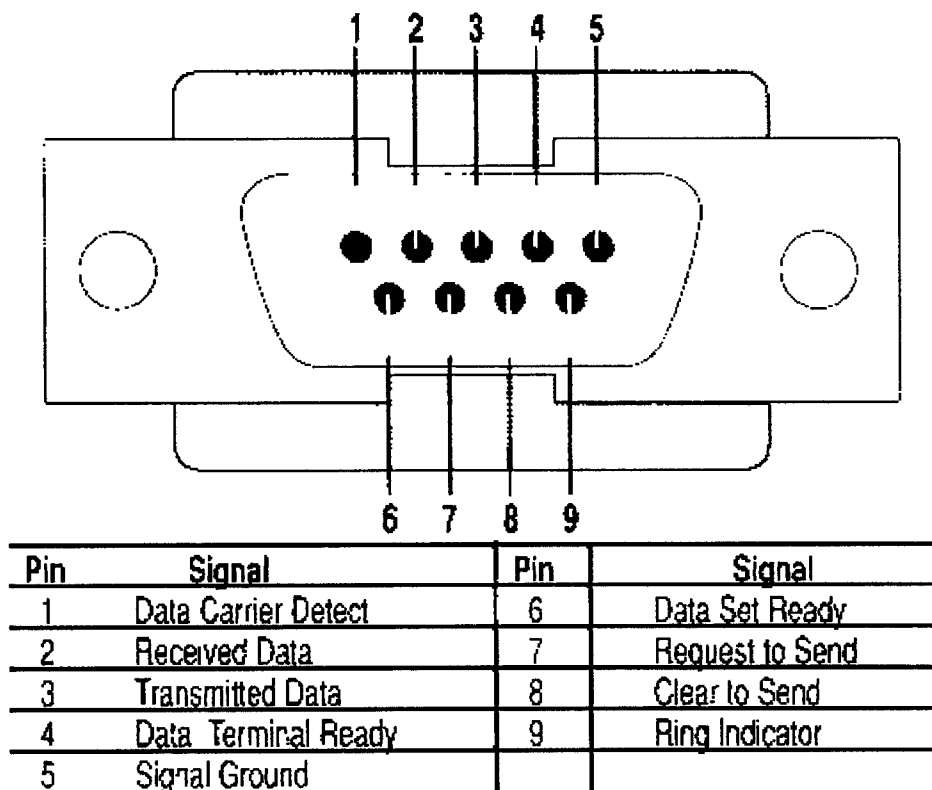


Fig 9 RS232-C pinouts

The below tabular column shows the names, direction of data transfer and the description of each pin

<b>Pin Number</b>	<b>Name</b>	<b>Direction</b>	<b>Description</b>
1	nCD	PC<-Peripheral	Carrier Detect
2	RXD	PC<- Peripheral	Received Data
3	TXD	PC->Peripheral	Transmitted Data
4	nDTR	PC->Peripheral	DataTerminalReady
5	SGND	na	Signal Ground
6	nDSR	PC<- Peripheral	Data Set Ready
7	nRTS	PC->Peripheral	Request to Send
8	nCTS	PC<- Peripheral	Clear to Send
9	nRI	PC<- Peripheral	Ring Indicator

na = not applicable

Prefix of n means signal is inverse-logic (in this case high output voltage for logical true.)

## CHAPTER V

### SOFTWARE

#### 5.1 PIC PROGRAM

```
/*  
This program will convert the 3- analog inputs into digital using inbuilt  
analog to digital converter technology. This will multiply the 3- ADC values  
and display using LCD. The software is written in Hi-Tech C  
*/
```

```
#include <pic.h>  
#include "delay.c"  
#include "delay.h"  
  
static bit PORTB_1 @ ((unsigned)&PORTB*8+1); // Register select  
static bit PORTB_2 @ ((unsigned)&PORTB*8+2); // Enable  
static bit PORTB_4 @ ((unsigned)&PORTB*8+4); // Enable  
static bit PORTB_5 @ ((unsigned)&PORTB*8+5); // Enable
```

```
port_init();  
lcd_init();  
ser_init();  
ADC_pf_init();  
ADC_volt_init();  
HTB_1();  
Disp_volt();  
ADC_curr_init();  
HTB_2();  
Disp_curr();  
Cal_pwr();  
HTB_4();  
disp_mesg();  
Disp_pf();  
main 1();
```



```
getin();
get_val();
rd_clk();
wr_clk();
ser_get();
```

```
#define DIEN ((PORTB_1 = 0), (PORTB_2=1), (PORTB_4=0),
(PORTB_5=0), (PORTB_5=1))
#define DIS ((PORTB_1 = 1), (PORTB_2=0), (PORTB_4=0),
(PORTB_5=0), (PORTB_5=1))
```

```
/*******
```

```
#define PERIOD 1000000 // period in µs - one second here
#define XTAL 4000000 // crystal frequency - 4MHz
#define ICLK (XTAL/4) // crystal is divided by four
#define SCALE 16 // prescale by 16 - check for overflow!
#define PRELOAD PERIOD*ICLK/SCALE/1000000
```

```
/*******
```

```
unsigned int volt,curr,curr1,TEST_1;
unsigned int v_100,v_10,v_1,v_temp;
unsigned int c_100,c_10,c_1,c_temp;
unsigned int f_100,f_10,f_1,f_temp;
unsigned int p_10000,p_1000,p_100,p_10,p_1,p_temp;
unsigned int power=0,pf,pf1,test,a,test1=0;
unsigned int a=1,temp_12;
```

```
main()
```

```
{
    int a;
    port_init();
    lcd_init();
    ser_init();
```

```

RC0 =0;
OPTION = 0xc3;           // prescale by 16
T0CS = 0;               // select internal clock
TMR0 = 0x00;           // preload timer
T0IE = 0;               // enable timer interrupt
GIE = 1;                // enable global interrupts

```

```

while(1)
{
if (T0IF == 0)
{

}
if (T0IF == 1)
{
    test= test+1;
    T0IF = 0;
    GIE=1;
}
if (test == 255)
{
    getin();
    test=0;
    test1 = test1+1;

    if (RCIF == 1)
    {
        int b;

        DelayMs(100);

        DelayMs(100);
        RCIF = 0;
        b = RCREG;
        RCREG = 0;

        if (b==1)
        {
            RC0 = 1;

        }
    }
}
}

```

```

                if (b == 2)
                {
                    RC0 = 0;
                }
            }
        if(test1 == 30)
        {

            send_dat();
            DelayMs(100);
            test1= 0;
        }

        }
    }
}

```

```

send_dat()
{
    unsigned int a3=0;
    int  a12,a11,a13;
    a3= temp_12;
    a11 = a3/255;
    a12 = a3%255;
    a13 =255;
    TXREG = a13;
    DelayMs(100);

    TXREG = a11;
    DelayMs(100);
    TXREG = a12;
    DelayMs(100);
}

```

```

getin()
{

```

```

ADC_volt init();

```

```
DelayMs(100);  
ADGO =0;  
ADIF =0;  
HTB_1();  
Disp_volt();
```

```
ADC_curr_init();  
DelayMs(100);  
ADGO =0;  
ADIF =0;  
HTB_2();  
Disp_curr();
```

```
ADC_pf_init();  
DelayMs(100);  
ADGO =0;  
ADIF =0;  
HTB_4();  
Disp_pf();  
DelayMs(100);  
Cal_pwr();  
HTB_3();  
Disp_pwr();
```

```
}
```

```
ser_init()
```

```
{
```

```
RP0=1;  
SPBRG = 0x33;  
BRGH = 0;  
SYNC = 0;  
RP0=0;  
SPEN = 1;  
CREN = 1;  
TX9 = 1;  
RP0=1;  
TXEN = 1;
```

```
}
```

```
send_data()
```

```
{
```

```
    TXREG= 1;
```

```
    DelayMs(100);
```

```
    TXREG= volt;
```

```
    DelayMs(100);
```

```
    TXREG= curr ;
```

```
    DelayMs(100);
```

```
    TXREG= pf;
```

```
    DelayMs(100);
```

```
}
```

```
HTB_1()
```

```
// Voltage value from Hex to BCD
```

```
{
```

```
    v_100 = volt/100;
```

```
    v_temp = volt%100;
```

```
    v_10  = v_temp/10;
```

```
    v_1   = v_temp%10;
```

```
}
```

```
HTB_2()
```

```
// Current value from Hex to BCD
```

```
{
```

```
    curr1 = curr;
```

```
    c_100 = curr/100;
```

```
    c_temp = curr%100;
```

```
    c_10  = c_temp/10;
```

```
    c_1   = c_temp%10;
```

```
}
```

```
HTB_3()
```

```
// Power value from Hex to BCD
```

```
{
```

```
    p_10000 = power/10000;
```

```
    p_temp  = power%10000;
```

```
    p_1000  = p_temp/1000;
```

```
    p_temp  = p_temp%1000;
```

```
    p_100   = p_temp/100;
```

```
    p temp  = p temp%100;
```

```
p_10    = p_temp/10;
p_1     = p_temp%10;
```

```
}
```

```
HTB_4() // Power Factor value from Hex to BCD
```

```
{
    pf= pf*2;
    f_100 = pf/100;
    f_temp = pf%100;
    f_10   = f_temp/10;
    f_1    = f_temp%10;
}
```

```
Cal_pwr() // Calculation of Energy
```

```
{
    power    = power + (volt * curr1 * pf)/(3600);
    temp_12 = power;
}
```

```
port_init()
```

```
{
    RP0=1;
    RP1=0;
    TRISD=0;
    TRISB=0;
    TRISA =0xff;
    TRISC0 =0;
    TRISC7 =1;
    TRISC6 =0;
    RP0=0;
    RP1=0;
    PORTD =0;
    RC0  =0;
    RB5  =1;
}
```

```
lcd_init()
```

```
{
    PORTD = 0x38;
```

```

rd_clk();

PORTD = 0x38;
rd_clk();

PORTD = 0x38;
rd_clk();

PORTD = 0x08;
rd_clk();

PORTD = 0x01;
rd_clk();
DelayMs(10);

PORTD = 0x0c;
rd_clk();

PORTD = 0x0c;
rd_clk();

PORTD = 0x0c;
rd_clk();
}

```

```

disp_mesg()
{
    PORTD = 0xce;
    rd_clk();

    PORTD = 'P';
    wr_clk();
}

```

```

Disp_pwr()
{
    /* Displays Multiplied values */
}

```

```
PORTD = 0x86;
rd_clk();

PORTD = 'E';
wr_clk();

PORTD = '!';
wr_clk();

PORTD = 0x00+0x30;
wr_clk();

PORTD = 0x00+0x30;
wr_clk();

PORTD = '!';
wr_clk();

PORTD = p_10000+0x30;
wr_clk();

PORTD = p_1000+0x30;
wr_clk();

PORTD = p_100+0x30;
wr_clk();

PORTD = p_10+0x30;
wr_clk();

PORTD = p_1+0x30;
wr_clk();
```

```
}
```

```
Disp_volt()
```

```
{
```

```
/* Displays Voltage values */
```



```
PORTD = 0x80;
rd_clk();

PORTD    = 'V';
wr_clk();

PORTD    = '!';
wr_clk();

PORTD    = v_100+0x30;
wr_clk();

PORTD    = v_10+0x30;
wr_clk();

PORTD    = v_1+0x30;
wr_clk();
```

```
}
```

```
Disp_pf()
```

```
{
```

```
/*    Displays Power Factor values    */
```

```
PORTD = 0xc8;
rd_clk();
```

```
PORTD    = 'P';
wr_clk();
```

```
PORTD    = 'F';
wr_clk();
```

```
PORTD    = f_100+0x30;
wr_clk();
```

```
PORTD    = '!';
wr_clk();
```

```

    PORTD    = f_10+0x30;
    wr_clk();

    PORTD    = f_1+0x30;
    wr_clk();

}

```

```

Disp_curr()
{
/*    Displays Current values    */

    PORTD = 0xc0;
    rd_clk();

    PORTD    = 'C';
    wr_clk();

    PORTD    = '-';
    wr_clk();

    PORTD    = c_100+0x30;
    wr_clk();

    PORTD    = '!';
    wr_clk();

    PORTD    = c_10+0x30;
    wr_clk();

    PORTD    = c_1+0x30;
    wr_clk();

}

```

```

rd_clk()
{
    DIS:

```

```
    PORTD = 0x04;
    DIEN;

    DelayMs(10);
    PORTD = 0x00;
    DIEN;
}
```

```
wr_clk()
{
    DIS;
    PORTD = 0x05;
    DIEN;
    DelayMs(10);
    PORTD = 0x01;
    DIEN;
}
```

```
ADC_volt_init()
{
    RP0 =1;
    OPTION =0x80;
    ADCON1=0x00;
    ADCON0=0x01; //Channel 0-0 ,
    DelayMs(1);
```

```
//ADC Start
    ADGO=0x01;
    DelayMs(50);
    volt = ADRESH;

}
```

```
ADC_pf_init()
{
    RP0 =1;
```

```
ADCON1=0x00;  
ADCON0=0x09; //Channel 0-0  
DelayMs(1);
```

```
//ADC Start
```

```
ADGO=0x01;  
DelayMs(50);  
pf = ADRESH;
```

```
}
```

```
ADC_curr_init()
```

```
{
```

```
ADCON1=0x00; // 7 channel Analog  
ADCON0=0x19; // Channel 0-0  
DelayMs(1);  
ADGO=0x01;  
DelayMs(50);  
curr = ADRESH;
```

```
}
```

## 5.2 C PROGRAM

*/\* This Program is written in C++. It obtains the values from the Microcontroller and displays them on the screen \*/*

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<process.h>
```

```
void main()
```

```
{
    int a, i=0,ah;
    char key;
    int t=0;
    unsigned int x,y,z;
    float ene,ta;
    clrscr();
```

```
start:
```

```
    a=system("mode com1,1200,n,8,1");        //used to set the baud rate
    gotoxy(35,6);
    printf(" Enter 1-Transmit 2-Receive");
    key=getche();
    if(key=='1')
        {
            goto tra;
        }
    else if(key=='2')
        {
            goto rec;
        }
    else
        {
            exit(0);
        }
```

```
//Transmission of Data
```

```
tra:
```

```
    gotoxy(35,8);  
    scanf("%d",&t);  
    outportb(0x3f8,t);  
    if(t!=0)  
        goto tra;
```

```
//Data receiving loop
```

```
rec:
```

```
    gotoxy(35,10);  
    printf(" Receive mode ");
```

```
    while(!kbhit())
```

```
    {
```

```
xx:
```

```
    z=0x0;  
    _AH=0x03;  
    _DX=0x00;  
    geninterrupt(0x14);  
    ah=_AH;  
    if( (ah & 0x01) == 0x01)
```

```
    {
```

```
        z=inport(0x3f8);  
        gotoxy(35,6);  
        if(z==255)  
            goto aa;
```

```
    }
```

```
    goto xx;
```

```
aa:
```

```
    i=0;  
    _AH=0x03;  
    _DX=0x00;  
    geninterrupt(0x14);  
    ah=_AH;  
    if( (ah & 0x01) == 0x01)
```

```
    {
```

```
        x=inportb(0x3f8);  
        i++;
```

```

    }
    if(i==0)
        goto aa;
    gotoxy(35,8);
bb:
    _AH=0x03;
    _DX=0x00;
    geninterrupt(0x14);
    ah=_AH;
    if( (ah & 0x01) == 0x01)
    {
        y=inportb(0x3f8);
        i++;
    }
    if(i==1)
        goto bb;
    i=0;
    gotoxy(20,12);
    ene=(x*255+y)*0.00001;
    printf("Energy reading per two minutes is %6.5f\n",ene);
    ta=ene*2;
    printf("\t\t Tariff for the Energy Consumed is %6.5f",ta);
    }
    goto start;
}

```

## **CHAPTER VI**

### **CONCLUSION**

The Microcontroller based Automated Energy Meter has been designed fabricated and tested and its operation is found to be satisfactory. In practice, the energy meters present are mostly analog devices and the readings obtained from them are less precise and accurate. With the adaptation of Microcontroller to obtain the Digital reading, the readings obtained are more precise and accurate than their analog counter parts.

The obtained Energy readings are also transmitted from the Microcontroller to the Personal Computer and henceforth, Energy consumed at a distant location is readily available at our desktop. This will pave way for a more sophisticated and simple way of Energy measurement in the near future.

The advantages of this device over Analog system is that,

1. Errors in human readings can be avoided.
2. Energy recorded is updated and periodically sent to the Personal Computer.



## **FURTHER DEVELOPMENTS**

The following developments can be made on this product

1. **Wireless Data transmission from the microcontroller to the Personal Computer**
2. **Using the Network Protocols and the Internet, the Energy Readings can be sent to any part of the World**

---

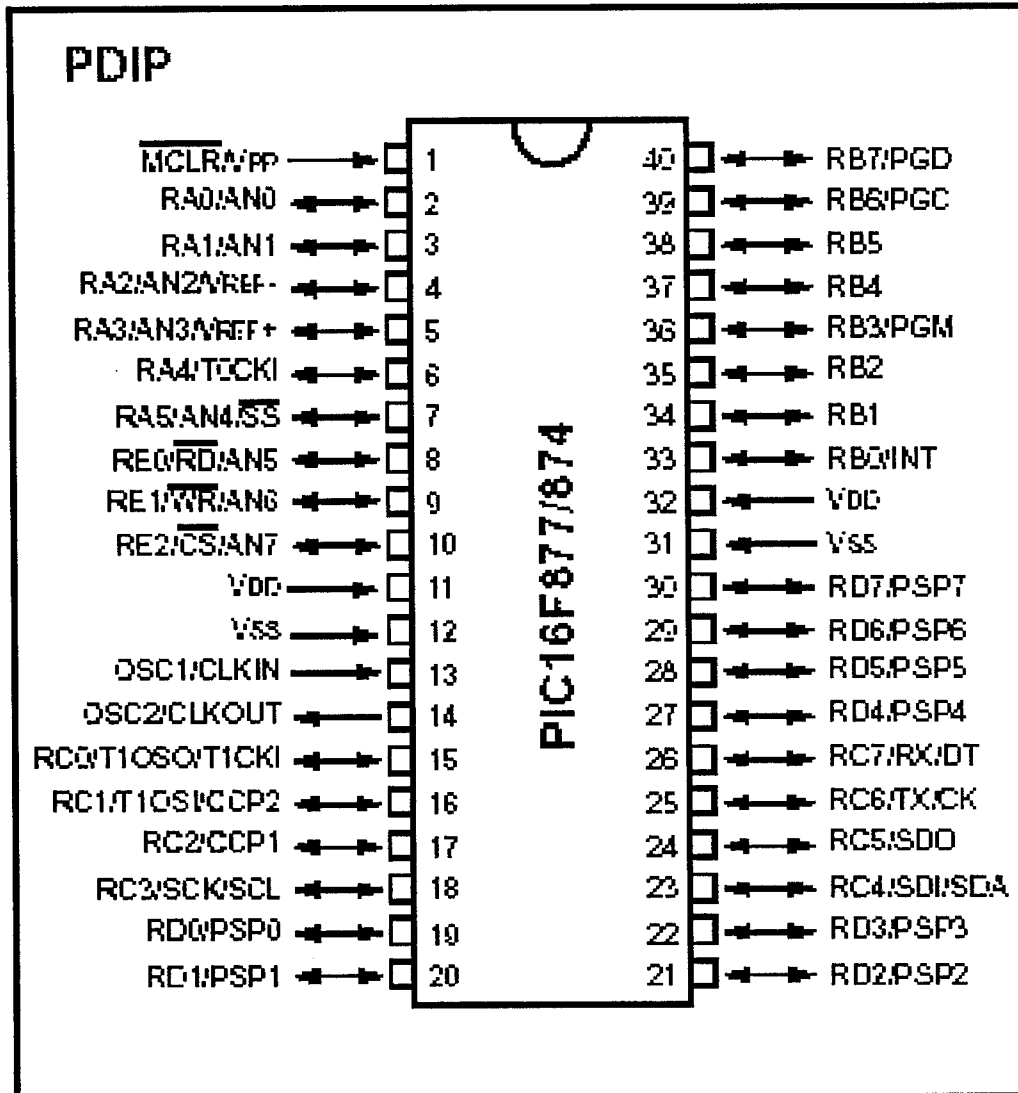
# **BIBLIOGRAPHY**

## **BIBLIOGRAPHY**

1. “Microchip Technical Library CD-ROM”, First Edition, 2001
2. Jan Axelson, “Serial Port Complete-Programming and circuits for RS232 and RS485 Links and Networks”, First Edition, Penram International Publishing, Mumbai, India. ISBN : 81-87972-03-3
3. Ronald J. Tocci, “Digital Systems-Principles and Applications”, Sixth Edition, Prentice-Hall, N.J.,USA. ISBN : 81-203-0985-5
4. Yahavant Kanetkar, “Let Us C”, Third Edition, BPB publications, New Delhi, India. ISBN : 81-7656-040-5

## APPENDIX

### PIC 16F877 PIN DIAGRAM



## 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 <sup>1,4)</sup>	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.
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.
RA0/AN0	2	3	19	I/O	TTL	PORTA is a bi-directional I/O port. 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/SS/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
RB0/INT	33	36	8	I/O	TTL/ST <sup>1)</sup>	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. 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 <sup>2)</sup>	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST <sup>2)</sup>	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.

## PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	PORTC is a bi-directional I/O port. 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 <sup>2</sup> C 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 <sup>2</sup> C 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.
RD0/PSP0	19	21	38	I/O	ST/TTL <sup>1,3</sup>	PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD1/PSP1	20	22	39	I/O	ST/TTL <sup>1,3</sup>	
RD2/PSP2	21	23	40	I/O	ST/TTL <sup>1,3</sup>	
RD3/PSP3	22	24	41	I/O	ST/TTL <sup>1,3</sup>	
RD4/PSP4	27	30	2	I/O	ST/TTL <sup>1,3</sup>	
RD5/PSP5	28	31	3	I/O	ST/TTL <sup>1,3</sup>	
RD6/PSP6	29	32	4	I/O	ST/TTL <sup>1,3</sup>	
RD7/PSP7	30	33	5	I/O	ST/TTL <sup>1,3</sup>	
RE0/ $\overline{RD}$ /AN5	8	9	25	I/O	ST/TTL <sup>1,3</sup>	PORTE is a bi-directional I/O port. RE0 can also be read control for the parallel slave port, or analog input5.
RE1/ $\overline{WR}$ /AN6	9	10	26	I/O	ST/TTL <sup>1,3</sup>	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/ $\overline{CS}$ /AN7	10	11	27	I/O	ST/TTL <sup>1,3</sup>	RE2 can also be select control for the parallel slave port, or analog input7.
V <sub>SS</sub>	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
V <sub>DD</sub>	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.

# PIC 16F877 REGISTER FILE MAP

File Address		File Address		File Address		File Address	
Indirect addr. <sup>(1)</sup>	00h	Indirect addr. <sup>(1)</sup>	80h	Indirect addr. <sup>(1)</sup>	100h	Indirect addr. <sup>(1)</sup>	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 <sup>(1)</sup>	08h	TRISD <sup>(1)</sup>	88h		108h		188h
PORTE <sup>(1)</sup>	09h	TRISE <sup>(1)</sup>	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 <sup>(2)</sup>	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved <sup>(2)</sup>	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h	General Purpose Register 16 Bytes	116h	General Purpose Register 16 Bytes	196h
CCP1CON	17h		97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch		9Ch		11Ch		19Ch
CCP2CON	1Dh		9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
	20h		A0h		120h		1A0h
General Purpose Register 96 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes	
			Eh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h-7Fh	1F0h
			FFh		17Fh		1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

Unimplemented data memory locations, read as '0'.

\* Not a physical register.

Note 1: These registers are not implemented on the PIC 16F876.

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