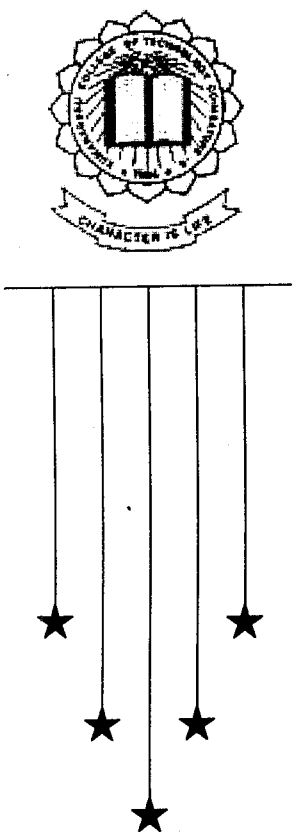
HARMONICS MEASURING METER

PROJECT REPORT

P- 1403



2002-2003

SUBMITTED BY

DEEPA MAHESWARI.V.

RIYAZ HUSSAIN .M.G.

VIDHYA.D.

GUIDED BY

Prof. S. GOVINDARAJU, M.E

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING IN

ELECTRONICS & COMMUNICATION ENGINEERING

OF THE BHARATHIAR UNIVERSITY,COIMBATORE.

Department of Electronics & Communication Engineering

Kumaraguru College of Technology

Coimbatore-641006.

Kumaraguru College of Technology

Coimbatore-641 006

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that this project entitled

HARMONICS MEASURING METER

has been submitted by

Ms.DEEPA MAHESWARI.V
Mr.RIYAZ HUSSAIN M.G.
Ms.VIDHYA.D

in partial fulfillment of the requirements for the award of degree of
Bachelor of Engineering in **Electronics & Communication branch** of **BHARATHIAR UNIVERSITY,COIMBATORE.**
during the academic year 2002-2003.

GUIDE

(Head of the Department)

certified that the candidate was examined by us in the Project Work.

Viva-Voce held on _____

University Register Number _____

(Internal Examiner)

(External Examiner)

Phone : 91-422-626020, 628351
Fax : 91-422-628508
E-mail : indusel@vsnl.com

INDUS ELECTRONICS INDUSTRY

PLOT No.3, ELECTRONICS INDUSTRIAL ESTATE, NEAR "SITRA", COIMBATORE - 641 014. INDIA

CERTIFICATE

This is to certify that the following B.E. (Branch: Electronics & Communication Engineering) students of KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE had undertaken their project "HARMONICS MEASURING METER" from April 2002 to February 2003 at our Industry and have successfully completed it.

Ms. V. DEEPA MAHESWARI

Mr. M. G. RIYAZ HUSSAIN

Ms. D. VIDHYA

Their performance during that period was found to be good.

We wish them all success.

For INDUS ELECTRONICS INDUSTRY

PLACE: COIMBATORE
DATE : MARCH 13, 2003.

(R. JAGADEESAN)
MANAGING PARTNER.

ACKNOWLEDGEMENT

With an informative, knowledge gaining and industrial exposure, our project guided us to contemplate on the field of harmonic analysis. Only a whole-hearted and perceptive guidance made this project a successful outcome for us.

At the outset, we express our sincere and profound sense of gratitude to our principal **Dr.K.K.Padmanaban** B.S.C.(Engg), M-Tech, Ph.D., for his kind patronage.

We profusely thank **Professor Muthuraman Ramaswamy,** M.E., MIE.,FIETE., MIEEE(USA)., MISTE., MBMESI., C.ENG(I), Head of the Department, Electronics & communication Engineering for his invaluable and gentle reminders that encouraged us reach our goal.

We are all the more grateful to **Mr.R.Jagadeesan**, M.E, Managing Partner of Indus Electronics Industry ,Coimbatore, for giving us the wonderful opportunity to undertake this project on behalf of his industry.

Though words are not enough, it is all that we have got to express our deepest gratitude to our guide **Mrs.Vasanthamani**, Manager, R&D, Indus Electronics Industry, coimbatore, for her valuable guidance and cooperation.

We are greatly indebted to **Prof. S.Govindaraju,** M.E., Assistant Professor, Department of Electronics and Communication Engineering, for his valuable guidance and constant encouragement without which our project would not have been a reality.

We are also greatly indebted to all our teachers who have given their valuable suggestions and constant encouragement during the course of our project work.

SYNOPSIS

Harmonics on power systems has created heavy loss, especially in the case of electrical motors. Our project is concerned about indicating the presence of these harmonics in the power line.

We use the concept of Fast Fourier Transform (FFT) to calculate the harmonics present therein. The 230V AC signal from the power line is stepped down to less than 5V. The analog input is converted to its corresponding digital values using A to D converter thus samples of the input signal is obtained. The data obtained is fed as an input to FFT program. The harmonics along with the fundamental frequency is calculated as the output. The Total Harmonic Distortion is calculated. The fundamental frequency, 3^{rd}, $5^{th,}$ 7^{th} harmonics and Total Harmonic Distortion are displayed on an LED array.

CONTENTS

CHAPTER 1

INTRODUCTION

INTRODUCTION

AIM OF THE PROJECT:

The system deals with the design and construction of a Harmonic Measuring Meter to measure the harmonic in the power line and also to find out the Total Harmonic Distortion. In general, the system can be used to calculate the D.C component, Fundamental frequency component, 3^{rd}, 5^{th} and 7^{th} harmonics in any power line. The system can be expanded in future for displaying harmonics up to any level and a suitable filter can be designed to remove the harmonics.

CHAPTER 2

MICROCONTROLLER 16F877

ABOUT THE MICROCONTROLLER

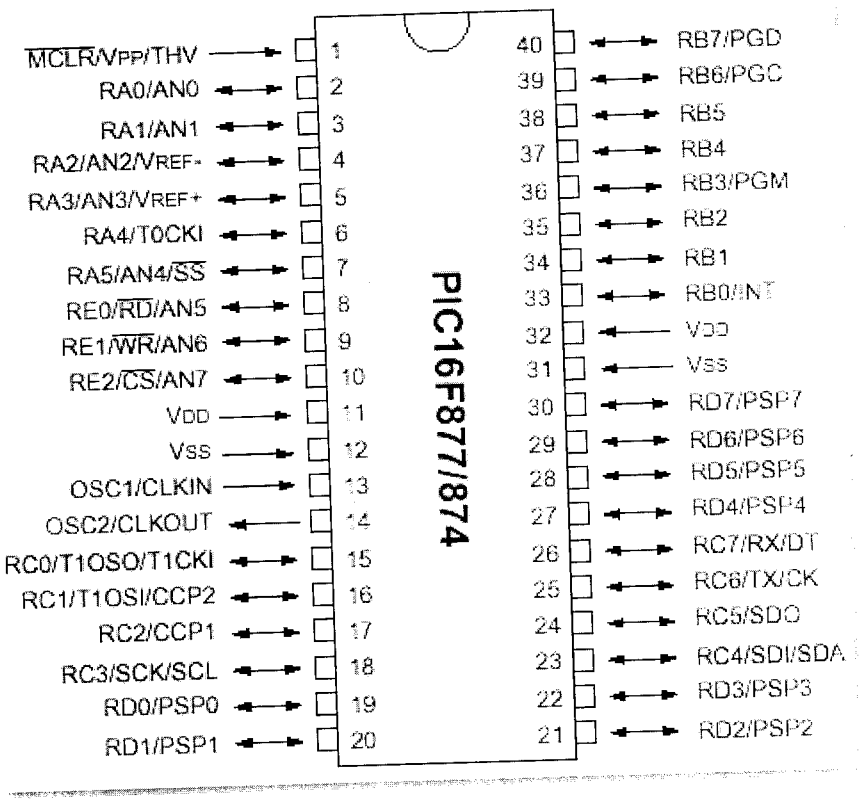
INTRODUCTION:

The microcontroller has inbuilt data memory (RAM) ,optional program memory (ROM or EPROM or EEPROM) and I/O ports to communicate with external environment. Microcontrollers are mostly preferred for standalone and real time applications.

FEATURES:

- High performance RISC CPU
- Pin compatible to PIC16C73/74/76/77
- Power-on-reset, Brown-out Reset
- Power-up-Timer, Watchdog timer
- Low power, high speed operation
- High sink/source current: 25mA

PIN DIAGRAM :

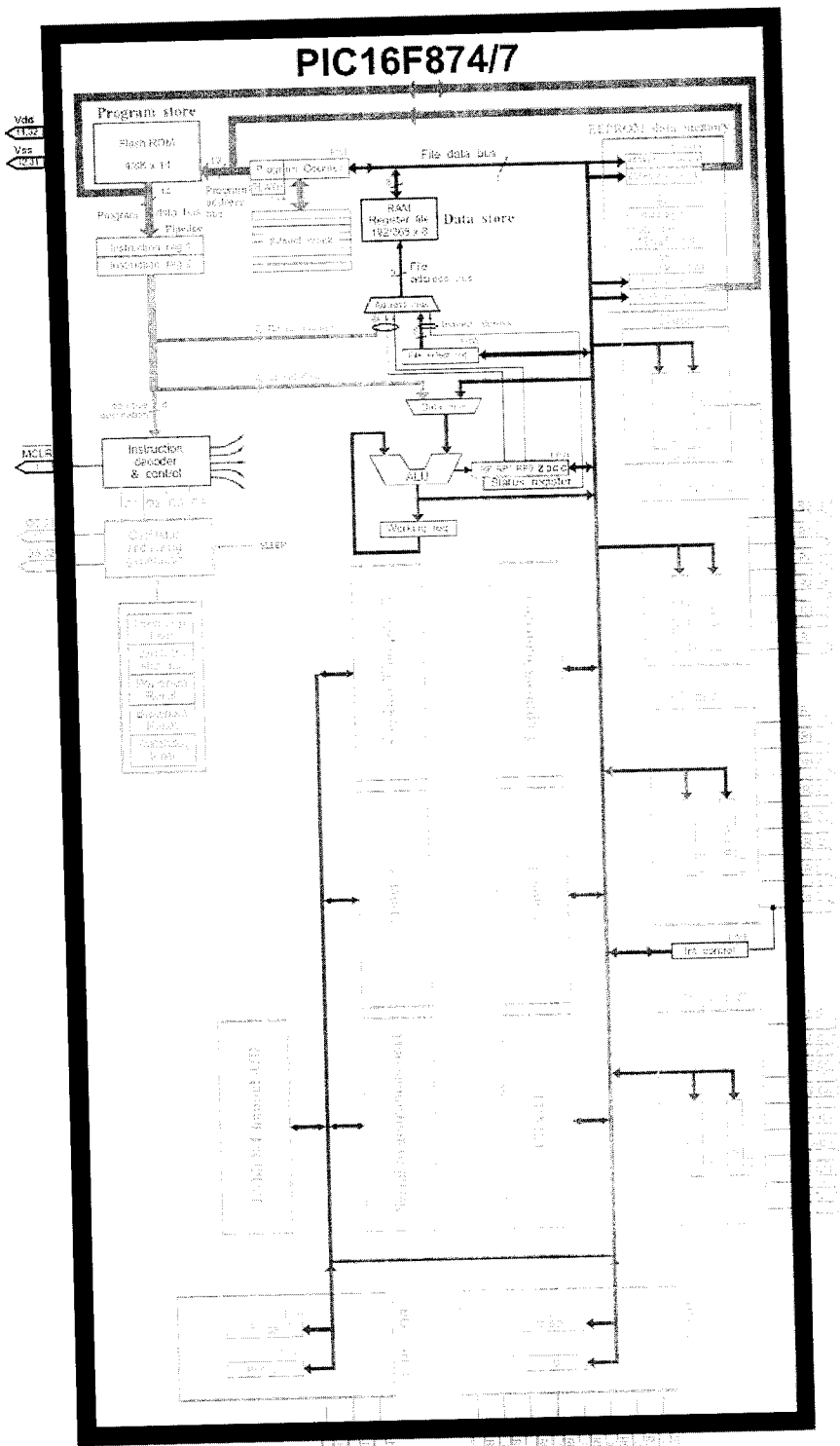

PIC16F877/874
MCLR/VPP/THV 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF- 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS 7
RE0/RD/AN5 8
RE1/WR/AN6 9
RE2/CS/AN7 10
VDD 11
VSS 12
OSC1/CLKIN 13
OSC2/CLKOUT 14
RC0/T1OSO/T1CKI 15
RC1/T1OSI/CCP2 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0 19
RD1/PSP1 20
40 RB7/PGD
39 RB6/PGC
38 RB5
37 RB4
36 RB3/PGM
35 RB2
34 RB1
33 RB0/INT
32 VDD
31 VSS
30 RD7/PSP7
29 RD6/PSP6
28 RD5/PSP5
27 RD4/PSP4
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3
21 RD2/PSP2

ARCHITECTURE OF 16F877:

This section provides brief description of each architecture features. These include

- Harvard architecture
- Long word Instructions
- Single word Instructions
- Single cycle Instructions
- Instruction pipelining
- Reduced Instruction Set
- Register File Architecture
- Orthogonal (symmetric) Instructions

ARCHITECTURE:



CENTRAL PROCESSING UNIT:

The CPU is responsible for fetching the correct instruction, decoding that instruction, and executing it. The CPU controls the program memory address bus, data memory address bus and address to the stack.

ARITHMETIC LOGIC UNIT:

The 8 bit ALU is a general purpose arithmetic and logic unit responsible for performing arithmetic and Boolean functions between the data in working register(W) and any register file. Unless mentioned, arithmetic operations are 2's complement in nature. W register is not an addressable one. The carry, digit carry and Zero bits in the status register are affected.

SPECIAL FUNCTION REGISTERS:

The special function registers are registers used by the CPU and Peripheral Modules for controlling the desired operation of the device. The SFR's can be classified into two sets, those associated with the "core" function and those associated with "peripheral" functions.

PROGRAM COUNTER:

The program counter specifies the address of the instruction to fetch for execution. The PC is 13 bits wide. The low byte is called PCL register which is readable and writable. The high byte called PCH register is not directly readable and writable.

STACK & STACK POINTER:

The stack allows a combination of up to 8 program cells and interrupts to occur .The stack contains the return address from the branch in program location. This is an 8 level deep x 13-bit wide hardware stack .The stack space is not part of either program or data space .The stack pointer is not readable or writable.

MEMORY ORGANISATION

The high performance microcontroller uses several distinct memory areas. These are program memory and data memory. The data memory is partitioned into multiple blocks which contain the General purpose registers and the special function registers.

DATA EEPROM AND FLASH PROGRAM MEMORY

The Data EEPROM and FLASH Program memory are readable and writable during normal operation over the entire range. The data memory is not directly mapped in the register file space. Instead it is indirectly addresses through the special function registers.

MASTER SYNCHRONOUS SERIAL PORT MODULE

The MSSP module is used for a serial interface useful for communicating with other peripheral or microcontroller devices.

INTERRUPTS

There are up to 14 internal/external interrupt sources. There are interrupt registers present In order to enable or disable the interrupts.

TIMER/COUNTER

There are 3 timers/counters in which Timer0 and Timer1 can act both as timer and counter while the Timer2 can act only as a timer and they have various features.Timer1 is a 8bit timer/counter while the timer2 is a 16 bit timer/counter and timer2 is a 8bit timer.

USART

The Universal Synchronous Asynchronous Receiver Transmitter(USART) module is one of the two serial I/O modules. The USART can be configured in the following modes:

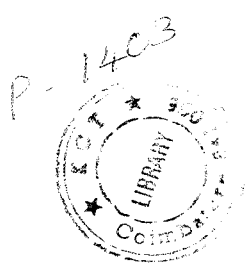
- Asynchronous (full duplex)
- Synchronous - Master (half duplex)
- Synchronous - Slave (half duplex)

ANALOG TO DIGITAL CONVERTER(A/D) MODULE

The A/D module has 8 inputs. The result of the 8 input A/D converter module is 10-bit digital number. The A/D module has 4 registers, where 2 of them are address registers and 2 of them are control registers.

I/O PORTS

The device offers a 6-bit and 8-bit I/O ports. There are 5 ports namely PORTA,PORTB,PORTC,PORTD and PORTE. The corresponding data direction registers are called TRIS registers. Among these ports PORTD can be configured as Parallel Slave port and PORTE controls the slave operation.



CHAPTER 3

HARMONICS

HARMONICS:

Non-linear loads can draw a number of current frequencies. These frequencies are generally multiples of the fundamental frequency (i.e. 50 Hz) and usually decrease in magnitude as their harmonic order increases. Harmonic is a term that describes sinusoidal waveforms that operate at a frequency that is a multiple of the fundamental 50 Hz frequency. When a current, or voltage, operates at other than the fundamental 50 Hz frequency it is said to operate at a specific harmonic order (3rd harmonics operate at 150 Hz; 5^{th} harmonics operate at 250 Hz).

Harmonics are the result of non-linear loads demanding a current waveform different from the shape of the applied voltage waveform.

HARMONIC CURRENTS:

Harmonic currents and voltages are integer multiples of the system's fundamental frequency. For example, with a fundamental frequency of 50Hz, the 3rd harmonic frequency is 150Hz (3 x 50Hz).

Nonlinear loads are the source of harmonic currents. That is, the load's current waveform is non-sinusoidal. As a result, the distorted current waveform is rich in sinusoidal harmonic current waveforms.

HARMONIC VOLTAGES:

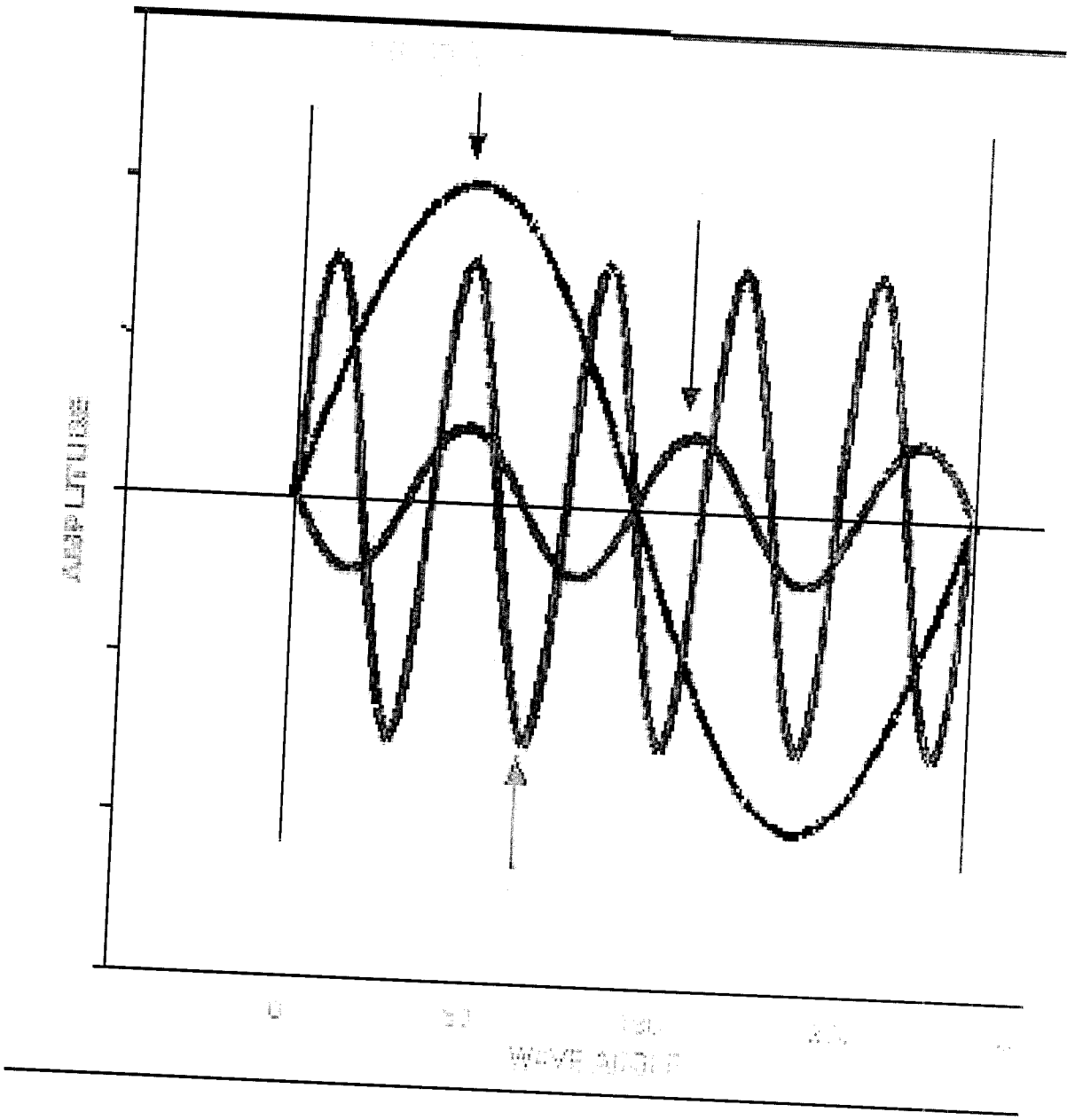
Nonlinear loads include electronic devices such as rectifiers, current controllers, AC and DC drives, cyclo-converters and devices with switch-mode power supplies such as computers monitors, telephone system's harmonic impedances cause the load-generated harmonic currents tc

produce harmonic voltages ($E_H = I_H \times Z_H$). Transformers, and feeder and branch circuit impedances will cause maximum systems, printers, scanners, and electronic lighting ballasts. The electrical distribution Voltages($E_H = I_H \times Z_H$). Transformers, and feeder and branch circuit impedances will cause maximum voltage distortion at the nonlinear loads.

EFFECT OF HARMONIC ON FUNDAMENTAL FREQUENCY:

The fundamental frequency's sinusoidal waveform, which is always predominant, becomes distorted by the addition of harmonic sinusoidal waveforms. The measure of distortion is given as Percent Total Harmonic Distortion of the Fundamental Waveform (%THD [voltage] & %THD [current]).Not all the harmonics will affect the power line, but only a few will cause major disturbance, especially 3^{rd} and 5^{th} harmonics.

FIGURE WITH FUNDAMENTAL FREQUENCY ALONG WITH 3RD AND 5TH HARMONICS:



CAUSES OF HARMONICS:

Harmonics are produced by "non-linear" loads in power systems. A non-linear load does not draw current in proportion with the applied 50 Hz sinusoidal voltage waveform. In other words, this equipment "gulps" electric current from the supply rather than taking current in a smooth fashion.

SOURCES OF HARMONICS:

Harmonics are created by the use of non-linear devices such as UPS systems, solid state variable speed motor drives, rectifiers, welders, arc furnaces, fluorescent ballasts and personal computers. Individual harmonic frequencies will vary in amplitude and phase angle, depending on the harmonic source.

SYMPTOMS:

In small quantities, harmonics are of little concern. However, too many non-linear loads in your facility can lead to significant power quality problems. How do you know if ycu have harmonics in your facility's electrical system? Here are a few symptoms to be aware of:

- Over heated motors and transformers
- Blown fuses or breakers tripping repeatedly
- Failure of power factor correction capacitors
- Timing errors in sensitive electronic equipment

TOTAL HARMONIC DISTORTION :

The ratio of a wave's harmonic content to its fundamental component, expressed as a percentage. Also called "harmonic factor," it is a measure of the extent to which a waveform is distorted by harmonic content. It is usually expressed as current or voltage "total harmonic distortion" or THD Harmonic distortion is found in both the voltage and the current waveform. Most current distortion is generated by electronic loads, also called non-linear loads. These non-linear loads might be single phase loads such as point-of-sale terminals, or three-phase like variable speed drives. As the current distortion is conducted through the normal system wiring, it creates voltage distortion according to Ohm's Law. While current distortion travels only along the power path of the non-linear load,

voltage distortion affects all loads connected to that particular bus or phase.

CURRENT DISTORTION:

Current distortion affects the power system and distribution equipment. It may directly or indirectly cause the destruction of loads or loss of product.

From the direct perspective, current distortion may cause transformers to overheat and fail even though they are not fully loaded. Conductors and conduit systems can also overheat leading to open circuits and downtime.

On three-phase systems, current distortion causes higher than expected currents in shared neutrals. A shared neutral is one that provides the return path for two or three-phases. Currents as high as 200% of the phase conductors have been seen in the field. This large level of current can easily burn up the neutral creating an

open neutral environment. This open neutral forces voltage swells and over voltages to exist. These voltage conditions easily destroy equipment, particularly power supplies. Thus, indirectly, the current distortion damaged the loads. One other indirect problem current distortion introduces is called resonance. Certain current harmonics may excite resonant frequencies in the system. This resonance can cause extremely high harmonic voltages, again possibly damaging equipment.

VOLTAGE DISTORTION:

Voltage distortion, on the other hand, directly affects loads. Distorted voltage can cause motors to overheat and vibrate excessively. It can also cause damage to the shaft of the motor. Even non-linear loads are prey to voltage distortion. Equipment ranging from computers to electronic ballast fluorescent lights may be damaged by voltage distortion.

If damage is occurring due to current distortion, except for high neutral current, then one solution is to reduce the distortion.

CHAPTER 4

FAST FOURIER TRANSFORM

FAST FOURIER TRANSFORM:

The Fast Fourier Transform (FFT) is a method for computing the DFT with reduced number of calculations, this computational efficiency is achieved by divide and conquer approach. FFT algorithms deals with the decomposition of N-point DFT into successively smaller DFT. In FFT, N can be expressed as $N=r^m$, where r is called radix of the FFT algorithm.

In radix-2 FFT the N-point sequence is decimated into 2-point sequence. From the results of 2-point DFTs 4-point DFT can be computed and so on, we will get N-point DFT. Performing radix- 2 FFT ,the value of N should be $N=2^m$,//the total number of complex addition is $N\log_2 N$ and complex multiplication is $(N/2)\log_2 N$.

Decimation in time(DIT) and decimation in frequency (DIF) are the two ways of computations of FFT .In DIT algorithm ,the N-point FFT of a time domain sequence x(n) converts to a frequency domain N-point sequence X(K). In DIF algorithm ,the N-point FFT of a frequency domain sequence X(K) converts to a time domain N-point sequence x(n).

DECIMATION IN TIME - RADIX-2 FFT:

In DIT algorithm, the input given is in bit reversed order and the output got is in normal order. The DIT algorithm is used here because output (frequency) is needed in the natural order.

FORMULA :

$$X(K) = \sum x(n)\ W_N^{kn} \quad , \qquad 0 \leq K \leq N - 1$$

Where $W_N = e^{-j2\Pi/N}$

CHAPTER 5

HARDWARE DESCRIPTION

HARDWARE DESCRIPTION

POWER SUPPLY DESCRIPTION:

PIC POWER SUPPLY:

The 220v AC signal is stepped down to 0-10 v supply. The DC output of 10v is obtained by passing it through a bridge rectifier. The +5v DC is regulated using a regulator IC 7805. AC spikes are removed using 10μF capacitor.

PIC INPUT SUPPLY:

The 230v AC to be tested for harmonics is stepped down to 4v. Since the PC will not accept negative signals the input is clamped to 2.5v using a clamper circuit with IC LM324.

CIRCUIT DESCRIPTION:

The output of the clamper circuit is given as an analog input to the PIC (Pin 2). The memory contents are cleared with a resistor-capacitor setup at pin 1.On key press a short

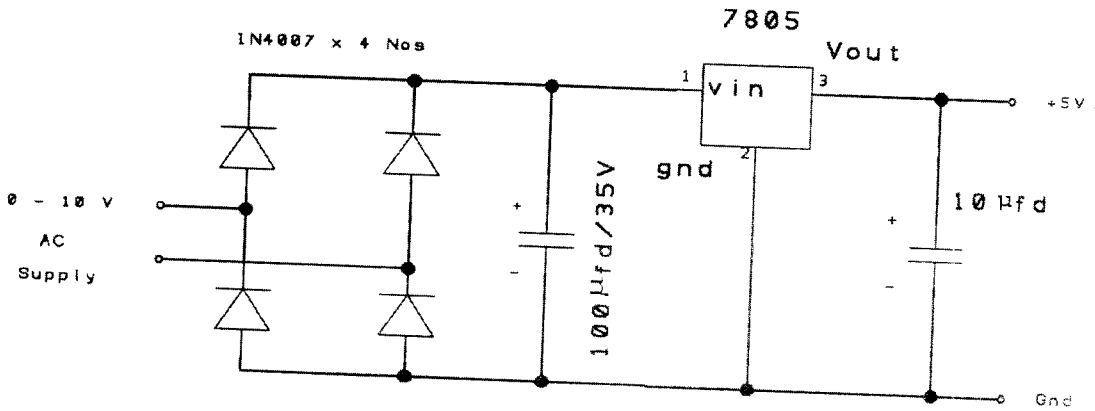
circuit or active low is found at RB7 (Pin 40) to generate an interrupt. A 16 MHz crystal is used as an external source and capacitors are included to remove the unwanted AC spikes. The analog signals are given as input to the PIC (Pin 2) which are sampled and converted into Digital signals using 10 bit A to D converter present within the PIC. This digital data is given as FFT input. The FFT program converts this time domain signals to frequency domain signals obtained in a natural order. Harmonics are calculated from these values. Total Harmonic Distortion is calculated from these harmonic values. As the key is pressed an interrupt is generated which displays the rms value of fundamental frequency. As the key is pressed again the third, fifth, seventh harmonics and Total Harmonic Distortion are displayed. On further key press it goes to the fundamental frequency value. PIC is programmed to do all these functions.

The port D [RD0 - RD3] is used as output port connected to the BCD to Seven Segment decoder (IC 74LS47). The port B [RB1 - RB3] connected to a 3 to 8 Decoder (IC 74HCT138) used as segment drive. The port E [RE0 - RE2] and port B [RB6] are connected to four LED's which are used to indicate fundamental frequency, third, fifth, seventh harmonics and Total Harmonic Distortion.

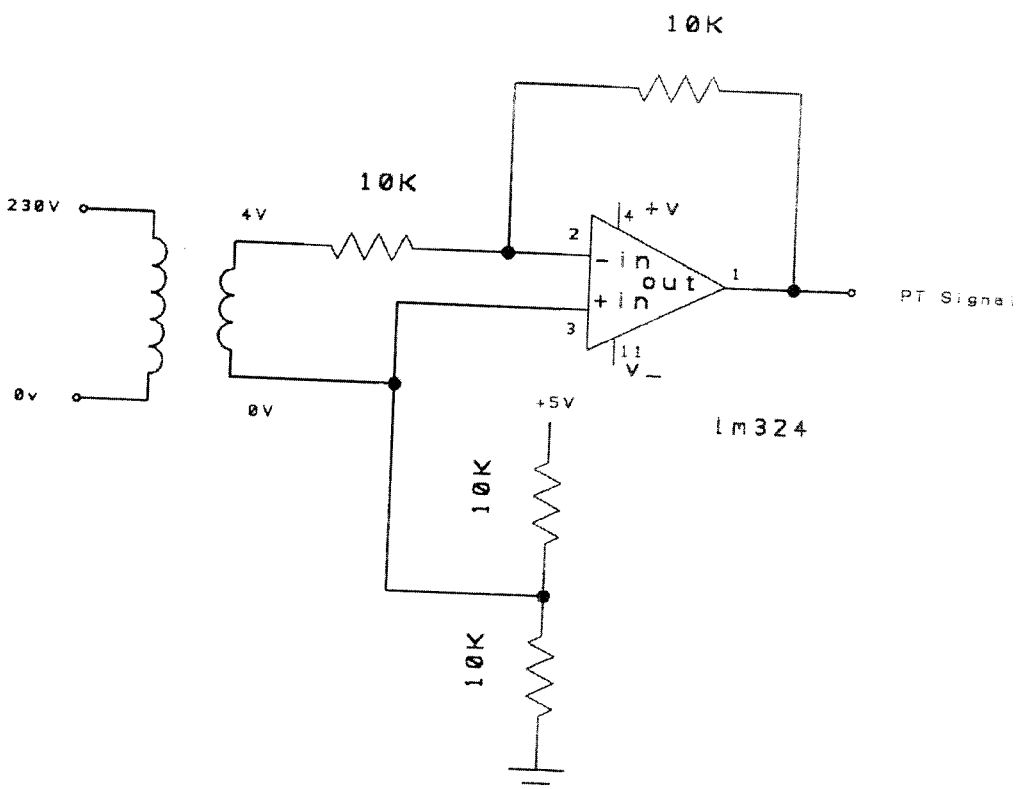
DISPLAY :

The BCD from the PIC is fed to the BCD to seven segment decoder IC (IC 7447) and the digit line segments are operated. The digits are driven using pull-up resistors and IC 2803.The seven segment LED's are connected in a common cathode format. The driver circuit for the seven segment selection is given from the IC 74HCT138 which is a 3 to 8 Decoder.

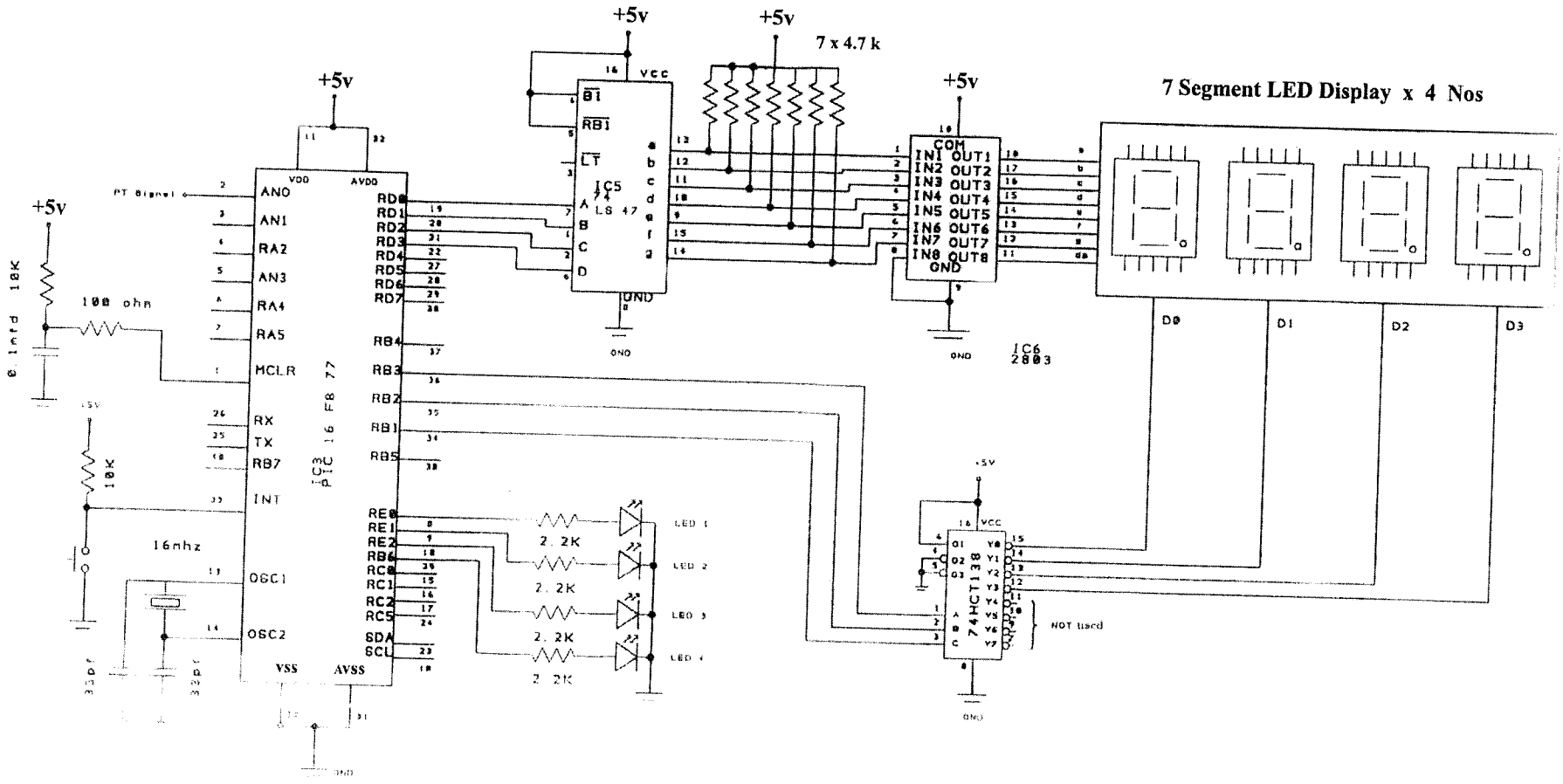
PIC POWER SUPPLY



PIC INPUT



HARMONICS MEASURING METER



CHAPTER 6

SOFTWARE DESCRIPTION

SOFTWARE DESCRIPTION

ALGORITHM :

STEP 1 : Initialise all the Registers and set key data to one.

STEP 2 : Load the TIMER 1 registers with 63bf and set start flag to one and half flag, measurement completed flag to zero.

STEP 3 : Start the Analog to Digital conversion along with TIMER1.

STEP 4 : The digital data is collected in an array.

STEP 5 : When the TIMER 1 reaches FFFF an interrupt is occurred and Interrupt Service Routine is called setting half flag to one.

STEP 6 : The TIMER1 again starts counting from 63bf meanwhile the Analog to Digital conversion takes place.

STEP 7 : When the TIMER1 reaches FFFF an interrupt occurs.

STEP 8 : Interrupt Service Routine is called setting measurement completed flag to one.

STEP 9 : The Analog to Digital operation occurs till measurement completed flag is set.

STEP 10: The Analog to Digital converted result is fed as input for FFT calculation.

STEP 11: From the FFT output ,Harmonics are calculated and store in an array.

STEP 12: The Total harmonic Distortion is calculated from the harmonics value obtained.

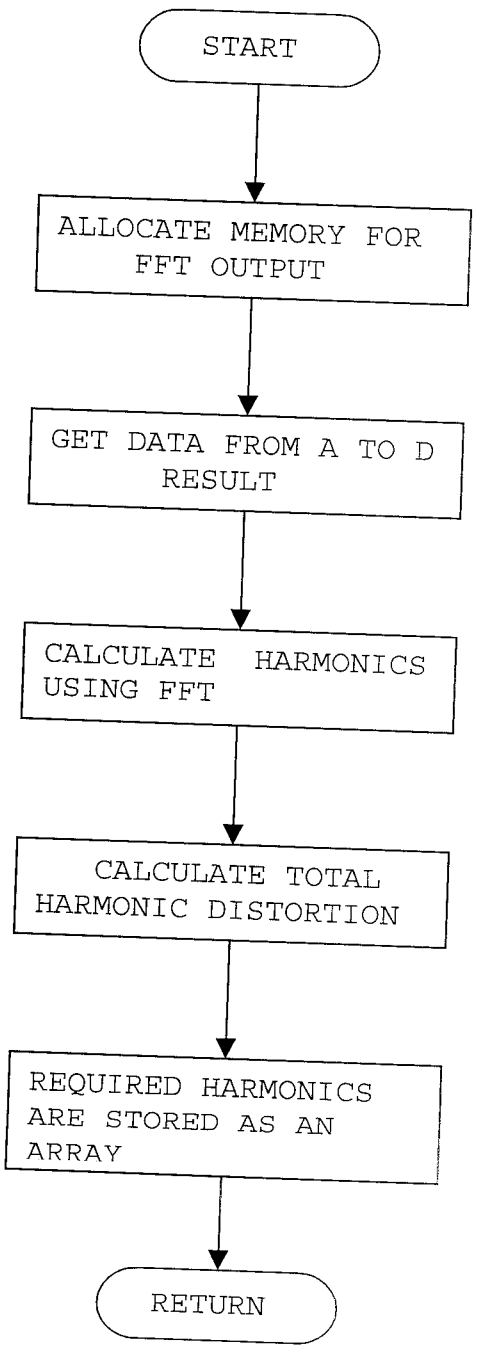
STEP 13: The key press interrupt flag is checked, if this flag is set then the RMS value of the 50Hz signal is displayed.

STEP 14: The second ,third ,fourth key press displays the third harmonic, fifth harmonic seventh harmonic& total harmonic distortion respectively. Further key press is rolled back to fundamental.

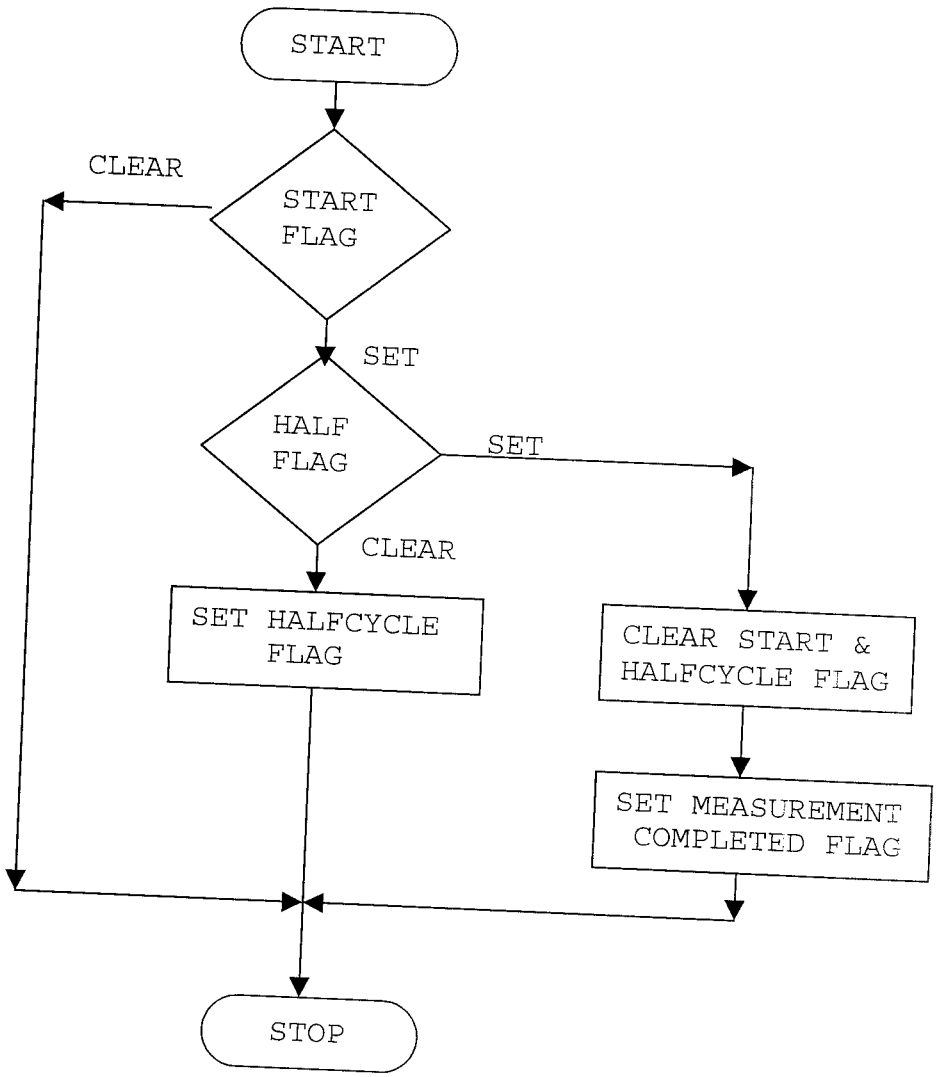
STEP15: The whole process is repeated continuously.

FLOW CHARTS:

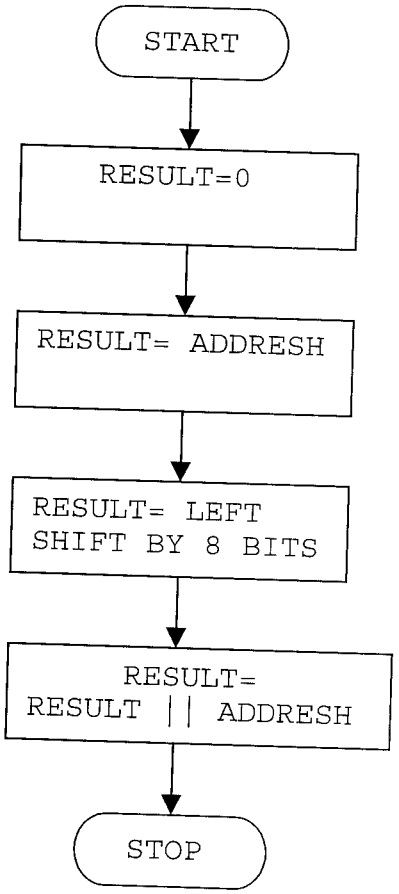
FAST FOURIER TRANSFORM:



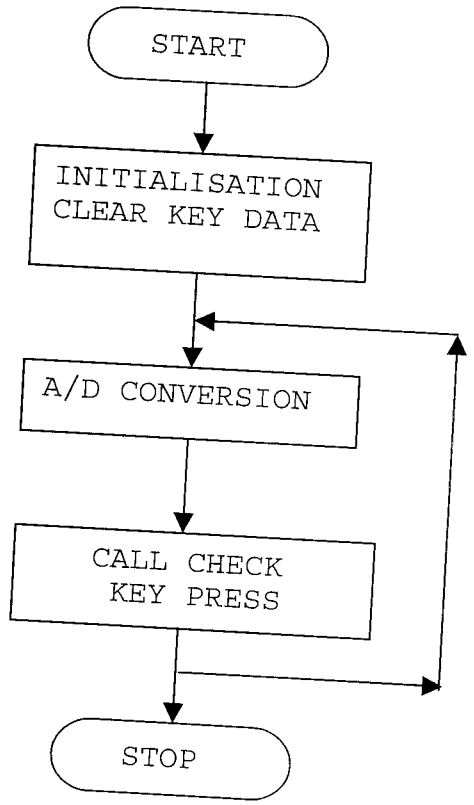
TIMER INTERRUPT SERVICE ROUTINE:



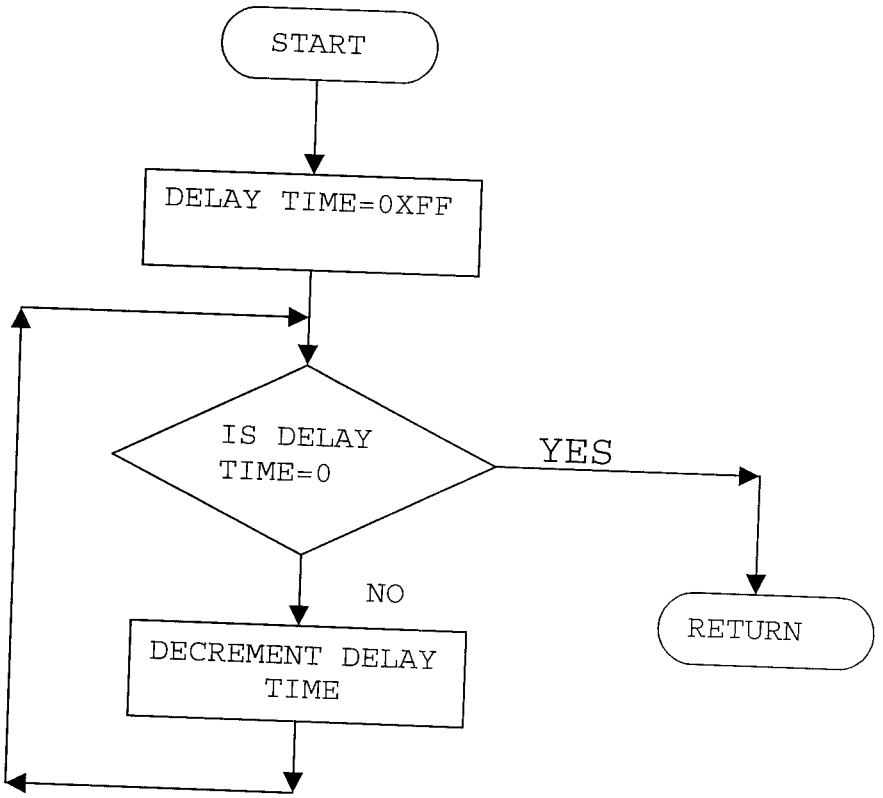
RESULT:


START
RESULT=0
RESULT= ADDRESH
RESULT= LEFT SHIFT BY 8 BITS
RESULT= RESULT || ADDRESH
STOP

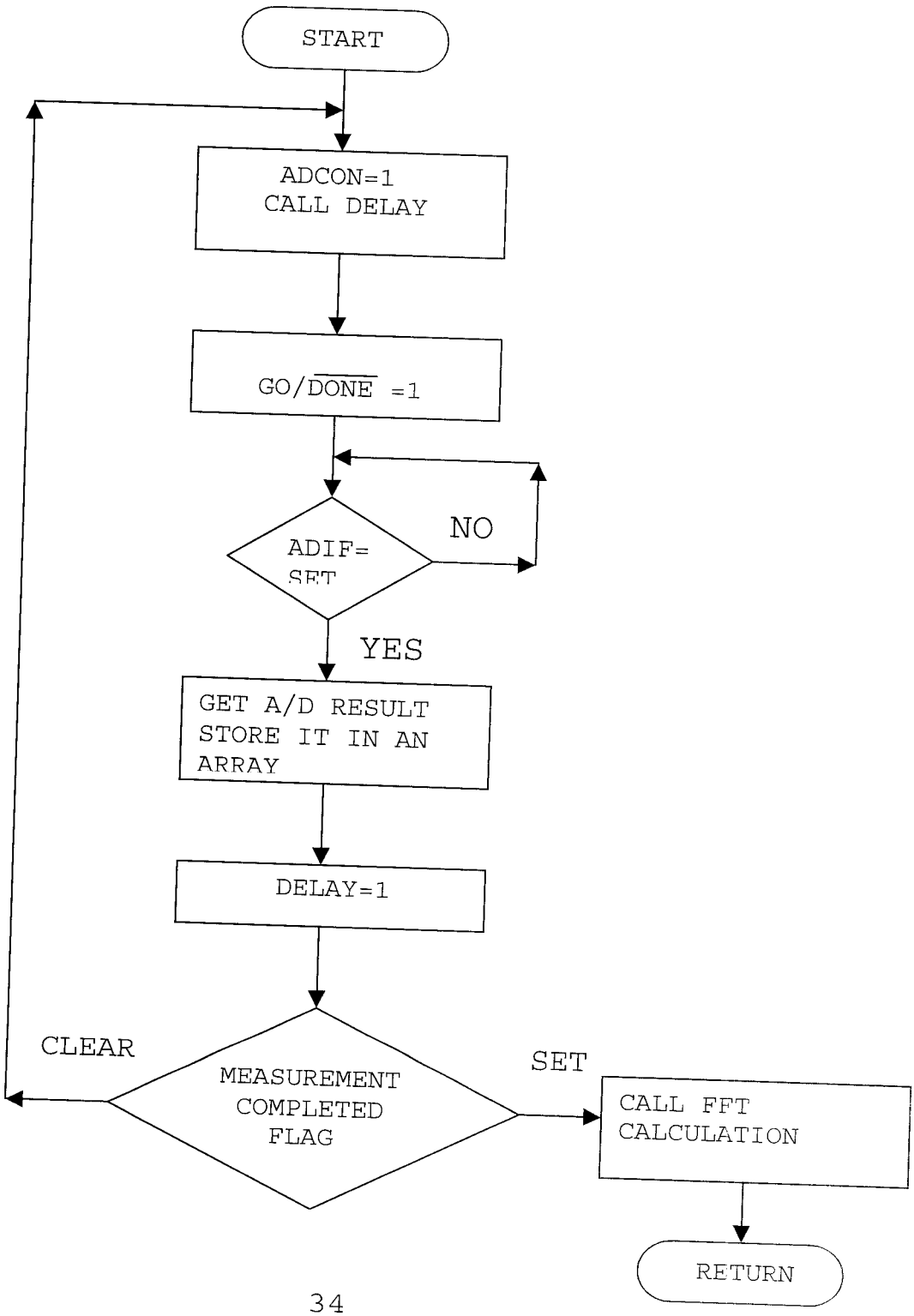
MAIN PROGRAM:



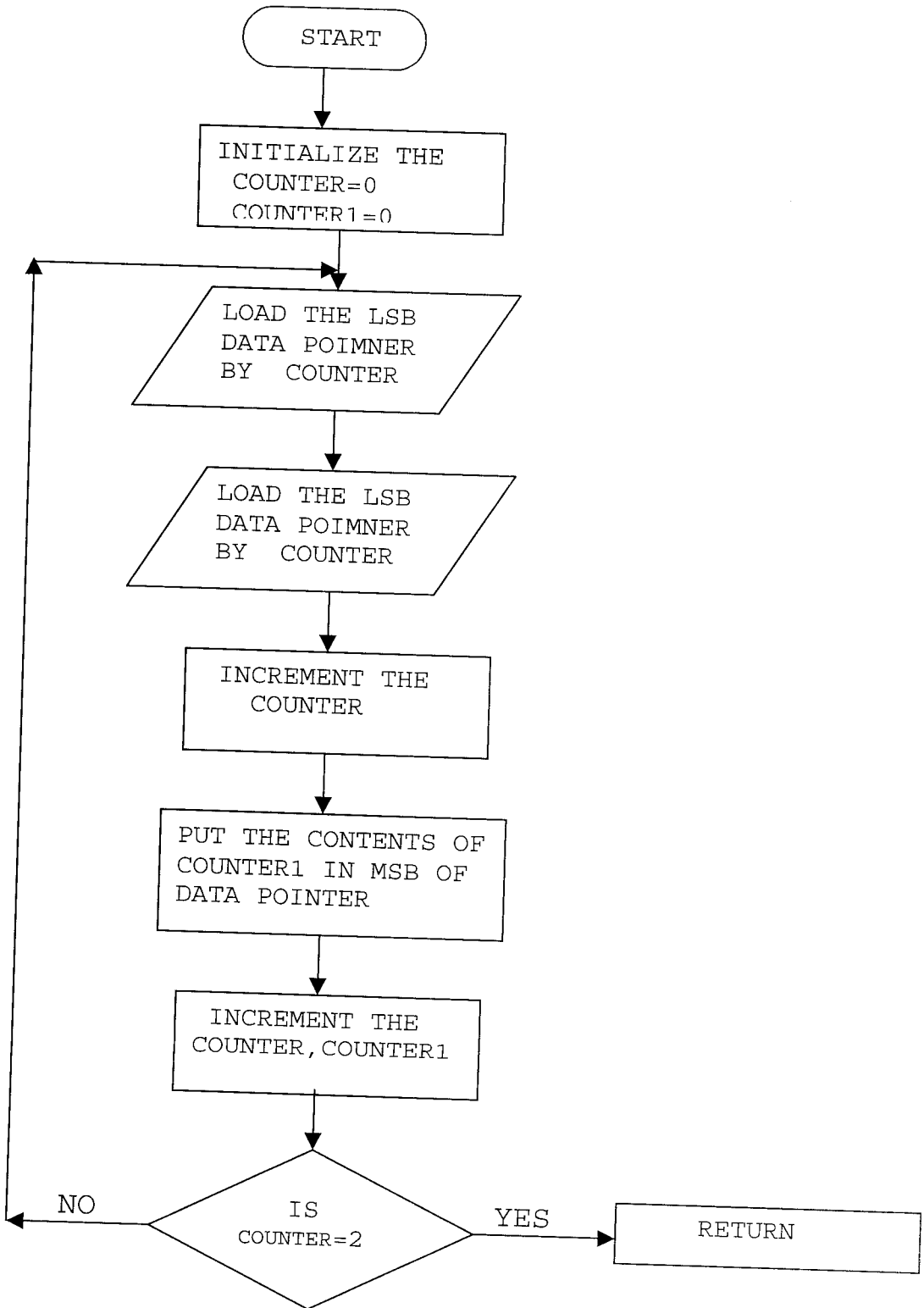
DELAY:


START
DELAY TIME=0XFF
IS DELAY TIME=0
YES
NO
RETURN
DECREMENT DELAY TIME

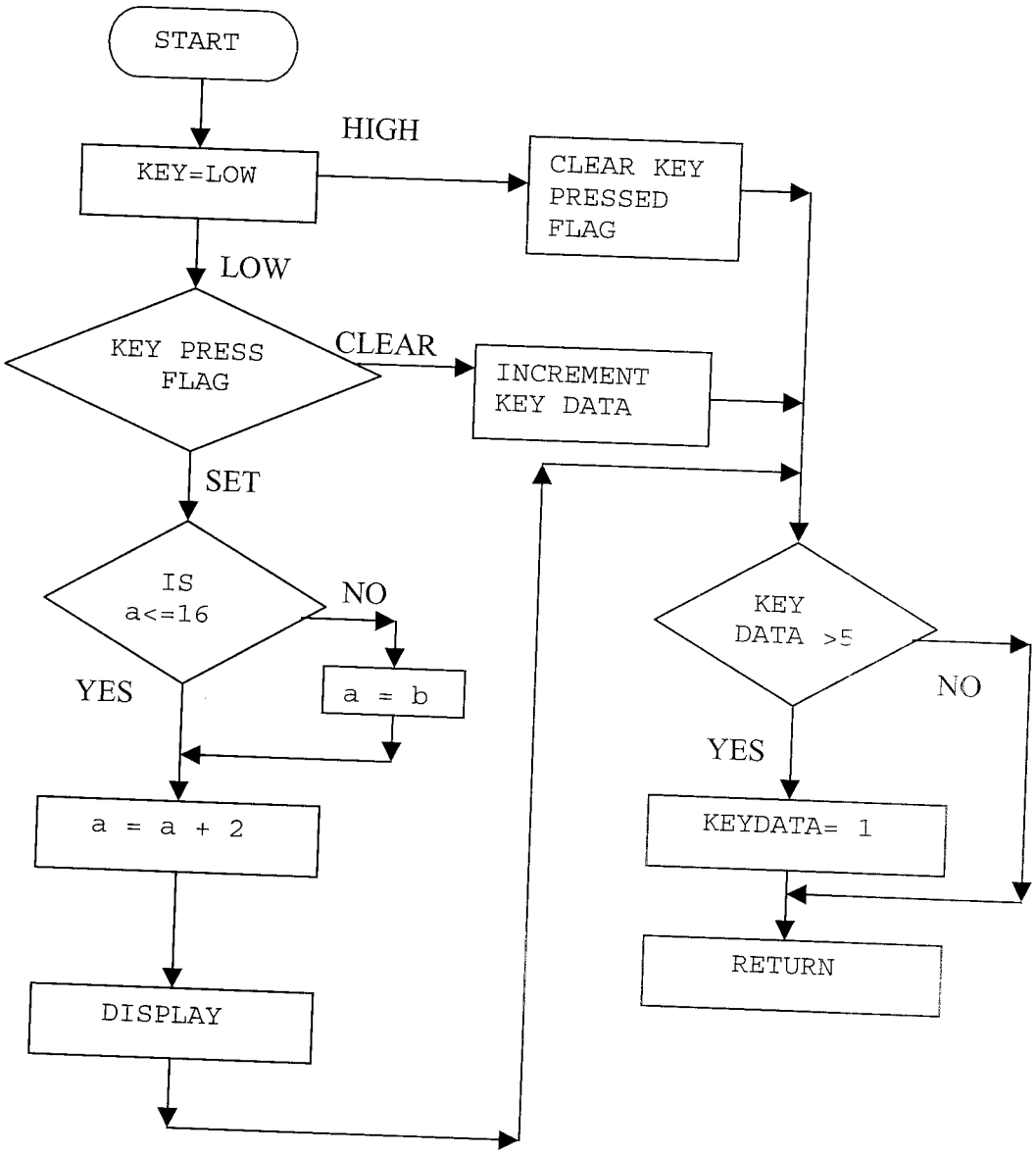
A TO D CONVERSION:



DISPLAY:


START
INITIALIZE THE
COUNTER=0
COUNTER1=0
LOAD THE LSB
DATA POIMNER
BY COUNTER
LOAD THE LSB
DATA POIMNER
BY COUNTER
INCREMENT THE
COUNTER
PUT THE CONTENTS OF
COUNTER1 IN MSB OF
DATA POINTER
INCREMENT THE
COUNTER,COUNTER1
IS
COUNTER=2
NO
YES
RETURN

KEYPRESS:


START
KEY=LOW
HIGH
CLEAR KEY PRESSED FLAG
LOW
KEY PRESS FLAG
CLEAR
INCREMENT KEY DATA
SET
IS a<=16
NO
a = b
YES
a = a + 2
DISPLAY
KEY DATA >5
NO
YES
KEYDATA= 1
RETURN

C PROGRAM:

```
// PROGRAM FOR PIC OPERATION
# include<pic1687x.h>
# include<math.h>

# define PI 3.141592
# define ANGLE(A,B) ((2*A*B*PI)/N)

float rmul(float,float,float,float);
float imul(float,float,float,float);
float rpart,ipart,input,output;

unsigned char keydata;

unsigned int res[80];
unsigned int harm[6];

bit start_flag,
    half_flag,
    measurementcomp_flag;

void initialise();
void A2D();
void keypress();
void delay();
void delay1();
unsigned int A2Dresult();
void FFT();
void display(unsigned int);

main()
{
    initialise();    // calls function initialize
    while(1)
    {
        A2D();       // calls function A2D
        keypress(); // calls function key press
    }}
```

```
void initialise()
{
// Initializes all the registers
     INTCON= 0xC8;
     PIR1= 0x00;
     PIE1= 0x41;
     ADCON0=0x80;
     ADCON1=0x2E;
     TRISA= 0x37;
     TRISB= 0x7C;
     TRISD= 0x70;
     keydata = 1;
}

void A2D()
{
     int i=0;

     TMR1L=0xBF;      //sets timer1 with 63BF
     TMR1H=0x63;
     T1CON=0x01;
     start_flag=1;
     half_flag=0;
     measurementcomp_flag=0;
     do
     {
          ADCON0=1;
          delay();       // acquisition time
          ADGO=1;        // starts A to D conversion
          while (ADIF!=1)
          res[i]= A2Dresult(); // digital samples
          i=i+1;
          delay1();
     }
     while(measurementcomp_flag==0);
     FFT();              // calls FFT function
}
```

```
void delay()
{
 // acquisition of analog signal
    int delay_time=0xFF;
    while (delay_time !=0)
    {
        --delay_time;
    }
}

void delay1()
{
 // sampling interval
    int i;
    for(i=0;i<5;i++)
        delay();
}

unsigned int A2Dresult()
{
 // storing A2D converted results
    int result=0;
    result=ADRESH;
    result=result<<8;
    result=result|ADRESL;
    return(result);
}

void Timer()
{
 // Timer Interrupt service routine
    if(start_flag==1)
    {
        if(half_flag==1)
        {
            start_flag=0;
            half_flag=0;
            measurementcomp_flag=1;
        }
```

```
  else
          half_flag=1;
      }
  }

  void keypress()
  {
   // Key press Interrupt service routine
      if (RBIF==1)
      {
          keydata=keydata+1;
      }
      if(keydata>5)
      {
          keydata=1;
      }
      else
      {
          RBIF=0;
      }
      display(harm[keydata]);
}

void display(unsigned int harm)
{
 // Displays the stored harmonic values
    int RB,counter=0,counter1=0;

    do
    {
        RD= 0;
        RD= RD | (harm & 0x0f);
        RB= RB & 0xfc;
        RB= RB|(counter & 0x03);
        counter1+=1;
        RD= 0;
        RD= RD|(harm & 0xf0);// LED segment value
        RB= RB|(counter & 0x03);
```

```
      counter+=1;
      counter1+=1;
      }
      while(counter1==2);
}

void fft()  // Calculation of FFT
{
   unsigned int rpart[64],ipart[64],output[64];
   float tp1,tp2,THD,THD1 = 0;
   int i,k,n,N=64;

   for(i=0;i<N;i++)
   {
       ipart[i] = 0;
   }

   for(k=0;k<n;k++)
   {
    tp1=0;
    tp2=0;

    for(n=0;n<N;n++)
    {

    tp1=tp1+rmul(rpart[n],ipart[n],cos(ANGLE(k,n)),
sin(ANGLE(k,n)));

    tp2=tp2+imul(rpart[n],ipart[n],cos(ANGLE(k,n)),
sin(ANGLE(k,n)));
    }
output[k]= sqrt(tp1*tp1+tp2*tp2);

output[i]=sqrt(rpart[i]*rpart[i]+ipart[i]*ipart[i];
    if(i == 0)
        output[0]=output[0]/2;
}
```

```
 for(n=2;n<N;n++)          // THD calculation
   {
     THD1 = THD1 + (output[n]) * (output[n]);
   }
   THD = sqrt(THD1)/output[1];
   harm[1]= output[1];
   harm[2]= output[3];
   harm[3]= output[5];
   harm[4]= THD;
 }
 float rmul(float a,float b,float c,float d)
 {
     float ans;
     ans=(a*c)+(b*d);
     return ans;
 }
 float imul(float a,float b,float c,float d)
 {
     float ans;
     ans=-(a*d)+(b*c);
     return (ans);
 }
```

ASSEMBLY LANGUAGE:

```
     processor    16F877
indf equ 0
rtcc equ 1
pc   equ 2
pcl  equ 2
status   equ 3
fsr  equ 4
porta    equ 5
portb    equ 6
portc    equ 7
pclath   equ 10
     global   _main
     signat   _main,66
     psect    text0,local,class=CODE,delta=2
     psect    text0
     file "D:\MYDIR\MYPROG\FFT\TRIAL\A2D.C"
     line 30
_main
; initialise();
     global   _initialise
     signat   _initialise,88
     FNCALL   _main,_initialise
     line 31
     fcall    (_initialise) ;calls the function
;initialise
; while(1)
     line 32
     goto l2
l3
; 33: {
;34: A2D();
     global   _A2D
     signat   _A2D,88
     FNCALL   _main,_A2D
     line 34
     fcall    (_A2D)       ;calls A to D function
;35: keypress();
     global   _keypress
     signat   _keypress,88
```

```
        FNCALL  _main,_keypress
        line35
        fcal(_keypress)  ;calls key press function
;36: }
        line36
l2
        line32
        goto l3
l4
;37: }
        line37
l1
        global  start
        ljmp start
        FNSIZE  _main,0,0
;39: void initialise()
;40: {
        psect   text1,local,class=CODE,delta=2
        psect   text1
        line40
_initialise             ;initializes the registers
;41: INTCON= 0xC8;
        line41
        movlw   ((-56))
        movwf   (((0Bh)))       ;volatile
;42: PIR1= 0x00;
        line42
        bcf status,5
        bcf status,6
        clrf(((0Ch)))       ;volatile
;43: PIE1= 0x41;
        line43
        movlw   ((041h))
        bsf status,5
        movwf   (((08Ch))^0x80)  ;volatile
;44: ADCON0=0x80;
        line44
        movlw   ((-128))
        bcf status,5
        movwf   (((01Fh)))  ;volatile
```

```
;45: ADCON1=0x2E;
     line 45
     movlw    ((02Eh))
     bsf  status,5
     movwf    (((09Fh))^0x80)   ;volatile
;46: TRISA= 0x37;
     line 46              ;intialises port A
     movlw    ((037h))
     movwf    (((085h))^0x80)   ;volatile
;47: TRISB= 0x7C;
     line 47              ;initialises port B
     movlw    ((07Ch))
     movwf    (((086h))^0x80)   ;volatile
;48: TRISD= 0x70;
     line 48             ;initialises portD
     movlw    ((070h))
     movwf    (((088h))^0x80)   ;volatile
;49: keydata = 1;
     global   _keydata
     line 49
     bcf  status,5
     clrf (((_keydata)))
     incf (((_keydata)))
;50: }
     line 50
l5
     bcf  status,6
     bcf  status,5
     return
     FNSIZE   _initialise,0,0
;52: void A2D()
;53: {
     psect    text2,local,class=CODE,delta=2
     psect    text2
     line 53
_A2D          ;A to D conversion
;54: int i=0;
;    _i assigned to ?a_A2D+0
_A2D$i   set  ?a_A2D+0
     line 54
```

```
        bcf  status,5
        bcf  status,6
        clrf (((?a_A2D+0)))
        clrf (((?a_A2D+0+1)))
 ;56: TMR1L=0xBF;
        line 56
        ;initialises timer1 to 63bf
        movlw    ((-65))
        movwf    (((0Eh)))    ;volatile
 ;57: TMR1H=0x63;
        line 57
        movlw    ((063h))
        movwf    (((0Fh)))    ;volatile
 ;58: T1CON=0x01;
        line 58
        movlw    ((01h))
        movwf    (((010h)))   ;volatile
 ;59: start_flag=1;
        global   _start_flag
        line 59
        bsf  (_start_flag/8),(_start_flag)&7
 ;60: half_flag=0;
        global   _half_flag
        line 60
        bcf  (_half_flag/8),(_half_flag)&7
 ;61: measurementcomp_flag=0;
        global   _measurementcomp_flag
        line 61
        bcf
        (_measurementcomp_flag/8),(_measurementcom
p_flag)&7
 ;62: do
        line 62
19
 ;63: {
 ;64: ADCON0=1;
        line 64
        movlw    ((01h))
        bcf  status,5
        bcf  status,6
```

```
        movwf   (((01Fh)))      ;volatile
    ;65: delay();
    ;acquisition time
        global  _delay
        signat  _delay,88
        FNCALL  _A2D,_delay
        line 65
        fcall   (_delay)        ;calls delay loop
    ;66: ADGO=1;
    ;starts A to D conversion
        line 66
        bsf  (0FAh/8),(0FAh)&7
    ;67: while (ADIF!=1)
    ;checks for completion of conversion
        line 67
        goto l10
    l11
    ;68: res[i]= A2Dresult();
    ;converted results are stored in res[i]
        global  _res
        global  _A2Dresult
        signat  _A2Dresult,74
        FNCALL  _A2D,_A2Dresult
        line 68
    ;calls A to D result function
        fcall   (_A2Dresult)
        movwf   (((?a_A2D+0))),w
        addwf   (((?a_A2D+0))),w
        addlw   ((_res))
        movwf   fsr

        bcf  3,7
        movf ((0+btemp)),w
        movwf   0
        incf fsr
        movf ((1+btemp)),w
        movwf   0

    l10
        line 67
```

```
        bcf  status,5
        bcf  status,6
        btfss    (066h/8),(066h)&7
        goto u11
        goto u10
  u11
        goto l11
  u10
  l12
  ;69: i=i+1;
  ;gets the next analog value
        line 69
        bcf  status,5
        bcf  status,6
        incf (((?a_A2D+0)))
        btfsc    status,2
        incf (((?a_A2D+0+1)))
  ;70: delay1();
  ;time for conversion
        global   _delay1
        signat   _delay1,88
        FNCALL   _A2D,_delay1
        line 70
  ;calls delay1 function
        fcall    (_delay1)
  ;71: }
  ;72: while(measurementcomp_flag==0);
        line 72
        btfss
        (_measurementcomp_flag/8),(_measurementcom
p_flag)&7
        goto u21
        goto u20
u21
        goto l9
u20
l8
;73: FFT();
;FFT calculation
      global   _FFT
```

```
        signat   _FFT,88
        FNCALL   _A2D,_FFT
        line 73
;calls FFT function
        fcall    (_FFT)
;74: }
        line 74
l6
        bcf status,6
        bcf status,5
        return
        FNSIZE   _A2D,2,0
        global   ?a_A2D
;76: void delay()
;delay function
;77: {
;time for acquisition
        psect    text3,local,class=CODE,delta=2
        psect    text3
        line 77
_delay
;78: int delay_time=0xFF;
;    _delay_time assigned to ?a_delay+0
_delay$delay_timeset ?a_delay+0
        line 78
        movlw    0FFh
        bcf status,5
        bcf status,6
        movwf    (((?a_delay+0)))
        clrf (((?a_delay+0+1)))
;79: while (delay_time !=0)
        line 79
        goto l14
l15
;80: {
;81: --delay_time;
        line 81
        movlw    -1
        bcf status,5
        bcf status,6
```

```
        addwf    (((?a_delay+0)))
        btfss    status,0
        decf (((?a_delay+0+1)))
    ;82: }
        line 82
    l14
        line 79
        bcf status,5
        bcf status,6
        movf (((?a_delay+0+1))),w
        iorwf    (((?a_delay+0))),w
        btfss    status,2
        goto u31
        goto u30
    u31
        goto l15
    u30
    l16
    ;83: }
        line 83
    l13
        bcf status,6
        bcf status,5
        return
        FNSIZE   _delay,2,0
        global   ?a_delay
    ;85: void delay1()
    ;delay1 function
    ;86: {
    ;time for conversion
        psect    text4,local,class=CODE,delta=2
        psect    text4
        line 86
_delay1
;87: int i;
;    _i assigned to ?a_delay1+0
_delay1$i    set ?a_delay1+0
;88: for(i=0;i<5;i++)
;calls the delay loop five times
    line 88
```

```
        bcf  status,5
        bcf  status,6
        clrf (((?a_delay1+0)))
        clrf (((?a_delay1+0+1)))
        movf (((?a_delay1+0+1))),w
        xorlw    80h
        movwf    btemp
        movlw    (0)^80h
        subwf    btemp,w
        movlw    05h
        ;setbank bits for (((?a_delay1+0)))
        btfsc    status,2
        subwf    (((?a_delay1+0))),w
        btfss    status,0
        goto u41
        goto u40
u41
        goto l18
u40
        goto l19
;89:  delay();
        line 89
l18
        FNCALL   _delay1,_delay

        fcall    (_delay)
        line 88
        incf (((?a_delay1+0)))
        btfsc    status,2
        incf (((?a_delay1+0+1)))
        movf (((?a_delay1+0+1))),w
        xorlw    80h
        movwf    btemp
        movlw    (0)^80h
        subwf    btemp,w
        movlw    05h
        ;setbank bits for (((?a_delay1+0)))
        btfsc    status,2
        subwf    (((?a_delay1+0))),w
        btfss    status,0
```

```
        goto u51
        goto u50
  u51
        goto l18
  u50
  l19
  ;90: }
        line 90
  l17
        bcf  status,6
        bcf  status,5
        return
        FNSIZE   _delay1,2,0
        global   ?a_delay1
  ;92: unsigned int A2Dresult()
  ;A to D converted result
  ;93: {
        psect    text5,local,class=CODE,delta=2
        psect    text5
        line 93
 _A2Dresult
 ;94: int result=0;
 ;    _result assigned to ?a_A2Dresult+0
 _A2Dresult$resultset ?a_A2Dresult+0
        line 94
        bcf  status,5
        bcf  status,6
        clrf (((?a_A2Dresult+0)))
        clrf (((?a_A2Dresult+0+1)))
 ;95: result=ADRESH;
        line 95
        movf (((01Eh))),w ;volatile
        movwf    (((?a_A2Dresult+0)))
        clrf (((?a_A2Dresult+0+1)))
 ;96: result=result<<8;
        line 96
        movf (((?a_A2Dresult+0))),w
        movwf    (((?a_A2Dresult+0+1)))
        clrf (((?a_A2Dresult+0)))
 ;97: result=result|ADRESL;
```

```
        line 97
        bsf  status,5
        movf (((09Eh))^0x80),w;volatile
        bcf  status,5
        iorwf    (((?a_A2Dresult+0)))
  ;98: return(result);
        line 98
        movf (((?a_A2Dresult+0+1))),w
        movwf    btemp+1
        movf (((?a_A2Dresult+0))),w
        movwf    btemp
        goto l21
  ;99: }
        line 99
  l21
        bcf  status,6
        bcf  status,5
        return
        FNSIZE   _A2Dresult,2,0
        global   ?a_A2Dresult
  ;101: void Timer()
  ;timer interrupt service routine
  ;102: {
        psect    text6,local,class=CODE,delta=2
        psect    text6
        global   _Timer
        signat   _Timer,88
        line 102
 _Timer
 ;103: if(start_flag==1)
        line 103
        bcf  status,5
        bcf  status,6
        btfss    (_start_flag/8),(_start_flag)&7
        goto u61
        goto u60
u61
        goto l23
u60
;104: {
```

```
;105: if(half_flag==1)
     line 105
     btfss    (_half_flag/8),(_half_flag)&7
     goto u71
     goto u70
u71
     goto l24
u70
;106: {
;107: start_flag=0;
     line 107
     bcf  (_start_flag/8),(_start_flag)&7
;108: half_flag=0;
     line 108
     bcf  (_half_flag/8),(_half_flag)&7
;109: measurementcomp_flag=1;
     line 109
     bsf
     (_measurementcomp_flag/8),(_measurementcom
p_flag)&7
;110: }
;111: else
     line 110
     goto l25
     line 111
l24
;112: half_flag=1;
     line 112
     bcf status,5
     bcf status,6
     bsf  (_half_flag/8),(_half_flag)&7
l25
;113: }
;114: }
     line 113
l23
     line 114
l22
     bcf status,6
     bcf status,5
```

```
        return
        FNSIZE  _Timer,0,0
  ;116: void keypress()
  ;117: {
        psect   text7,local,class=CODE,delta=2
        psect   text7
        line 117
  _keypress
  ;118: if (RBIF==1)
  ;checks for key press flag
        line 118
        btfss   (058h/8),(058h)&7
        goto u81
        goto u80
  u81
        goto l27
  u80
  ;119: {
  ;120: keydata=keydata+1;
        line 120
        bcf  status,5
        bcf  status,6
        incf (((_keydata)))
  ;121: }
  ;122: if(keydata>5)
        line 121
  l27
        line 122
        movlw   ((06h))
        bcf  status,5
        bcf  status,6
        subwf   (((_keydata))),w
        btfss   status,0
        goto u91
        goto u90
u91
      goto l28
u90
;123: {
;124: keydata=1;
```

```
        line 124
        clrf (((_keydata)))
        incf (((_keydata)))
;125: }
;126: else
        line 125
        goto l29
        line 126
l28
;127: {
;128: RBIF=0;
        line 128
        bcf  (058h/8),(058h)&7
;129: }
        line 129
l29
;130: display(harm[keydata]);
;calls display function
        global   _display
        signat   _display,4216
        global   _harm
        global   ?_display
        FNCALL   _keypress,_display
        line 130
        bcf  status,5
        bcf  status,6
        movwf    (((_keydata))),w
        addwf    (((_keydata))),w
        addlw    ((_harm))
        movwf    fsr

        bcf  3,7
        movf 0,w
        movwf    (((?_display)))
        incf fsr
        movf 0,w
        movwf    (((?_display+1)))

        fcall    (_display)
;131: }
```

```
        line131
    l26
        bcf  status,6
        bcf  status,5
        return
        FNSIZE   _keypress,0,0
    ;133: void display(unsigned int harm)
    ;display function
    ;134: {
    ;       param _harm assigned to ?_display+0
    _display$harm    set ?_display+0
        psect    text8,local,class=CODE,delta=2
        psect    text8
        line134
    _display
    ;135: int RB,counter=0,counter1=0;
    ;    _RB assigned to ?a_display+0
    _display$RB  set ?a_display+0
    ;    _counter assigned to ?a_display+2
    _display$counter set ?a_display+2
        line135
        bcf  status,5
        bcf  status,6
        clrf (((?a_display+2)))
        clrf (((?a_display+2+1)))
    ;    _counter1 assigned to ?a_display+4
    _display$counter1set ?a_display+4
        clrf (((?a_display+4)))
        clrf (((?a_display+4+1)))
    ;137: do
        line137
    l33
    ;138: {
    ;139: RD= 0;
        line139
        bsf  status,5
        bsf  status,6
        bcf  (0C60h/8)^0x180,(0C60h)&7
    ;140: RD= RD | (harm & 0x0f);
        line140
```

```
        bcf  status,5
        bcf  status,6
        rrf  (((?_display+0))),w
        btfsc    status,0
        goto u101
        goto u100
 u101
        bsf  status,5
        bsf  status,6
        bsf  (0C60h/8)^0x180,(0C60h)&7
 u100
 ;141: RB= RB & 0xfc;
        line 141
        movlw    0FCh
        bcf  status,5
        bcf  status,6
        andwf    (((?a_display+0)))
        clrf (((?a_display+0+1)))
 ;142: RB= RB|(counter & 0x03);
        line 142
        movf (((?a_display+2+1))),w
        movwf    btemp+1
        movf (((?a_display+2))),w
        movwf    btemp
        movlw    03h
        andwf    btemp
        clrf btemp+1
        movwf    btemp,w
        iorwf    (((?a_display+0)))
        movwf    btemp+1,w
        iorwf    (((?a_display+0+1)))
 ;143: counter1+=1;
        line 143
        incf (((?a_display+4)))
        btfsc    status,2
        incf (((?a_display+4+1)))
 ;144: RD= 0;
        line 144
        bsf  status,5
        bsf  status,6
```

```
        bcf  (0C60h/8)^0x180,(0C60h)&7
  ;145: RD= RD | (harm & 0xf0);
        line145
  ;146: RB= RB|(counter & 0x03);
        line146
        bcf  status,5
        bcf  status,6
        movf (((?a_display+2+1))),w
        movwf    btemp+1
        movf (((?a_display+2))),w
        movwf    btemp
        movlw    03h
        andwf    btemp
        clrf btemp+1
        movwf    btemp,w
        iorwf    (((?a_display+0)))
        movwf    btemp+1,w
        iorwf    (((?a_display+0+1)))
  ;147: counter+=1;
        line147
        incf (((?a_display+2)))
        btfsc    status,2
        incf (((?a_display+2+1)))
  ;148: counter1+=1;
        line148
        incf (((?a_display+4)))
        btfsc    status,2
        incf (((?a_display+4+1)))
  ;149: }
  ;150: while(counter1==2);
        line150
        movf (((?a_display+4))),w
        xorlw    2
        iorwf (((?a_display+4+1))),w

        btfsc    status,2
        goto u111
        goto u110
u111
        goto l33
```

```
u110
l32
;151: }
    line151
l30
    bcf  status,6
    bcf  status,5
    return
    FNSIZE   _display,6,2
    global   ?a_display
    global   ?_display
;153: void fft()
;FFT function
;154: {
    psect    text9,local,class=CODE,delta=2
    psect    text9
    global   _fft
    signat   _fft,88
    line154
_fft
;155: unsigned int
rpart[64],ipart[64],output[64];
;    _rpart assigned to ?a_fft+0
_fft$rpart   set  ?a_fft+0
;    _ipart assigned to ?a_fft+128
_fft$ipart   set  ?a_fft+128
;    _output assigned to ?a_fft+256
_fft$output  set  ?a_fft+256
;156: float tp1,tp2,THD,THD1 = 0;
;    _tp1 assigned to ?a_fft+384
_fft$tp1 set  ?a_fft+384
;    _tp2 assigned to ?a_fft+387
_fft$tp2 set  ?a_fft+387
;    _THD assigned to ?a_fft+390
_fft$THD set  ?a_fft+390
;    _THD1 assigned to ?a_fft+393
_fft$THD1    set  ?a_fft+393
    line156
    movlw
    low(((float24(0.00000000000000000)))
```

```
        bcf  status,5
        bcf  status,6
        movwf    (((?a_fft+393)))
        movlw
        high(((float24(0.00000000000000000)))
        movwf    (((?a_fft+393+1)))
        movlw    low
highword(((float24(0.00000000000000000)))
        movwf    (((?a_fft+393+2)))
;157: int i,k,n,N=64;
;    _i assigned to ?a_fft+396
_fft$i   set  ?a_fft+396
;    _k assigned to ?a_fft+398
_fft$k   set  ?a_fft+398
;    _n assigned to ?a_fft+400
_fft$n   set  ?a_fft+400
;    _N assigned to ?a_fft+402
_fft$N   set  ?a_fft+402
        line157
        movlw    040h
        movwf    (((?a_fft+402)))
        clrf (((?a_fft+402+1)))
;159: for(i=0;i<N;i++)
        line159
        clrf (((?a_fft+396)))
        clrf (((?a_fft+396+1)))
        goto138
;160: {
        line160
l35
;161: ipart[i] = 0;
        line161
        bcf  status,5
        bcf  status,6
        movwf    (((?a_fft+396))),w
        addwf    (((?a_fft+396))),w
        addlw    ((?a_fft+128))
        movwf    fsr

        bcf  3,7
```

```
        movlw    ((0))
        movwf    0
        incf fsr
        movlw    ((0))
        movwf    0
 ;162: }
        line 159
        incf (((?a_fft+396)))
        btfsc    status,2
        incf (((?a_fft+396+1)))
 l38
        bcf  status,5
        bcf  status,6
        movf (((?a_fft+396+1))),w
        xorlw    80h
        movwf    btemp
        movwf    (((?a_fft+402+1))),w
        xorlw    80h
        subwf    btemp,w
        btfss    status,2
        goto u125
        movwf    (((?a_fft+402))),w
        subwf    (((?a_fft+396))),w
u125
        btfss    status,0
        goto u121
        goto u120
u121
        goto l35
u120
l36
;164: for(k=0;k<n;k++)
        line 164
        bcf  status,5
        bcf  status,6
        clrf (((?a_fft+398)))
        clrf (((?a_fft+398+1)))
        goto l42
;165: {
        line 165
```

```
 139
 ;166: tp1=0;
      line 166
      movlw
      low(((float24(0.00000000000000000)))
      bcf  status,5
      bcf  status,6
      movwf     (((?a_fft+384)))
      movlw
      high(((float24(0.00000000000000000))))
      movwf     (((?a_fft+384+1)))
      movlw     low
 highword(((float24(0.00000000000000000))))
      movwf     (((?a_fft+384+2)))
 ;167: tp2=0;
      line 167
      movlw
      low(((float24(0.00000000000000000))))
      movwf     (((?a_fft+387)))
      movlw
      high(((float24(0.00000000000000000))))
      movwf     (((?a_fft+387+1)))
      movlw     low
 highword(((float24(0.00000000000000000))))
      movwf     (((?a_fft+387+2)))
 ;169: for(n=0;n<N;n++)
      line 169
      clrf (((?a_fft+400)))
      clrf (((?a_fft+400+1)))
      goto 146
 ;170: {
      line 170
 143
 ;171:
 tp1=tp1+rmul(rpart[n],ipart[n],cos(((2*k*n*3.14
 1592)/N)),sin(((2*k*n*3.141592)/N)));
      global   _rmul
      signat   _rmul,16499
      global   _cos
      signat   _cos,4212
```

```
global   _sin
signat   _sin,4212
global   ?_cos
FNCALL   _fft,_cos
global   ?_sin
FNCALL   _fft,_sin
global   ?_rmul
FNARG    _rmul,_cos
FNARG    _rmul,_sin
FNCALL   _fft,_rmul
line171
bcf  status,5
bcf  status,6
movwf    (((?a_fft+400))),w
addwf    (((?a_fft+400))),w
addlw    ((?a_fft+0))
movwf    fsr

bcf  3,7
movf 0,w
movwf    btemp
incf fsr
movf 0,w
movwf    btemp+1
global   lwtoft
fcall    lwtoft
movwf    btemp,w
movwf    (((?_rmul)))
movwf    btemp+1,w
movwf    (((?_rmul+1)))
movwf    btemp+2,w
movwf    (((?_rmul+2)))
movwf    (((?a_fft+400))),w
addwf    (((?a_fft+400))),w
addlw    ((?a_fft+128))
movwf    fsr

bcf  3,7
movf 0,w
movwf    btemp
```

```
    incf fsr
    movf 0,w
    movwf    btemp+1
    global   lwtoft
    fcall    lwtoft
    movwf    btemp,w
    movwf    ((0+(?_rmul+03h)))
    movwf    btemp+1,w
    movwf    ((1+(?_rmul+03h)))
    movwf    btemp+2,w
    movwf    ((2+(?_rmul+03h)))
    movf (((?a_fft+402+1))),w
    movwf    btemp+1
    movf (((?a_fft+402))),w
    movwf    btemp
    global   awtofl
    fcall    awtofl
    movf btemp,w
    movwf    f2793+0
    movf btemp+1,w
    movwf    f2793+0+1
    movf btemp+2,w
    movwf    f2793+0+2
    movf btemp+3,w
    movwf    f2793+0+3
    movf (((?a_fft+400+1))),w
    movwf    btemp+3
    movf (((?a_fft+400))),w
    movwf    btemp+2
    movf (((?a_fft+398+1))),w
    movwf    btemp+1
    movf (((?a_fft+398))),w
    movwf    btemp
    bcf  status,0
    rlf  btemp
    rlf  btemp+1
    global   awmul
    fcall    awmul
    movwf    btemp+4,w
    movwf    btemp
```

```
        movwf   btemp+5,w
        movwf   btemp+1
        global  awtofl
        fcall   awtofl
        movlw   low(((3.1415920000000000)))
        movwf   btemp+4
        movlw   high(((3.1415920000000000)))
        movwf   btemp+5
        movlw   low
highword(((3.1415920000000000)))
        movwf   btemp+6
        movlw   high
highword(((3.1415920000000000)))
        movwf   btemp+7
        global  flmul
        fcall   flmul
        movf f2793+0,w
        movwf   btemp+4
        movf f2793+0+1,w
        movwf   btemp+5
        movf f2793+0+2,w
        movwf   btemp+6
        movf f2793+0+3,w
        movwf   btemp+7
        global  fldiv
        fcall   fldiv
        movwf   btemp,w
        movwf   (((?_cos)))
        movwf   btemp+1,w
        movwf   (((?_cos+1)))
        movwf   btemp+2,w
        movwf   (((?_cos+2)))
        movwf   btemp+3,w
        movwf   (((?_cos+3)))

        fcall   (_cos)
        movf btemp+1,w
        movwf   btemp
        movf btemp+2,w
        movwf   btemp+1
```

```
movf btemp+3,w
movwf    btemp+2
movwf    btemp,w
movwf    ((0+(?_rmul+06h)))
movwf    btemp+1,w
movwf    ((1+(?_rmul+06h)))
movwf    btemp+2,w
movwf    ((2+(?_rmul+06h)))
movf (((?a_fft+402+1))),w
movwf    btemp+1
movf (((?a_fft+402))),w
movwf    btemp
global   awtofl
fcall    awtofl
movf btemp,w
movwf    f2793+4
movf btemp+1,w
movwf    f2793+4+1
movf btemp+2,w
movwf    f2793+4+2
movf btemp+3,w
movwf    f2793+4+3
movf (((?a_fft+400+1))),w
movwf    btemp+3
movf (((?a_fft+400))),w
movwf    btemp+2
movf (((?a_fft+398+1))),w
movwf    btemp+1
movf (((?a_fft+398))),w
movwf    btemp
bcf  status,0
rlf  btemp
rlf  btemp+1
global   awmul
fcall    awmul
movwf    btemp+4,w
movwf    btemp
movwf    btemp+5,w
movwf    btemp+1
global   awtofl
```

```
        fcall    awtofl
        movlw    low(((3.1415920000000000)))
        movwf    btemp+4
        movlw    high(((3.1415920000000000)))
        movwf    btemp+5
        movlw    low
highword(((3.1415920000000000)))
        movwf    btemp+6
        movlw    high
highword(((3.1415920000000000)))
        movwf    btemp+7
        global   flmul
        fcall    flmul
        movf f2793+4,w
        movwf    btemp+4
        movf f2793+4+1,w
        movwf    btemp+5
        movf f2793+4+2,w
        movwf    btemp+6
        movf f2793+4+3,w
        movwf    btemp+7
        global   fldiv
        fcall    fldiv
        movwf    btemp,w
        movwf    (((?_sin)))
        movwf    btemp+1,w
        movwf    (((?_sin+1)))
        movwf    btemp+2,w
        movwf    (((?_sin+2)))
        movwf    btemp+3,w
        movwf    (((?_sin+3)))

        fcall    (_sin)
        movf btemp+1,w
        movwf    btemp
        movf btemp+2,w
        movwf    btemp+1
        movf btemp+3,w
        movwf    btemp+2
        movwf    btemp,w
```

```
        movwf     ((0+(?_rmul+09h)))
        movwf     btemp+1,w
        movwf     ((1+(?_rmul+09h)))
        movwf     btemp+2,w
        movwf     ((2+(?_rmul+09h)))

        fcall     (_rmul)
        movf ((0+btemp)),w
        movwf     btemp+3
        movf ((1+btemp)),w
        movwf     btemp+4
        movf ((2+btemp)),w
        movwf     btemp+5
        movlw     ((?a_fft+384))
        movwf     fsr
        bcf  3,7
        global    ftadd_f
        fcall     ftadd_f
;172:
tp2=tp2+imul(rpart[n],ipart[n],cos(((2*k*n*3.14
1592)/N)),sin(((2*k*n*3.141592)/N)));
      global    _imul
      signat    _imul,16499
      global    ?_imul
      FNARG     _imul,_cos
      FNARG     _imul,_sin
      FNCALL    _fft,_imul
      line 172
      bcf  status,5
      bcf  status,6
      movwf     (((?a_fft+400))),w
      addwf     (((?a_fft+400))),w
      addlw     ((?a_fft+0))
      movwf     fsr

     bcf  3,7
     movf 0,w
     movwf     btemp
     incf fsr
     movf 0,w
```

```
movwf    btemp+1
global   lwtoft
fcall    lwtoft
movwf    btemp,w
movwf    (((?_imul)))
movwf    btemp+1,w
movwf    (((?_imul+1)))
movwf    btemp+2,w
movwf    (((?_imul+2)))
movwf    (((?a_fft+400))),w
addwf    (((?a_fft+400))),w
addlw    ((?a_fft+128))
movwf    fsr

bcf  3,7
movf 0,w
movwf    btemp
incf fsr
movf 0,w
movwf    btemp+1
global   lwtoft
fcall    lwtoft
movwf    btemp,w
movwf    ((0+(?_imul+03h)))
movwf    btemp+1,w
movwf    ((1+(?_imul+03h)))
movwf    btemp+2,w
movwf    ((2+(?_imul+03h)))
movf (((?a_fft+402+1))),w
movwf    btemp+1
movf (((?a_fft+402))),w
movwf    btemp
global   awtof1
fcall    awtof1
movf btemp,w
movwf    f2793+0
movf btemp+1,w
movwf    f2793+0+1
movf btemp+2,w
movwf    f2793+0+2
```

```
        movf btemp+3,w
        movwf    f2793+0+3
        movf (((?a_fft+400+1))),w
        movwf    btemp+3
        movf (((?a_fft+400))),w
        movwf    btemp+2
        movf (((?a_fft+398+1))),w
        movwf    btemp+1
        movf (((?a_fft+398))),w
        movwf    btemp
        bcf  status,0
        rlf  btemp
        rlf  btemp+1
        global   awmul
        fcall    awmul
        movwf    btemp+4,w
        movwf    btemp
        movwf    btemp+5,w
        movwf    btemp+1
        global   awtofl
        fcall    awtofl
        movlw    low(((3.1415920000000000)))
        movwf    btemp+4
        movlw    high(((3.1415920000000000)))
        movwf    btemp+5
        movlw    low
highword(((3.1415920000000000)))
        movwf    btemp+6
        movlw    high
highword(((3.1415920000000000)))
        movwf    btemp+7
        global   flmul
        fcall    flmul
        movf f2793+0,w
        movwf    btemp+4
        movf f2793+0+1,w
        movwf    btemp+5
        movf f2793+0+2,w
        movwf    btemp+6
        movf f2793+0+3,w
```

```
movwf    btemp+7
global   fldiv
fcall    fldiv
movwf    btemp,w
movwf    (((?_cos)))
movwf    btemp+1,w
movwf    (((?_cos+1)))
movwf    btemp+2,w
movwf    (((?_cos+2)))
movwf    btemp+3,w
movwf    (((?_cos+3)))

fcall    (_cos)
movf btemp+1,w
movwf    btemp
movf btemp+2,w
movwf    btemp+1
movf btemp+3,w
movwf    btemp+2
movwf    btemp,w
movwf    ((0+(?_imul+06h)))
movwf    btemp+1,w
movwf    ((1+(?_imul+06h)))
movwf    btemp+2,w
movwf    ((2+(?_imul+06h)))
movf (((?a_fft+402+1))),w
movwf    btemp+1
movf (((?a_fft+402))),w
movwf    btemp
global   awtofl
fcall    awtofl
movf btemp,w
movwf    f2793+4
movf btemp+1,w
movwf    f2793+4+1
movf btemp+2,w
movwf    f2793+4+2
movf btemp+3,w
movwf    f2793+4+3
movf (((?a_fft+400+1))),w
```

```
        movwf    btemp+3
        movf (((?a_fft+400))),w
        movwf    btemp+2
        movf (((?a_fft+398+1))),w
        movwf    btemp+1
        movf (((?a_fft+398))),w
        movwf    btemp
        bcf  status,0
        rlf  btemp
        rlf  btemp+1
        global   awmul
        fcall    awmul
        movwf    btemp+4,w
        movwf    btemp
        movwf    btemp+5,w
        movwf    btemp+1
        global   awtofl
        fcall    awtofl
        movlw    low(((3.1415920000000000)))
        movwf    btemp+4
        movlw    high(((3.1415920000000000)))
        movwf    btemp+5
        movlw    low
highword(((3.1415920000000000)))
        movwf    btemp+6
        movlw    high
highword(((3.1415920000000000)))
        movwf    btemp+7
        global   flmul
        fcall    flmul
        movf f2793+4,w
        movwf    btemp+4
        movf f2793+4+1,w
        movwf    btemp+5
        movf f2793+4+2,w
        movwf    btemp+6
        movf f2793+4+3,w
        movwf    btemp+7
        global   fldiv
        fcall    fldiv
```

```
        movwf   btemp,w
        movwf   (((?_sin)))
        movwf   btemp+1,w
        movwf   (((?_sin+1)))
        movwf   btemp+2,w
        movwf   (((?_sin+2)))
        movwf   btemp+3,w
        movwf   (((?_sin+3)))

        fcall   (_sin)
        movf btemp+1,w
        movwf   btemp
        movf btemp+2,w
        movwf   btemp+1
        movf btemp+3,w
        movwf   btemp+2
        movwf   btemp,w
        movwf   ((0+(?_imul+09h)))
        movwf   btemp+1,w
        movwf   ((1+(?_imul+09h)))
        movwf   btemp+2,w
        movwf   ((2+(?_imul+09h)))

        fcall   (_imul)
        movf ((0+btemp)),w
        movwf   btemp+3
        movf ((1+btemp)),w
        movwf   btemp+4
        movf ((2+btemp)),w
        movwf   btemp+5
        movlw   ((?a_fft+387))
        movwf   fsr
        bcf 3,7
        global  ftadd_f
        fcall   ftadd_f
;173: }

    line169
    bcf status,5
    bcf status,6
```

```
    incf (((?a_fft+400)))
    btfsc   status,2
    incf (((?a_fft+400+1)))

l46
    bcf  status,5
    bcf  status,6
    movf (((?a_fft+400+1))),w
    xorlw   80h
    movwf   btemp
    movwf   (((?a_fft+402+1))),w
    xorlw   80h
    subwf   btemp,w
    btfss   status,2
    goto u135
    movwf   (((?a_fft+402))),w
    subwf   (((?a_fft+400))),w
u135

    btfss   status,0
    goto u131
    goto u130
u131
    goto l43
u130
l44
;174: output[k]= sqrt(tp1*tp1+tp2*tp2);
;gives the FFT output
    global  _sqrt
    signat  _sqrt,4212
    global  ?_sqrt
    FNCALL  _fft,_sqrt
    line 174
    bcf  status,5
    bcf  status,6
    movf (((?a_fft+387))),w
    movwf   btemp+3
    movf (((?a_fft+387+1))),w
    movwf   btemp+4
    movf (((?a_fft+387+2))),w
```

```
movwf     btemp+5
movf (((?a_fft+387))),w
movwf     btemp
movf (((?a_fft+387+1))),w
movwf     btemp+1
movf (((?a_fft+387+2))),w
movwf     btemp+2
global    ftmul
fcall     ftmul
movf btemp,w
movwf     f2793+0
movf btemp+1,w
movwf     f2793+0+1
movf btemp+2,w
movwf     f2793+0+2

movf (((?a_fft+384))),w
movwf     btemp+3
movf (((?a_fft+384+1))),w
movwf     btemp+4
movf (((?a_fft+384+2))),w
movwf     btemp+5
movf (((?a_fft+384))),w
movwf     btemp
movf (((?a_fft+384+1))),w
movwf     btemp+1
movf (((?a_fft+384+2))),w
movwf     btemp+2
global    ftmul
fcall     ftmul
movf f2793+0,w
movwf     btemp+3
movf f2793+0+1,w
movwf     btemp+4
movf f2793+0+2,w
movwf     btemp+5
global    ftadd
fcall     ftadd
movf btemp+2,w
movwf     btemp+3
```

```
        movf btemp+1,w
        movwf    btemp+2
        movf btemp,w
        movwf    btemp+1
        clrf btemp
        movwf    btemp,w
        movwf    (((?_sqrt)))
        movwf    btemp+1,w
        movwf    (((?_sqrt+1)))
        movwf    btemp+2,w
        movwf    (((?_sqrt+2)))
        movwf    btemp+3,w
        movwf    (((?_sqrt+3)))

        fcall    (_sqrt)
        global   fltol
        fcall    fltol
        movwf    (((?a_fft+398))),w
        addwf    (((?a_fft+398))),w
        addlw    ((?a_fft+256))
        movwf    fsr

      bcf  3,7
      movf btemp,w
      movwf    0
      incf fsr
      movf btemp+1,w
      movwf    0
;175: output[i]= (2 *
;sqrt(rpart[i]*rpart[i]+ipart[i]*ipart[i])/N);
      line 175
      movf (((?a_fft+402+1))),w
      movwf    btemp+1
      movf (((?a_fft+402))),w
      movwf    btemp
      global   awtofl
      fcall    awtofl
     movf btemp,w
     movwf    f2793+0
     movf btemp+1,w
```

```
movwf     f2793+0+1
movf btemp+2,w
movwf     f2793+0+2
movf btemp+3,w
movwf     f2793+0+3
movwf     (((?a_fft+396))),w
addwf     (((?a_fft+396))),w
addlw     ((?a_fft+128))
movwf     fsr

bcf  3,7
movf 0,w
movwf     btemp+2
incf fsr
movf 0,w
movwf     btemp+3
movwf     (((?a_fft+396))),w
addwf     (((?a_fft+396))),w
addlw     ((?a_fft+128))
movwf     fsr

bcf  3,7
movf 0,w
movwf     btemp
incf fsr
movf 0,w
movwf     btemp+1
global    lwmul
fcall     lwmul
movf btemp+4,w
movwf     f2793+4
movf btemp+5,w
movwf     f2793+4+1
movwf     (((?a_fft+396))),w
addwf     (((?a_fft+396))),w
addlw     ((?a_fft+0))
movwf     fsr

bcf  3,7
movf 0,w
```

```
movwf    btemp+2
incf fsr
movf 0,w
movwf    btemp+3
movwf    (((?a_fft+396))),w
addwf    (((?a_fft+396))),w
addlw    ((?a_fft+0))
movwf    fsr

bcf  3,7
movf 0,w
movwf    btemp
incf fsr
movf 0,w
movwf    btemp+1
global   lwmul
fcall    lwmul
movwf    f2793+4,w
addwf    btemp+4
btfsc    status,0
incf btemp+5
movwf    f2793+4+1,w
addwf    btemp+5
movwf    btemp+4,w
movwf    btemp
movwf    btemp+5,w
movwf    btemp+1
global   lwtofl
fcall    lwtofl
movwf    btemp,w
movwf    (((?_sqrt)))
movwf    btemp+1,w
movwf    (((?_sqrt+1)))
movwf    btemp+2,w
movwf    (((?_sqrt+2)))
movwf    btemp+3,w
movwf    (((?_sqrt+3)))

fcall    (_sqrt)
movlw    low(((2.0000000000000000)))
```

```
        movwf   btemp+4
        movlw   high(((2.0000000000000000)))
        movwf   btemp+5
        movlw   low
    highword(((2.0000000000000000)))
        movwf   btemp+6
        movlw   high
    highword(((2.0000000000000000)))
        movwf   btemp+7
        global  flmul
        fcall   flmul
        movf f2793+0,w
        movwf   btemp+4
        movf f2793+0+1,w
        movwf   btemp+5
        movf f2793+0+2,w
        movwf   btemp+6
        movf f2793+0+3,w
        movwf   btemp+7
        global  fldiv
        fcall   fldiv
        global  fltol
        fcall   fltol
        movwf   (((?a_fft+396))),w
        addwf   (((?a_fft+396))),w
        addlw   ((?a_fft+256))
        movwf   fsr

        bcf  3,7
        movf btemp,w
        movwf   0
        incf fsr
        movf btemp+1,w
        movwf   0
;176: if(i == 0)

        line 176
        movf (((?a_fft+396+1))),w
        iorwf   (((?a_fft+396))),w
        btfss   status,2
```

```
        goto u141
        goto u140
    u141
        goto l47
    u140
    ;177: output[0]=output[0]/2;
    ;DC component
        line 177
        bcf  status,0
        rrf  (((?a_fft+256+1)))
        rrf  (((?a_fft+256)))
    ;178: }
    l47
        line 164
        bcf  status,5
        bcf  status,6
        incf (((?a_fft+398)))
        btfsc   status,2
        incf (((?a_fft+398+1)))
   l42
        bcf  status,5
        bcf  status,6
        movf (((?a_fft+398+1))),w
        xorlw   80h
        movwf   btemp
        movwf   (((?a_fft+400+1))),w
        xorlw   80h
        subwf   btemp,w
        btfss   status,2
        goto u155
        movwf   (((?a_fft+400))),w
        subwf   (((?a_fft+398))),w
  u155

        btfss   status,0
        goto u151
        goto u150
  u151
       goto l39
  u150
```

```
140
;180: for(n=2;n<N;n++)
     line 180
     movlw    02h
     bcf  status,5
     bcf  status,6
     movwf    (((?a_fft+400)))
     clrf (((?a_fft+400+1)))
     goto 151
;181: {
     line 181

148
;182: THD1 = THD1 + (output[n]) * (output[n]);
;total harmonic distortion calculation
     line 182
     bcf  status,5
     bcf  status,6
     movwf    (((?a_fft+400))),w
     addwf    (((?a_fft+400))),w
     addlw    ((?a_fft+256))
     movwf    fsr

    bcf  3,7
    movf 0,w
    movwf    btemp+2
    incf fsr
    movf 0,w
    movwf    btemp+3
    movwf    (((?a_fft+400))),w
    addwf    (((?a_fft+400))),w
    addlw    ((?a_fft+256))
    movwf    fsr

   bcf  3,7
   movf 0,w
   movwf    btemp
   incf fsr
   movf 0,w
   movwf    btemp+1
```

```
    global   lwmul
    fcall    lwmul
    movwf    btemp+4,w
    movwf    btemp
    movwf    btemp+5,w
    movwf    btemp+1
    global   lwtoft
    fcall    lwtoft
    movf btemp,w
    movwf    btemp+3
    movf btemp+1,w
    movwf    btemp+4
    movf btemp+2,w
    movwf    btemp+5
    movlw    ((?a_fft+393))
    movwf    fsr
    bcf 3,7
    global   ftadd_f
    fcall    ftadd_f
;183: }
    line 180
    bcf status,5
    bcf status,6
    incf (((?a_fft+400)))
    btfsc    status,2
    incf (((?a_fft+400+1)))

l51
    bcf status,5
    bcf status,6
    movf (((?a_fft+400+1))),w
    xorlw    80h
    movwf    btemp
    movwf    (((?a_fft+402+1))),w
    xorlw    80h
    subwf    btemp,w
    btfss    status,2
    goto u165
    movwf    (((?a_fft+402))),w
    subwf    (((?a_fft+400))),w
```

```
u165

     btfss    status,0
     goto u161
     goto u160
u161
     goto l48
u160
l49
;184: THD = sqrt(THD1)/output[1];
     line 184
     bcf  status,5
     bcf  status,6
     movf ((1+(?a_fft+256+02h))),w
     movwf    btemp+1
     movf ((0+(?a_fft+256+02h))),w
     movwf    btemp
     global   lwtofl
     fcall    lwtofl
     movf btemp,w
     movwf    f2793+0
     movf btemp+1,w
     movwf    f2793+0+1
     movf btemp+2,w
     movwf    f2793+0+2
     movf btemp+3,w
     movwf    f2793+0+3
     movf (((?a_fft+393+2))),w
     movwf    btemp+3
     movf (((?a_fft+393+1))),w
     movwf    btemp+2
     movf (((?a_fft+393))),w
     movwf    btemp+1
     clrf btemp
     movwf    btemp,w
     movwf    (((?_sqrt)))
     movwf    btemp+1,w
     movwf    (((?_sqrt+1)))
     movwf    btemp+2,w
     movwf    (((?_sqrt+2)))
```

```
        movwf     btemp+3,w
        movwf     (((?_sqrt+3)))

        fcall     (_sqrt)
        movf f2793+0,w
        movwf     btemp+4
        movf f2793+0+1,w
        movwf     btemp+5
        movf f2793+0+2,w
        movwf     btemp+6
        movf f2793+0+3,w
        movwf     btemp+7
        global    fldiv
        fcall     fldiv
        movf btemp+1,w
        movwf     btemp
        movf btemp+2,w
        movwf     btemp+1
        movf btemp+3,w
        movwf     btemp+2
        movwf     btemp,w
        movwf     (((?a_fft+390)))
        movwf     btemp+1,w
        movwf     (((?a_fft+390+1)))
        movwf     btemp+2,w
        movwf     (((?a_fft+390+2)))
;185: harm[1]= output[1];
     line 185
     movf ((0+(?a_fft+256+02h))),w
     movwf     ((0+(_harm+02h)))
     movf ((1+(?a_fft+256+02h))),w
     movwf     ((1+(_harm+02h)))
;186: harm[2]= output[3];
     line 186
     movf ((0+(?a_fft+256+06h))),w
     movwf     ((0+(_harm+04h)))
     movf ((1+(?a_fft+256+06h))),w
     movwf     ((1+(_harm+04h)))
;187: harm[3]= output[5];
     line 187
```

```
        movf ((0+(?a_fft+256+0Ah))),w
        movwf     ((0+(_harm+06h)))
        movf ((1+(?a_fft+256+0Ah))),w
        movwf     ((1+(_harm+06h)))
  ;188: harm[4]= THD;
        line188
        movf (((?a_fft+390))),w
        movwf     btemp
        movf (((?a_fft+390+1))),w
        movwf     btemp+1
        movf (((?a_fft+390+2))),w
        movwf     btemp+2
        global    fttol
        fcall     fttol
        movwf     btemp,w
        movwf     ((0+(_harm+08h)))
        movwf     btemp+1,w
        movwf     ((1+(_harm+08h)))
  ;189: }
        line189
  l34
        bcf  status,6
        bcf  status,5
        return
        FNSIZE    _fft,412,0
        global    ?a_fft
  f2793     equ  ?a_fft+404
  ;190: float rmul(float a,float b,float c,float
  ;d)
  ;191: {
  ;         param _a assigned to ?_rmul+0
  _rmul$a  set  ?_rmul+0
  ;         param _b assigned to ?_rmul+3
  _rmul$b  set  ?_rmul+3
  ;         param _c assigned to ?_rmul+6
  _rmul$c  set  ?_rmul+6
  ;         param _d assigned to ?_rmul+9
  _rmul$d  set  ?_rmul+9
      psect     text10,local,class=CODE,delta=2
      psect     text10
```

```
    line 191
_rmul
;192: float ans;
;    _ans assigned to ?a_rmul+0
_rmul$ans    set ?a_rmul+0
;193: ans=(a*c)+(b*d);
    line 193
    bcf  status,5
    bcf  status,6
    movf (((?_rmul+9))),w
    movwf    btemp+3
    movf (((?_rmul+9+1))),w
    movwf    btemp+4
    movf (((?_rmul+9+2))),w
    movwf    btemp+5
    movf (((?_rmul+3))),w
    movwf    btemp
    movf (((?_rmul+3+1))),w
    movwf    btemp+1
    movf (((?_rmul+3+2))),w
    movwf    btemp+2
    global   ftmul
    fcall    ftmul
    movf btemp,w
    movwf    f2853+0
    movf btemp+1,w
    movwf    f2853+0+1
    movf btemp+2,w
    movwf    f2853+0+2

    movf (((?_rmul+6))),w
    movwf    btemp+3
    movf (((?_rmul+6+1))),w
    movwf    btemp+4
    movf (((?_rmul+6+2))),w
    movwf    btemp+5
    movf (((?_rmul+0))),w
    movwf    btemp
    movf (((?_rmul+0+1))),w
    movwf    btemp+1
```

```
    movf (((?_rmul+0+2))),w
    movwf   btemp+2
    global  ftmul
    fcall   ftmul
    movf f2853+0,w
    movwf   btemp+3
    movf f2853+0+1,w
    movwf   btemp+4
    movf f2853+0+2,w
    movwf   btemp+5
    global  ftadd
    fcall   ftadd
    movwf   btemp,w
    movwf   (((?a_rmul+0)))
    movwf   btemp+1,w
    movwf   (((?a_rmul+0+1)))
    movwf   btemp+2,w
    movwf   (((?a_rmul+0+2)))
;194: return ans;
    line 194
    movf (((?a_rmul+0))),w
    movwf   btemp
    movf (((?a_rmul+0+1))),w
    movwf   btemp+1
    movf (((?a_rmul+0+2))),w
    movwf   btemp+2
    goto l52
;195: }
    line 195
l52
    bcf  status,6
    bcf  status,5
    return
    FNSIZE   _rmul,6,12
    global   ?a_rmul
    global   ?_rmul
f2853    equ  ?a_rmul+3
;196: float imul(float a,float b,float c,float
;d)
;197: {
```

```
;       param _a assigned to ?_imul+0
_imul$a  set  ?_imul+0
;       param _b assigned to ?_imul+3
_imul$b  set  ?_imul+3
;       param _c assigned to ?_imul+6
_imul$c  set  ?_imul+6
;       param _d assigned to ?_imul+9
_imul$d  set  ?_imul+9
    psect    text11,local,class=CODE,delta=2
    psect    text11
    line 197
_imul
;198: float ans;
;    _ans assigned to ?a_imul+0
_imul$ans    set  ?a_imul+0
;199: ans=-(a*d)+(b*c);
    line 199
    bcf  status,5
    bcf  status,6
    movf (((?_imul+9))),w
    movwf    btemp+3
    movf (((?_imul+9+1))),w
    movwf    btemp+4
    movf (((?_imul+9+2))),w
    movwf    btemp+5
    movf (((?_imul+0))),w
    movwf    btemp
    movf (((?_imul+0+1))),w
    movwf    btemp+1
    movf (((?_imul+0+2))),w
    movwf    btemp+2
    global   ftmul
    fcall    ftmul
    movf btemp,w
    movwf    f2863+0
    movf btemp+1,w
    movwf    f2863+0+1
    movf btemp+2,w
    movwf    f2863+0+2
```

```
        movf (((?_imul+6))),w
        movwf    btemp+3
        movf (((?_imul+6+1))),w
        movwf    btemp+4
        movf (((?_imul+6+2))),w
        movwf    btemp+5
        movf (((?_imul+3))),w
        movwf    btemp
        movf (((?_imul+3+1))),w
        movwf    btemp+1
        movf (((?_imul+3+2))),w
        movwf    btemp+2
        global   ftmul
        fcall    ftmul
        movf f2863+0,w
        movwf    btemp+3
        movf f2863+0+1,w
        movwf    btemp+4
        movf f2863+0+2,w
        movwf    btemp+5
        global   ftsub
        fcall    ftsub
        movwf    btemp,w
        movwf    (((?a_imul+0)))
        movwf    btemp+1,w
        movwf    (((?a_imul+0+1)))
        movwf    btemp+2,w
        movwf    (((?a_imul+0+2)))
;200: return (ans);
        line200
        movf (((?a_imul+0))),w
        movwf    btemp
        movf (((?a_imul+0+1))),w
        movwf    btemp+1
        movf (((?a_imul+0+2))),w
        movwf    btemp+2
        goto l53
;201: }
        line201
l53
```

```
        bcf  status,6
        bcf  status,5
        return
        FNSIZE   _imul,6,12
        global   ?a_imul
        global   ?_imul
f2863   equ  ?a_imul+3
        psect    text12,local,class=CODE,delta=2
        psect    text12
_ACKDT  equ  1165
_ACKEN  equ  1164
_ACKSTAT equ 1166
_ADCS0  equ  254
_ADCS1  equ  255
_ADDEN  equ  195
_ADFM   equ  1279
_ADGO   equ  250
_ADIE   equ  1126
_ADIF   equ  102
_ADON   equ  248
_BCLIE  equ  1131
_BCLIF  equ  107
_BOR equ 1136
_BRGH   equ  1218
_CARRY  equ  24
_CCP1IE equ  1122
_CCP1IF equ  98
_CCP1M0 equ  184
_CCP1M1 equ  185
_CCP1M2 equ  186
_CCP1M3 equ  187
_CCP1X  equ  189
_CCP1Y  equ  188
_CCP2IE equ  1128
_CCP2IF equ  104
_CCP2M0 equ  232
_CCP2M1 equ  233
_CCP2M2 equ  234
_CCP2M3 equ  235
_CCP2X  equ  237
```

```
_CCP2Y   equ 236
_CHS0    equ 251
_CHS1    equ 252
_CHS2    equ 253
_CKP equ 164
_CREN    equ 196
_CSRC    equ 1223
_DC  equ 25
_EEIE    equ 1132
_EEIF    equ 108
_EEPGD   equ 3175
_FERR    equ 194
_GCEN    equ 1167
_GIE equ 95
_IBF equ 1103
_IBOV    equ 1101
_INTE    equ 92
_INTEDG  equ 1038
_INTF    equ 89
_IRP equ 31
_OBF equ 1102
_OERR    equ 193
_PCFG0   equ 1272
_PCFG1   equ 1273
_PCFG2   equ 1274
_PCFG3   equ 1275
_PD  equ 27
_PEIE    equ 94
_PEN equ 1162
_POR equ 1137
_PS0 equ 1032
_PS1 equ 1033
_PS2 equ 1034
_PSA equ 1035
_PSPIE   equ 1127
_PSPIF   equ 103
_PSPMODE equ 1100
_RA0 equ 40
_RA1 equ 41
_RA2 equ 42
```

```
_RA3 equ  43
_RA4 equ  44
_RA5 equ  45
_RB0 equ  48
_RB1 equ  49
_RB2 equ  50
_RB3 equ  51
_RB4 equ  52
_RB5 equ  53
_RB6 equ  54
_RB7 equ  55
_RBIE    equ  91
_RBIF    equ  88
_RBPU    equ  1039
_RC0 equ  56
_RC1 equ  57
_RC2 equ  58
_RC3 equ  59
_RC4 equ  60
_RC5 equ  61
_RC6 equ  62
_RC7 equ  63
_RCEN    equ  1163
_RCIE    equ  1125
_RCIF    equ  101
_RD  equ  3168
_RD0 equ  64
_RD1 equ  65
_RD2 equ  66
_RD3 equ  67
_RD4 equ  68
_RD5 equ  69
_RD6 equ  70
_RD7 equ  71
_RE0 equ  72
_RE1 equ  73
_RE2 equ  74
_RP0 equ  29
_RP1 equ  30
_RSEN    equ  1161
```

```
_RX9 equ 198
_RX9D    equ 192
_SEN equ 1160
_SPEN    equ 199
_SREN    equ 197
_SSPEN   equ 165
_SSPIE   equ 1123
_SSPIF   equ 99
_SSPM0   equ 160
_SSPM1   equ 161
_SSPM2   equ 162
_SSPM3   equ 163
_SSPOV   equ 166
_STAT_BF equ 1184
_STAT_CKE    equ 1190
_STAT_DA equ 1189
_STAT_P  equ 1188
_STAT_RW equ 1186
_STAT_S  equ 1187
_STAT_SMP    equ 1191
_STAT_UA equ 1185
_SYNC    equ 1220
_T0CS    equ 1037
_T0IE    equ 93
_T0IF    equ 90
_T0SE    equ 1036
_T1CKPS0 equ 132
_T1CKPS1 equ 133
_T1OSCEN equ 131
_T1SYNC  equ 130
_T2CKPS0 equ 144
_T2CKPS1 equ 145
_TMR1CS  equ 129
_TMR1IE  equ 1120
_TMR1IF  equ 96
_TMR1ON  equ 128
_TMR2IE  equ 1121
_TMR2IF  equ 97
_TMR2ON  equ 146
_TO  equ 28
```

```
_TOUTPS0 equ  147
_TOUTPS1 equ  148
_TOUTPS2 equ  149
_TOUTPS3 equ  150
_TRISA0  equ  1064
_TRISA1  equ  1065
_TRISA2  equ  1066
_TRISA3  equ  1067
_TRISA4  equ  1068
_TRISA5  equ  1069
_TRISB0  equ  1072
_TRISB1  equ  1073
_TRISB2  equ  1074
_TRISB3  equ  1075
_TRISB4  equ  1076
_TRISB5  equ  1077
_TRISB6  equ  1078
_TRISB7  equ  1079
_TRISC0  equ  1080
_TRISC1  equ  1081
_TRISC2  equ  1082
_TRISC3  equ  1083
_TRISC4  equ  1084
_TRISC5  equ  1085
_TRISC6  equ  1086
_TRISC7  equ  1087
_TRISD0  equ  1088
_TRISD1  equ  1089
_TRISD2  equ  1090
_TRISD3  equ  1091
_TRISD4  equ  1092
_TRISD5  equ  1093
_TRISD6  equ  1094
_TRISD7  equ  1095
_TRMT    equ  1217
_TX9 equ  1222
_TX9D    equ  1216
_TXEN    equ  1221
_TXIE    equ  1124
_TXIF    equ  100
```

```
_WCOL    equ 167
_WR equ 3169
_WREN    equ 3170
_WRERR   equ 3171
_ZERO    equ 26
    psect    rbit_0,class=BANK0,bit,space=1
    global   clear_bit0
    psect    rbit_0
_half_flag
    ds  1
_measurementcomp_flag
    ds  1
_start_flag
    ds  1
_ADCON0  equ 31
_ADCON1  equ 159
_ADRESH  equ 30
_ADRESL  equ 158
_CCP1CON equ 23
_CCP2CON equ 29
_CCPR1H  equ 22
_CCPR1L  equ 21
_CCPR2H  equ 28
_CCPR2L  equ 27
_EEADR   equ 269
_EEADRH  equ 271
_EECON1  equ 396
_EECON2  equ 397
_EEDATA  equ 268
_EEDATH  equ 270
_FSR equ 4
_INDF    equ 0
_INTCON  equ 11
_OPTION  equ 129
_PCL equ 2
_PCLATH  equ 10
_PCON    equ 142
_PIE1    equ 140
_PIE2    equ 141
_PIR1    equ 12
```

```
_PIR2    equ 13
_PORTA   equ 5
_PORTB   equ 6
_PORTC   equ 7
_PORTD   equ 8
_PORTE   equ 9
_PR2 equ 146
_RCREG   equ 26
_RCSTA   equ 24
_SPBRG   equ 153
_SSPADD  equ 147
_SSPBUF  equ 19
_SSPCON  equ 20
_SSPCON2 equ 145
_SSPSTAT equ 148
_STATUS  equ 3
_T1CON   equ 16
_T2CON   equ 18
_TMR0    equ 1
_TMR1H   equ 15
_TMR1L   equ 14
_TMR2    equ 17
_TRISA   equ 133
_TRISB   equ 134
_TRISC   equ 135
_TRISD   equ 136
_TRISE   equ 137
_TXREG   equ 25
_TXSTA   equ 152
    psect    rbss_0,class=BANK0,space=1
    global   clear_bank0
    psect    rbss_0
_keydata
    ds   1
    global   _input
_input
    ds   3
    global   _ipart
_ipart
    ds   3
```

```
    global   _output
_output
    ds   3
    global   _rpart
_rpart
    ds   3
_harm
    ds   12
_res
    ds   160
    psect    temp,ovrld,class=BANK0,space=1
btemp
    ds   8
    global   used_btemp7
    global   used_btemp6
    global   used_btemp5
    global   used_btemp4
    global   used_btemp3
    global   used_btemp2
    global   used_btemp1
    global   used_btemp0
    end
```

CHAPTER 7

CONCLUSION

CONCLUSION

FUTURE EXPANSION:

The project can be expanded in future to display more harmonics needed. The harmonics measured can be removed by designing appropriate filters. These filters can be placed in appropriate positions to remove harmonics.

CONCLUSION:

Industries desire for contemporary methods to detect harmonics and demanding ways to remove the same. This project would be very helpful in detecting a harmonic present in the power line with a reduction in cost and thus save the equipment.

The circuit was tested with a distorted input signal and the rms values were measured. Thus the project has been successfully completed to the company's requirement.

CHAPTER 8

BIBLIOGRAPHY

BIBLIOGRAPHY

BOOKS:

- ELECTRICAL MEASUREMENT AND MEASURING INSTRUMENTS
  - GOLDING & WIDDIS
- DIGITAL SIGNAL PROCESSING
  - JOHN G.PROAKIS &
  - DIMITRIS G.MANOLAKIS
- PIC 16F877 MICROCONTROLLER –MID RANGE REFERENCE MANUAL
- FAST FOURIER ALGORITHM
  - ERIC BRIGHAM
- THE C PROGRAMMING LANGUAGE
  - BRIAN W.KERNIGHAN &
  - DENNIS RITCHIE
- MICROPROCESSOR AND INTERFACING CIRCUITS
  - DOUGLAS HALL

WEBSITES:

WWW.GOOGLE.COM

WWW.NPL.COM

WWW.MICROCHIP.COM

WWW.ECE.WPI.EDU

CHAPTER 9

ANNEXURE

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

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.
$\overline{MCLR}$/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/$\overline{SS}$/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^2C modes.
RC4/SDI/SDA	15	15	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C 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 CSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/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/$\overline{SS}$/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/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/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.

2.2.2 SPECIAL FUNCTION REGISTERS

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. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
01h	TMR0	Timer0 Module Register								xxxx xxxx	47
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	29
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	33
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	35
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	36
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
0Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	22
0Dh	PIR2	—	(5)	—	EEIF	BCLIF	—	—	CCP2IF	-r-0 0--0	24
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	51
11h	TMR2	Timer2 Module Register								0000 0000	55
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	55
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	70, 73
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	67
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	57
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	57
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	58
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	96
19h	TXREG	USART Transmit Data Register								0000 0000	99
1Ah	RCREG	USART Receive Data Register								0000 0000	101
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	57
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	57
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	58
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	116
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	111

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 1											
80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
83h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	29
86h	TRISB	PORTB Data Direction Register								1111 1111	31
87h	TRISC	PORTC Data Direction Register								1111 1111	33
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	35
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	37
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
8Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	21
8Dh	PIE2	—	(5)	—	EEIE	BCLIE	—	—	CCP2IE	-r-0 0--0	23
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	25
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	68
92h	PR2	Timer2 Period Register								1111 1111	55
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	73, 74
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	66
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	95
99h	SPBRG	Baud Rate Generator Register								0000 0000	97
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	—	Unimplemented								—	—
9Dh	—	Unimplemented								—	—
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	116
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	0--- 0000	112

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 2											
100h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
101h	TMR0	Timer0 Module Register								xxxx xxxx	47
102h(3)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	26
103h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
104h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
10Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	41
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	41
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						xxxx xxxx	41
10Fh	EEADRH	—	—	—	EEPROM Address Register High Byte					xxxx xxxx	41
Bank 3											
180h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
182h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
183h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
184h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	31
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
18Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	41, 42
18Dh	EECON2	EEPROM Control Register2 (not a physical register)								---- ----	41
18Eh	—	Reserved maintain clear								0000 0000	—
18Fh	—	Reserved maintain clear								0000 0000	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

TABLE 12-6: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Devices				Power-on Reset, Brown-out Reset	$\overline{MCLR}$ Resets, WDT Reset	Wake-up via WDT or Interrupt
PIE2	873	874	876	877	-r-0 0--0	-r-0 0--0	-r-u u--u
PCON	873	874	876	877	---- --qq	---- --uu	---- --uu
PR2	873	874	876	877	1111 1111	1111 1111	1111 1111
SSPADD	873	874	876	877	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	873	874	876	877	--00 0000	--00 0000	--uu uuuu
TXSTA	873	874	876	877	0000 -010	0000 -010	uuuu -uuu
SPBRG	873	874	876	877	0000 0000	0000 0000	uuuu uuuu
ADRESL	873	874	876	877	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON1	873	874	876	877	0--- 0000	0--- 0000	u--- uuuu
EEDATA	873	874	876	877	0--- 0000	0--- 0000	u--- uuuu
EEADR	873	874	876	877	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEDATH	873	874	876	877	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEADRH	873	874	876	877	xxxx xxxx	uuuu uuuu	uuuu uuuu
EECON1	873	874	876	877	x--- x000	u--- u000	u--- uuuu
EECON2	873	874	876	877	---- ----	---- ----	---- ----

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition, r = reserved, maintain clear

Note 1: One or more bits in INTCON, PIR1 and/or PIR2 will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

3: See Table 12-5 for RESET value for specific condition.

FIGURE 12-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ TIED TO VDD)

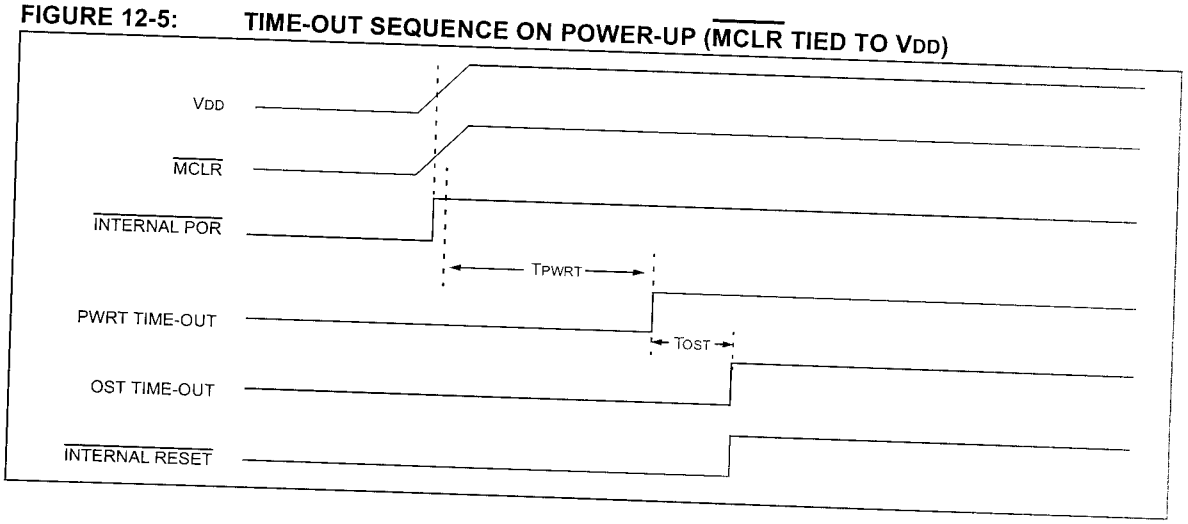


FIGURE 12-6: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1

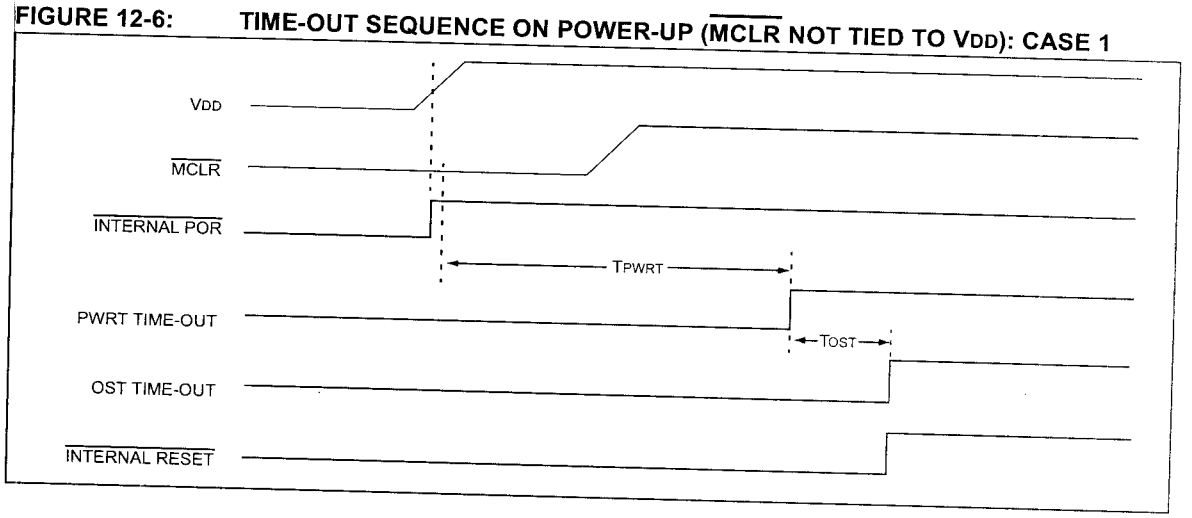


FIGURE 12-7: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2

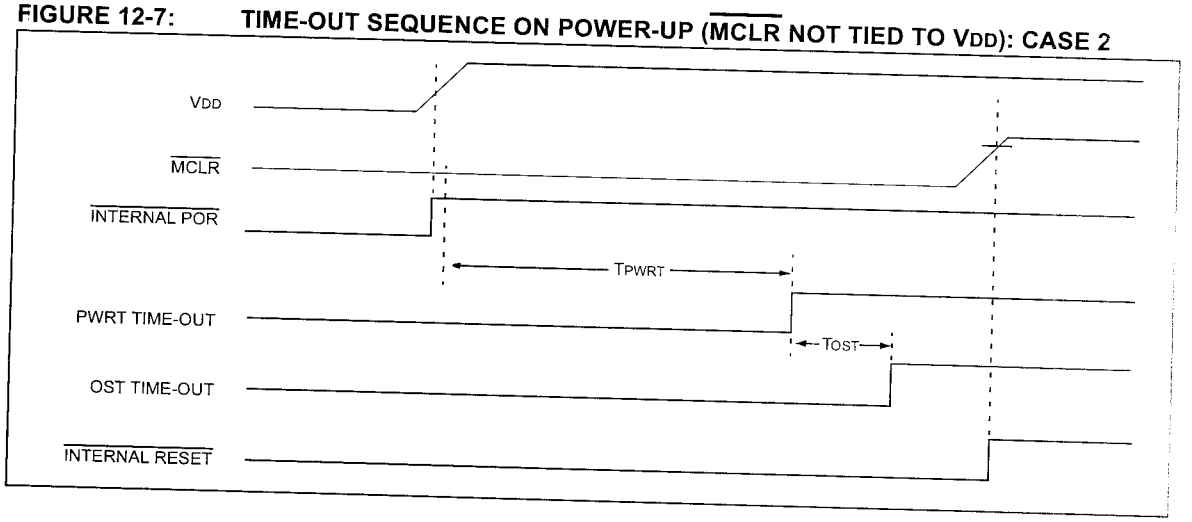
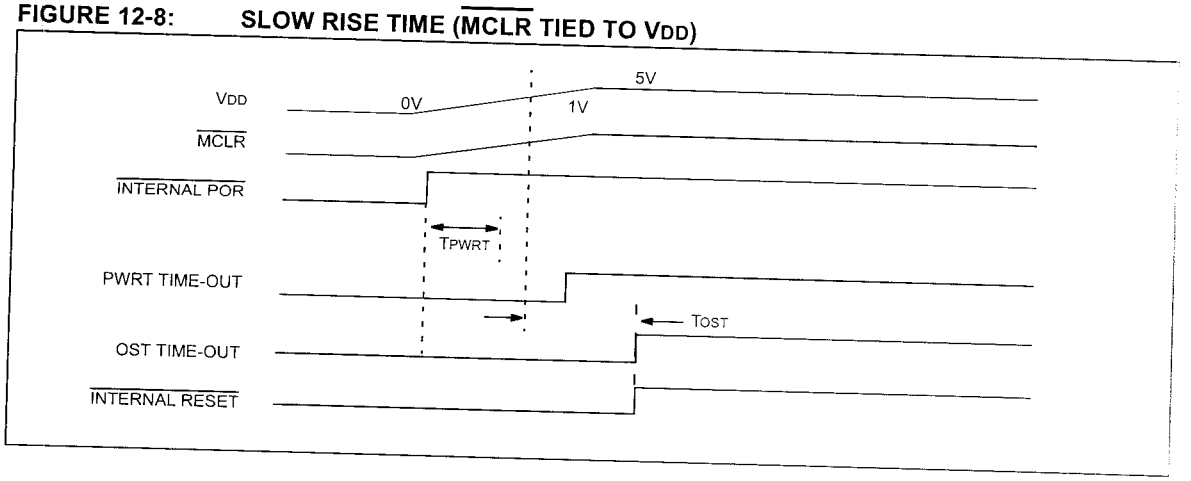


FIGURE 12-8: SLOW RISE TIME (MCLR TIED TO VDD)



12.10 Interrupts

The PIC16F87X family has up to 14 sources of interrupt. The interrupt control register (INTCON) records individual interrupt requests in flag bits. It also has individual and global interrupt enable bits.

Note:	Individual interrupt flag bits are set, regardless of the status of their corresponding mask bit, or the GIE bit.

A global interrupt enable bit, GIE (INTCON<7>) enables (if set) all unmasked interrupts, or disables (if cleared) all interrupts. When bit GIE is enabled, and an interrupt's flag bit and mask bit are set, the interrupt will vector immediately. Individual interrupts can be disabled through their corresponding enable bits in various registers. Individual interrupt bits are set, regardless of the status of the GIE bit. The GIE bit is cleared on RESET.

The "return from interrupt" instruction, `RETFIE`, exits the interrupt routine, as well as sets the GIE bit, which re-enables interrupts.

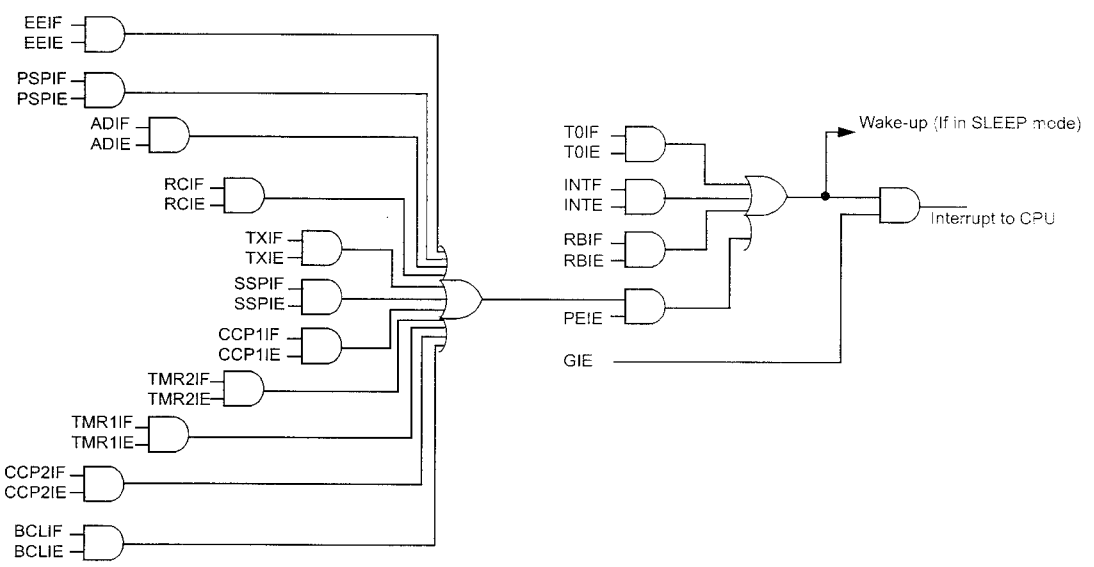
The RB0/INT pin interrupt, the RB port change interrupt, and the TMR0 overflow interrupt flags are contained in the INTCON register.

The peripheral interrupt flags are contained in the special function registers, PIR1 and PIR2. The corresponding interrupt enable bits are contained in special function registers, PIE1 and PIE2, and the peripheral interrupt enable bit is contained in special function register INTCON.

When an interrupt is responded to, the GIE bit is cleared to disable any further interrupt, the return address is pushed onto the stack and the PC is loaded with 0004h. Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

For external interrupt events, such as the INT pin or PORTB change interrupt, the interrupt latency will be three or four instruction cycles. The exact latency depends when the interrupt event occurs. The latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set, regardless of the status of their corresponding mask bit, PEIE bit, or GIE bit.

FIGURE 12-9: INTERRUPT LOGIC



The following table shows which devices have which interrupts.

Device	T0IF	INTF	RBIF	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	EEIF	BCLIF	CCP2IF
PIC16F876/873	Yes	Yes	Yes	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PIC16F877/874	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

TABLE 13-2: PIC16F87X INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

Note: Additional information on the mid-range instruction set is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

15.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

Ambient temperature under bias ..... -55 to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any pin with respect to V_{SS} (except V_{DD}, $\overline{MCLR}$, and RA4) ..... -0.3 V to (V_{DD} + 0.3 V)
Voltage on V_{DD} with respect to V_{SS} ..... -0.3 to +7.5 V
Voltage on $\overline{MCLR}$ with respect to V_{SS} **(Note 2)** ..... 0 to +14 V
Voltage on RA4 with respect to Vss ..... 0 to +8.5 V
Total power dissipation **(Note 1)** ..... 1.0 W
Maximum current out of V_{SS} pin ..... 300 mA
Maximum current into V_{DD} pin ..... 250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ± 20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD}) ..... ± 20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 25 mA
Maximum current sunk by PORTA, PORTB, and PORTE (combined) **(Note 3)** ..... 200 mA
Maximum current sourced by PORTA, PORTB, and PORTE (combined) **(Note 3)** ..... 200 mA
Maximum current sunk by PORTC and PORTD (combined) **(Note 3)** ..... 200 mA
Maximum current sourced by PORTC and PORTD (combined) **(Note 3)** ..... 200 mA

Note 1: Power dissipation is calculated as follows: $P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

2: Voltage spikes below V_{SS} at the $\overline{MCLR}$ pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$ pin, rather than pulling this pin directly to V_{SS}.

3: PORTD and PORTE are not implemented on PIC16F873/876 devices.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

FIGURE 15-6: EXTERNAL CLOCK TIMING

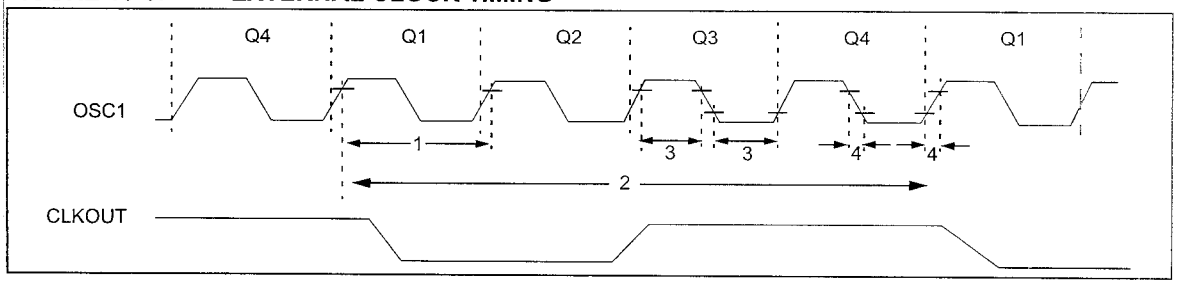


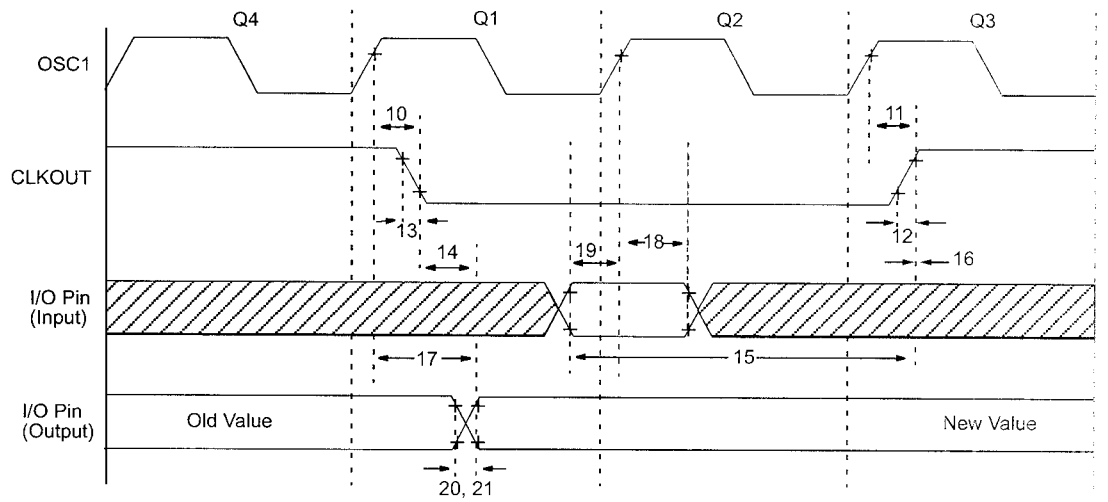
TABLE 15-1: EXTERNAL CLOCK TIMING REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	FOSC	External CLKIN Frequency **(Note 1)**	DC	—	4	MHz	XT and RC osc mode
			DC	—	4	MHz	HS osc mode (-04)
			DC	—	10	MHz	HS osc mode (-10)
			DC	—	20	MHz	HS osc mode (-20)
			DC	—	200	kHz	LP osc mode
		Oscillator Frequency **(Note 1)**	DC	—	4	MHz	RC osc mode
			0.1	—	4	MHz	XT osc mode
			4	—	10	MHz	HS osc mode (-10)
			4	—	20	MHz	HS osc mode (-20)
			5	—	200	kHz	LP osc mode
1	TOSC	External CLKIN Period **(Note 1)**	250	—	—	ns	XT and RC osc mode
			250	—	—	ns	HS osc mode (-04)
			100	—	—	ns	HS osc mode (-10)
			50	—	—	ns	HS osc mode (-20)
			5	—	—	µs	LP osc mode
		Oscillator Period **(Note 1)**	250	—	—	ns	RC osc mode
			250	—	10,000	ns	XT osc mode
			250	—	—	ns	HS osc mode (-04)
			100	—	250	ns	HS osc mode (-10)
			50	—	250	ns	HS osc mode (-20)
			5	—	—	µs	LP osc mode
2	TCY	Instruction Cycle Time **(Note 1)**	200	TCY	DC	ns	TCY = 4/FOSC
3	TosL, TosH	External Clock in (OSC1) High or Low Time	100	—	—	ns	XT oscillator
			2.5	—	—	µs	LP oscillator
			15	—	—	ns	HS oscillator
4	TosR, TosF	External Clock in (OSC1) Rise or Fall Time	—	—	25	ns	XT oscillator
			—	—	50	ns	LP oscillator
			—	—	15	ns	HS oscillator

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time-base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions, with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min." values with an external clock applied to the OSC1/CLKIN pin. When an external clock input is used, the "max." cycle time limit is "DC" (no clock) for all devices.

FIGURE 15-7: CLKOUT AND I/O TIMING



Note: Refer to Figure 15-5 for load conditions.

TABLE 15-2: CLKOUT AND I/O TIMING REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
10*	TosH2ckL	OSC1↑ to CLKOUT↓		—	75	200	ns	(Note 1)
11*	TosH2ckH	OSC1↑ to CLKOUT↑		—	75	200	ns	(Note 1)
12*	TckR	CLKOUT rise time		—	35	100	ns	(Note 1)
13*	TckF	CLKOUT fall time		—	35	100	ns	(Note 1)
14*	TckL2ioV	CLKOUT ↓ to Port out valid		—	—	0.5TCY + 20	ns	(Note 1)
15*	TioV2ckH	Port in valid before CLKOUT ↑		TOSC + 200	—	—	ns	(Note 1)
16*	TckH2iol	Port in hold after CLKOUT ↑		0	—	—	ns	(Note 1)
17*	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid		—	100	255	ns	
18*	TosH2iol	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	Standard (F)	100	—	—	ns	
			Extended (LF)	200	—	—	ns	
19*	TioV2osH	Port input valid to OSC1↑ (I/O in setup time)		0	—	—	ns	
20*	TioR	Port output rise time	Standard (F)	—	10	40	ns	
			Extended (LF)	—	—	145	ns	
21*	TioF	Port output fall time	Standard (F)	—	10	40	ns	
			Extended (LF)	—	—	145	ns	
22††*	Tinp	INT pin high or low time		TCY	—	—	ns	
23††*	Trbp	RB7:RB4 change INT high or low time		TCY	—	—	ns	

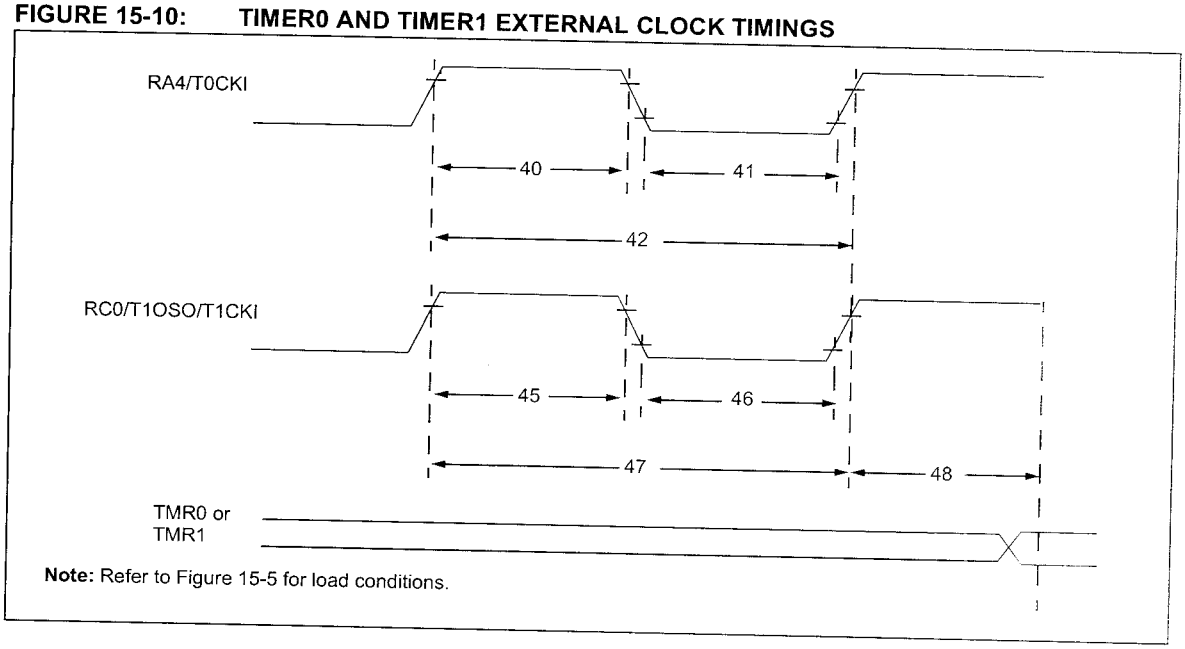
* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

†† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in RC mode where CLKOUT output is 4 x TOSC.

FIGURE 15-10: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS



Note: Refer to Figure 15-5 for load conditions.

TABLE 15-4: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Param No.	Symbol	Characteristic			Min	Typ†	Max	Units	Conditions
40*	Tt0H	T0CKI High Pulse Width		No Prescaler	0.5TCY + 20	—	—	ns	Must also meet parameter 42
				With Prescaler	10	—	—	ns	
41*	Tt0L	T0CKI Low Pulse Width		No Prescaler	0.5TCY + 20	—	—	ns	Must also meet parameter 42
				With Prescaler	10	—	—	ns	
42*	Tt0P	T0CKI Period		No Prescaler	TCY + 40	—	—	ns	
				With Prescaler	Greater of: 20 or TCY + 40 / N	—	—	ns	N = prescale value (2, 4...., 256)
45*	Tt1H	T1CKI High Time	Synchronous, Prescaler = 1		0.5TCY + 20	—	—	ns	Must also meet parameter 47
			Synchronous, Prescaler = 2,4,8	Standard(F)	15	—	—	ns	
				Extended(LF)	25	—	—	ns	
			Asynchronous	Standard(F)	30	—	—	ns	
				Extended(LF)	50	—	—	ns	
46*	Tt1L	T1CKI Low Time	Synchronous, Prescaler = 1		0.5TCY + 20	—	—	ns	Must also meet parameter 47
			Synchronous, Prescaler = 2,4,8	Standard(F)	15	—	—	ns	
				Extended(LF)	25	—	—	ns	
			Asynchronous	Standard(F)	30	—	—	ns	
				Extended(LF)	50	—	—	ns	
47*	Tt1P	T1CKI input period	Synchronous	Standard(F)	Greater of: 30 OR TCY + 40 / N	—	—	ns	N = prescale value (1, 2, 4, 8)
				Extended(LF)	Greater of: 50 OR TCY + 40 / N				N = prescale value (1, 2, 4, 8)
			Asynchronous	Standard(F)	60	—	—	ns	
				Extended(LF)	100	—	—	ns	
	Ft1	Timer1 oscillator input frequency range (oscillator enabled by setting bit T1OSCEN)			DC	—	200	kHz	
48	TCKEZtmr1	Delay from external clock edge to timer increment			2TOSC	—	7TOSC	—	

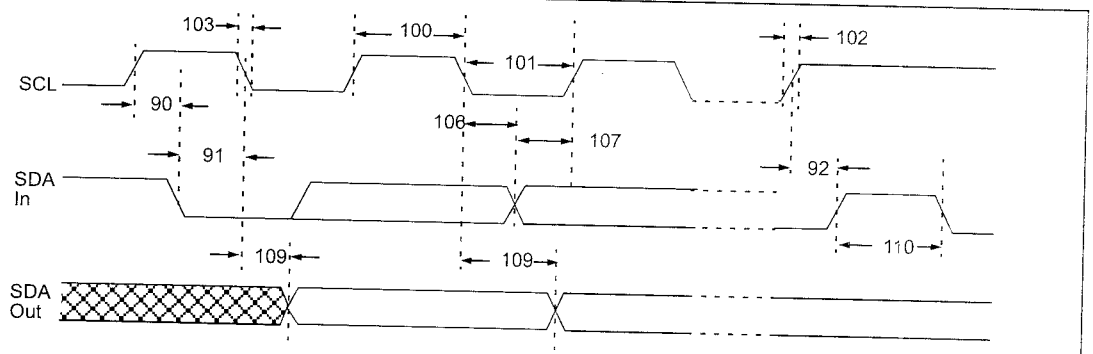
* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

TABLE 15-8: I²C BUS START/STOP BITS REQUIREMENTS

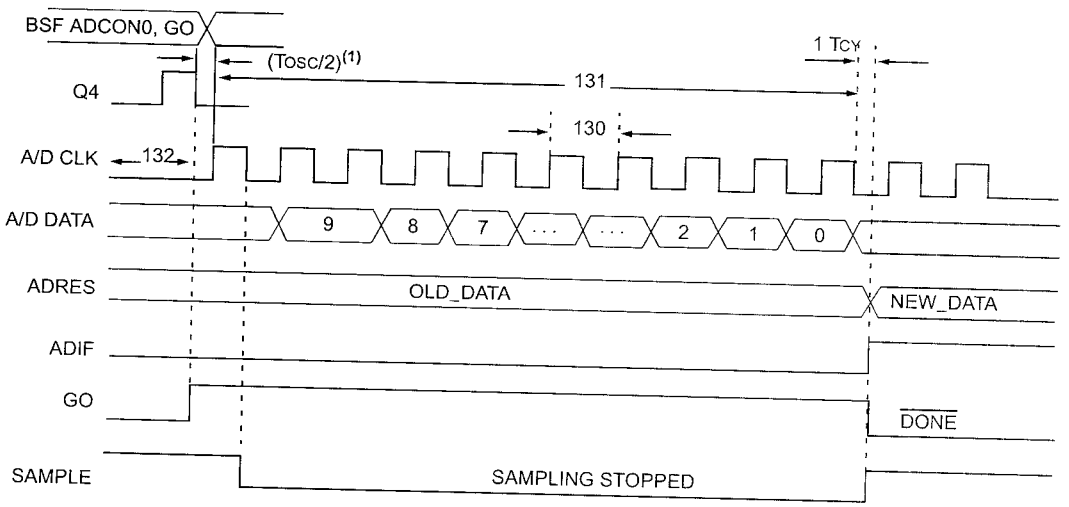
Parameter No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
90	Tsu:sta	START condition Setup time	100 kHz mode	4700	—	—	ns	Only relevant for Repeated START condition
			400 kHz mode	600	—	—		
91	Thd:sta	START condition Hold time	100 kHz mode	4000	—	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	600	—	—		
92	Tsu:sto	STOP condition Setup time	100 kHz mode	4700	—	—	ns	
			400 kHz mode	600	—	—		
93	Thd:sto	STOP condition Hold time	100 kHz mode	4000	—	—	ns	
			400 kHz mode	600	—	—		

FIGURE 15-18: I²C BUS DATA TIMING



Note: Refer to Figure 15-5 for load conditions.

FIGURE 15-21: A/D CONVERSION TIMING



Note: If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

TABLE 15-13: A/D CONVERSION REQUIREMENTS

Param No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
130	TAD	A/D clock period	Standard(F)	1.6	—	—	µs	TOSC based, VREF ≥ 3.0V
			Extended(LF)	3.0	—	—	µs	TOSC based, VREF ≥ 2.0V
			Standard(F)	2.0	4.0	6.0	µs	A/D RC mode
			Extended(LF)	3.0	6.0	9.0	µs	A/D RC mode
131	TCNV	Conversion time (not including S/H time) **(Note 1)**			—	12	TAD	
132	TACQ	Acquisition time		**(Note 2)**	40	—	µs	
				10*	—	—	µs	The minimum time is the amplifier settling time. This may be used if the "new" input voltage has not changed by more than 1 LSb (i.e., 20.0 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD).
134	TGO	Q4 to A/D clock start		—	TOSC/2 §	—	—	If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification ensured by design.

Note 1: ADRES register may be read on the following TCY cycle.

2: See Section 11.1 for minimum conditions.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter[2]		Value	Unit
V_I	DC Input Voltage	for V_O= 5 to 18V	35	V
		for V_O= 20, 24V	40	
I_O	Output Current		Internally Limited	
P_{tot}	Power Dissipation		Internally Limited	
T_{stg}	Storage Temperature Range		-65 to 150	°C
T_{op}	Operating Junction Temperature Range	for L7800	-55 to 150	°C
		for L7800C	0 to 150	

Absolute Maximum Ratings are those values beyond which damage to the device may occur. Functional operation under these condition is not implied.

THERMAL DATA

Symbol	Parameter	D^2PAK	TO-220	TO-220FP	TO-3	Unit
$R_{thj\text{-}case}$	Thermal Resistance Junction-case Max	3	5	5	4	°C/W
$R_{thj\text{-}amb}$	Thermal Resistance Junction-ambient Max	62.5	50	60	35	°C/W

SCHEMATIC DIAGRAM

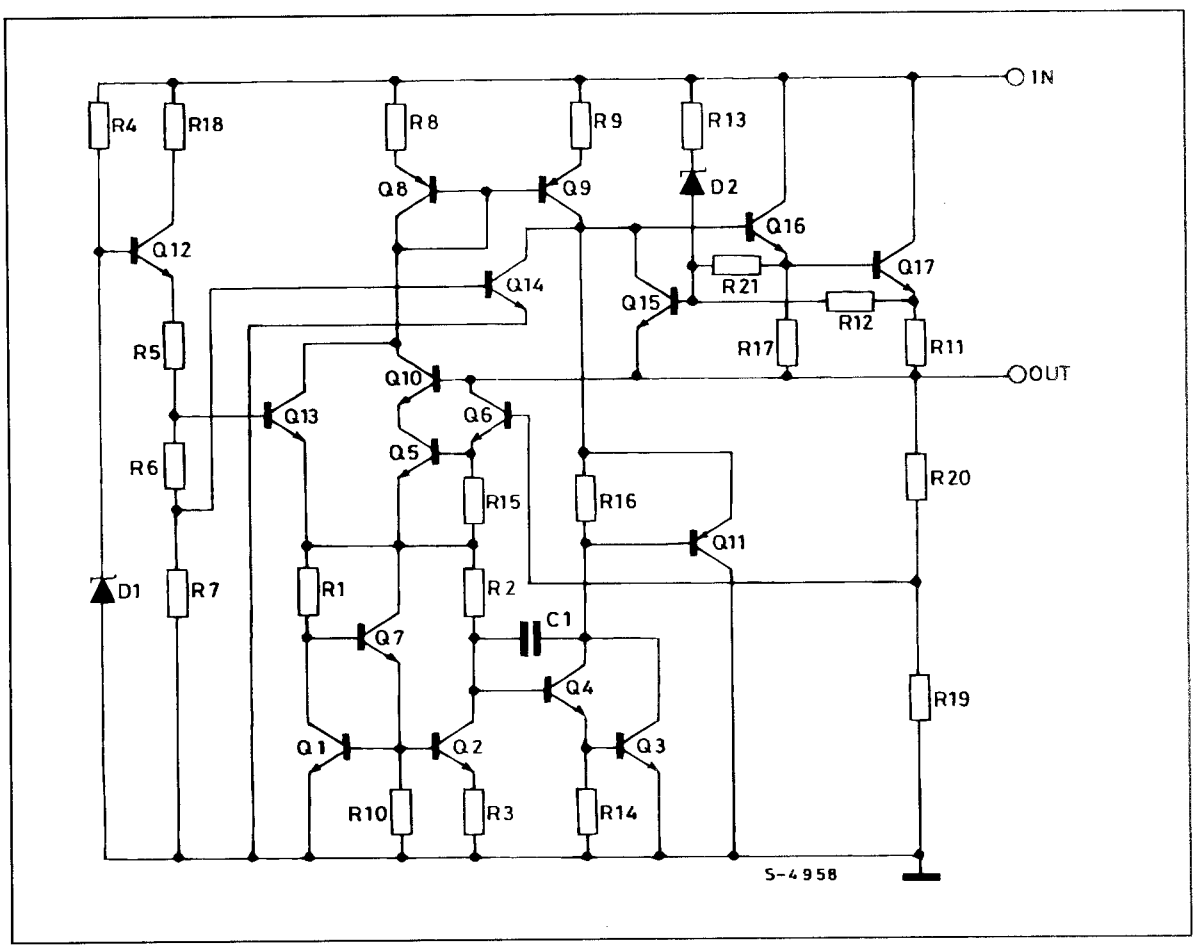
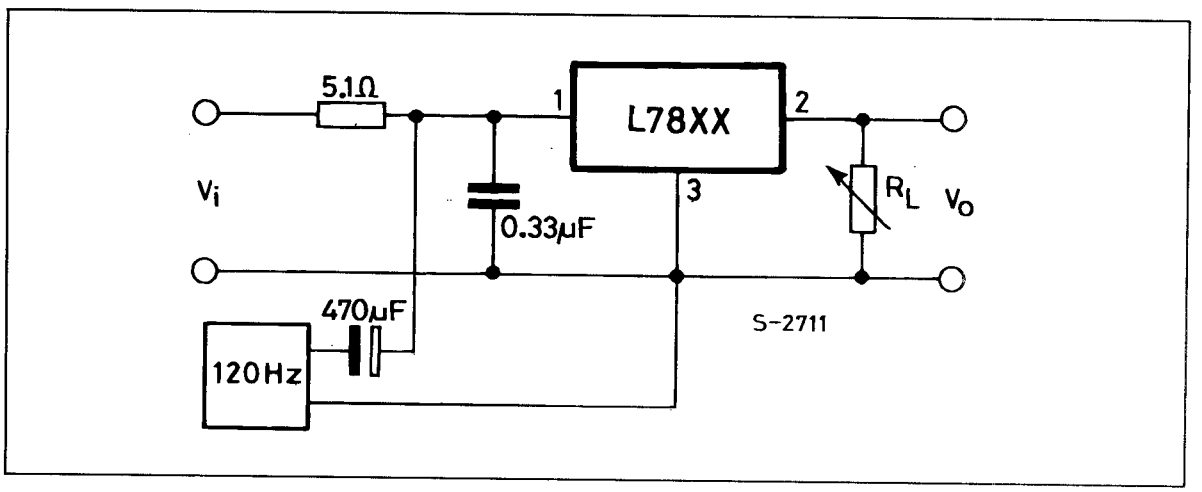


Figure 3 : Ripple Rejection



ELECTRICAL CHARACTERISTICS OF L7805 (refer to the test circuits, T_J = -55 to 150°C, V_I = 10V, I_O = 500 mA, C_I = 0.33 µF, C_O = 0.1 µF unless otherwise specified).

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_O	Output Voltage	T_J = 25°C	4.8	5	5.2	V
V_O	Output Voltage	I_O = 5 mA to 1 A $P_O \leq$ 15W V_I = 8 to 20 V	4.65	5	5.35	V
ΔV_O(*)	Line Regulation	V_I = 7 to 25 V T_J = 25°C		3	50	mV
		V_I = 8 to 12 V T_J = 25°C		1	25	
ΔV_O(*)	Load Regulation	I_O = 5 mA to 1.5 A T_J = 25°C			100	mV
		I_O = 250 to 750 mA T_J = 25°C			25	
I_d	Quiescent Current	T_J = 25°C			6	mA
ΔI_d	Quiescent Current Change	I_O = 5 mA to 1 A			0.5	mA
		V_I = 8 to 25 V			0.8	
$\Delta V_O/\Delta T$	Output Voltage Drift	I_O = 5 mA		0.6		mV/°C
eN	Output Noise Voltage	B =10Hz to 100KHz T_J = 25°C			40	µV/V_O
SVR	Supply Voltage Rejection	V_I = 8 to 18 V f = 120Hz	68			dB
V_d	Dropout Voltage	I_O = 1 A T_J = 25°C		2	2.5	V
R_O	Output Resistance	f = 1 KHz		17		mΩ
I_{sc}	Short Circuit Current	V_I = 35 V T_J = 25°C		0.75	1.2	A
I_{scp}	Short Circuit Peak Current	T_J = 25°C	1.3	2.2	3.3	A

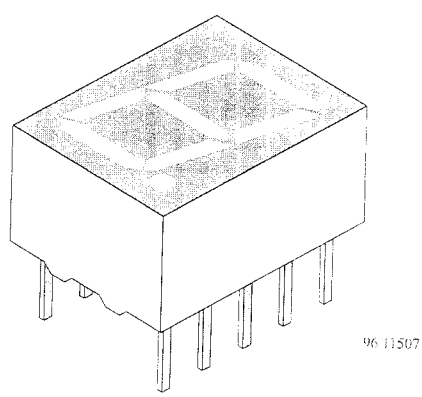
(*) Load and line regulation are specified at constant junction temperature. Changes in V_O due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.

Standard 7– Segment Display 10 mm

Color	Type	Circuitry
Red	TDSR315.	Common anode
Red	TDSR316.	Common cathode
Orange red	TDSO315.	Common anode
Orange red	TDSO316.	Common cathode
Yellow	TDSY315.	Common anode
Yellow	TDSY316.	Common cathode
Green	TDSG315.	Common anode
Green	TDSG316.	Common cathode

Description

The TDS.31.. series are 10 mm character seven segment LED displays in a very compact package.
The displays are designed for a viewing distance up to 6 meters and available in four bright colors. The grey package surface and the evenly lighted untinted segments provide an optimum on-off contrast.
All displays are categorized in luminous intensity groups. That allows users to assemble displays with uniform appearence.
Typical applications include instruments, panel meters, point-of-sale terminals and household equipment.



Features

- Evenly lighted segments
- Grey package surface
- Untinted segments
- Luminous intensity categorized
- Yellow and green categorized for color
- Wide viewing angle
- Suitable for DC and high peak current

Applications

Panel meters
Test- and measure- equipment
Point-of-sale terminals
Control units

Absolute Maximum Ratings

T_{amb} = 25°C, unless otherwise specified

TDSR315. /TDSR316. , TDSO315. /TDSO316. , TDSY315. /TDSY316. , TDSG315. /TDSG316. , /

Parameter	Test Conditions	Type	Symbol	Value	Unit
Reverse voltage per segment or DP			V_R	6	V
DC forward current per segment or DP		TDSR315./316.	I_F	30	mA
		TDSO315./316.	I_F	20	mA
		TDSY315./316.	I_F	20	mA
		TDSG315./316.	I_F	20	mA
Surge forward current per segment or DP	$t_p \leq 10$ µs (non repetitive)	TDSR315./316.	I_{FSM}	0.5	A
		TDSO315./316.	I_{FSM}	0.15	A
		TDSY315./316.	I_{FSM}	0.15	A
		TDSG315./316.	I_{FSM}	0.15	A
Power dissipation	$T_{amb} \leq 45$°C		P_V	480	mW
Junction temperature			T_j	100	°C
Operating temperature range			T_{amb}	–40 to + 85	°C
Storage temperature range			T_{stg}	–40 to + 85	°C
Soldering temperature	t ≤ 3 sec, 2mm below seating plane		T_{sd}	260	°C
Thermal resistance LED junction/ambient			R_{thJA}	120	K/W

Optical and Electrical Characteristics

T_{amb} = 25°C, unless otherwise specified

Red (TDSR315. , TDSR316.)

Parameter	Test Conditions	Type	Symbol	Min	Typ	Max	Unit
Luminous intensity per segment (digit average) [1]	I_F = 10 mA	TDSR 3150/3160	I_V	180			µcd
Dominant wavelength	I_F = 10 mA		λ_d		655		nm
Peak wavelength	I_F = 10 mA		λ_p		660		nm
Angle of half intensity	I_F = 10 mA		φ		±50		deg
Forward voltage per segment or DP	I_F = 20 mA		V_F		1.6	2	V
Reverse voltage per segment or DP	I_R = 10 µA		V_R	6	15		V
[1] I_{Vmin} and I_V groups are mean values of segments a to g							

Orange red (TDSO315., TDSO316.)

Parameter	Test Conditions	Type	Symbol	Min	Typ	Max	Unit
Luminous intensity per segment (digit average) [1]	I_F = 10 mA	TDSO 3150/3160	I_V	450			µcd
Dominant wavelength	I_F = 10 mA		λ_d	612		625	nm
Peak wavelength	I_F = 10 mA		λ_p		630		nm
Angle of half intensity	I_F = 10 mA		φ		±50		deg
Forward voltage per segment or DP	I_F = 20 mA		V_F		2	3	V
Reverse voltage per segment or DP	I_R = 10 µA		V_R	6	15		V
[1] I_{Vmin} and I_V groups are mean	values of segments a to g						

Yellow (TDSY315., TDSY316.)

Parameter	Test Conditions	Type	Symbol	Min	Typ	Max	Unit
Luminous intensity per segment (digit average) [1]	I_F = 10 mA	TDSY 3150/3160	I_V	450			µcd
Dominant wavelength	I_F = 10 mA		λ_d	581		594	nm
Peak wavelength	I_F = 10 mA		λ_p		585		nm
Angle of half intensity	I_F = 10 mA		φ		±50		deg
Forward voltage per segment or DP	I_F = 20 mA		V_F		2.4	3	V
Reverse voltage per segment or DP	I_R = 10 µA		V_R	6	15		V
[1] I_{Vmin} and I_V groups are mean	values of segments a to g						

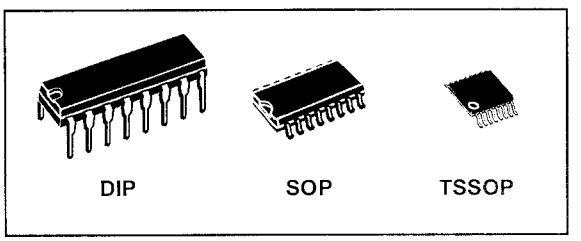
Green (TDSG315., TDSG316.)

Parameter	Test Conditions	Type	Symbol	Min	Typ	Max	Unit
Luminous intensity per segment (digit average) [1]	I_F = 10 mA	TDSG 3150/3160	I_V	450			µcd
Dominant wavelength	I_F = 10 mA		λ_d	562		575	nm
Peak wavelength	I_F = 10 mA		λ_p		565		nm
Angle of half intensity	I_F = 10 mA		φ		±50		deg
Forward voltage per segment or DP	I_F = 20 mA		V_F		2.4	3	V
Reverse voltage per segment or DP	I_R = 10 µA		V_R	6	15		V
[1] I_{Vmin} and I_V groups are mean	values of segments a to g						

M74HC138

3 TO 8 LINE DECODER (INVERTING)

- HIGH SPEED:
  t_{PD} = 13ns (TYP.) at V_{CC} = 6V
- LOW POWER DISSIPATION:
  I_{CC} = 4μA(MAX.) at T_A=25°C
- HIGH NOISE IMMUNITY:
  $V_{NIH} = V_{NIL}$ = 28 % V_{CC} (MIN.)
- SYMMETRICAL OUTPUT IMPEDANCE:
  $|I_{OH}| = I_{OL}$ = 4mA (MIN)
- BALANCED PROPAGATION DELAYS:
  $t_{PLH} \cong t_{PHL}$
- WIDE OPERATING VOLTAGE RANGE:
  V_{CC} (OPR) = 2V to 6V
- PIN AND FUNCTION COMPATIBLE WITH 74 SERIES 138



ORDER CODES

PACKAGE	TUBE	T & R
DIP	M74HC138B1R	
SOP	M74HC138M1R	M74HC138RM13TR
TSSOP		M74HC138TTR

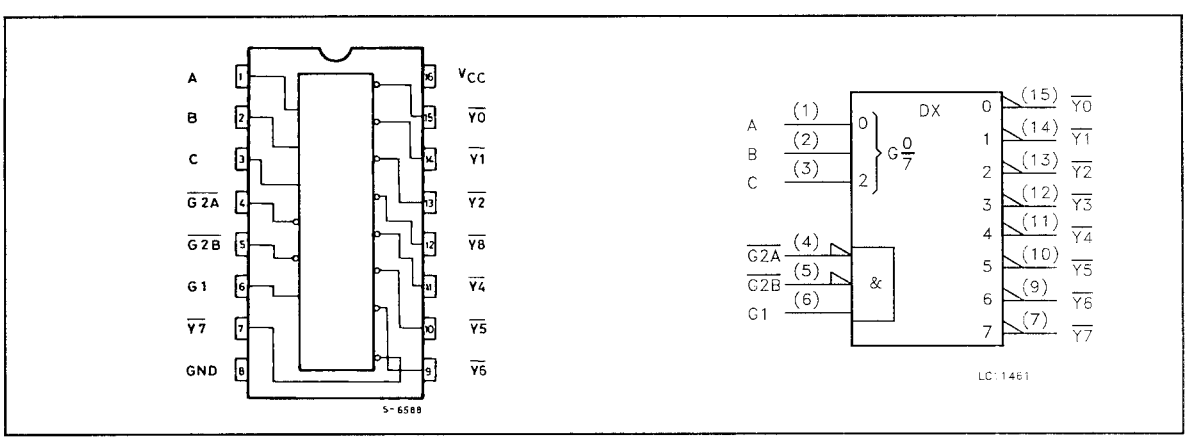
DESCRIPTION

The M74HC138 is an high speed CMOS 3 TO 8 LINE DECODER fabricated with silicon gate C^2MOS technology.

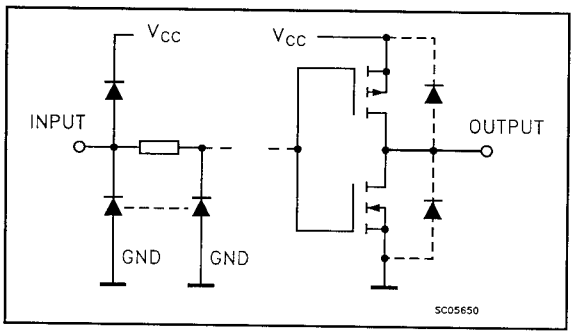
If the device is enabled, 3 binary select inputs (A, B, and C) determine which one of the outputs will go low. If enable input G1 is held low or either $\overline{G2A}$ or $\overline{G2B}$ is held high, the decoding function is inhibited and all the 8 outputs go high. Three enable inputs are provided to ease cascade connection and application of address decoders for memory systems.

All inputs are equipped with protection circuits against static discharge and transient excess voltage.

PIN CONNECTION AND IEC LOGIC SYMBOLS



INPUT AND OUTPUT EQUIVALENT CIRCUIT



PIN DESCRIPTION

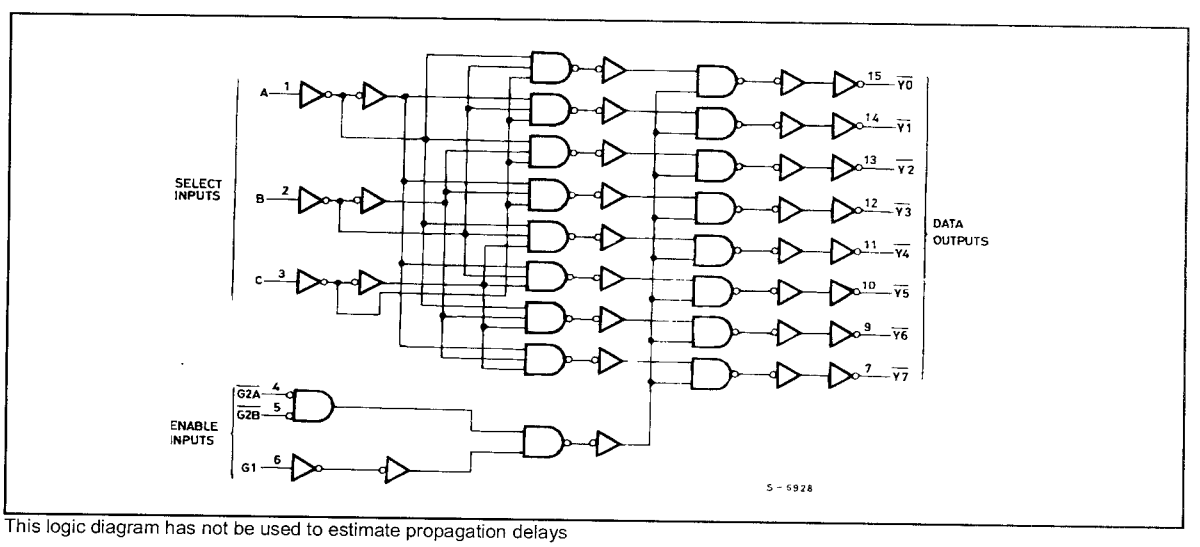
PIN No	SYMBOL	NAME AND FUNCTION
1, 2, 3	A, B, C	Address Inputs
4, 5	$\overline{G2A}$, $\overline{G2B}$	Enable Inputs
6	G1	Enable Input
9, 10, 11, 12, 13, 14, 15, 7	$\overline{Y0}$ to $\overline{Y7}$	Data Outputs
8	GND	Ground (0V)
16	V_{CC}	Positive Supply Voltage

TRUTH TABLE

INPUTS						OUTPUTS							
ENABLE			SELECT										
$\overline{G2B}$	$\overline{G2A}$	G1	C	B	A	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$	$\overline{Y4}$	$\overline{Y5}$	$\overline{Y6}$	$\overline{Y7}$
X	X	L	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
H	X	X	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

X : Don't Care

LOGIC DIAGRAM



This logic diagram has not be used to estimate propagation delays

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	-0.5 to +7	V
V_I	DC Input Voltage	-0.5 to V_{CC} + 0.5	V
V_O	DC Output Voltage	-0.5 to V_{CC} + 0.5	V
I_{IK}	DC Input Diode Current	± 20	mA
I_{OK}	DC Output Diode Current	± 20	mA
I_O	DC Output Current	± 25	mA
I_{CC} or I_{GND}	DC V_{CC} or Ground Current	± 50	mA
P_D	Power Dissipation	500(*)	mW
T_{stg}	Storage Temperature	-65 to +150	°C
T_L	Lead Temperature (10 sec)	300	°C

Absolute Maximum Ratings are those values beyond which damage to the device may occur. Functional operation under these conditions is not implied

(*) 500mW at 65 °C; derate to 300mW by 10mW/°C from 65°C to 85°C

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter		Value	Unit
V_{CC}	Supply Voltage		2 to 6	V
V_I	Input Voltage		0 to V_{CC}	V
V_O	Output Voltage		0 to V_{CC}	V
T_{op}	Operating Temperature		-55 to 125	°C
t_r, t_f	Input Rise and Fall Time	V_{CC} = 2.0V	0 to 1000	ns
		V_{CC} = 4.5V	0 to 500	ns
		V_{CC} = 6.0V	0 to 400	ns

DC SPECIFICATIONS

Symbol	Parameter	Test Condition		Value							Unit
		V_{CC} (V)		T_A = 25°C			-40 to 85°C		-55 to 125°C		
				Min.	Typ.	Max.	Min.	Max.	Min.	Max.	
V_{IH}	High Level Input Voltage	2.0		1.5			1.5		1.5		V
		4.5		3.15			3.15		3.15		
		6.0		4.2			4.2		4.2		
V_{IL}	Low Level Input Voltage	2.0				0.5		0.5		0.5	V
		4.5				1.35		1.35		1.35	
		6.0				1.8		1.8		1.8	
V_{OH}	High Level Output Voltage	2.0	I_O=-20 µA	1.9	2.0		1.9		1.9		V
		4.5	I_O=-20 µA	4.4	4.5		4.4		4.4		
		6.0	I_O=-20 µA	5.9	6.0		5.9		5.9		
		4.5	I_O=-4.0 mA	4.18	4.31		4.13		4.10		
		6.0	I_O=-5.2 mA	5.68	5.8		5.63		5.60		
V_{OL}	Low Level Output Voltage	2.0	I_O=20 µA		0.0	0.1		0.1		0.1	V
		4.5	I_O=20 µA		0.0	0.1		0.1		0.1	
		6.0	I_O=20 µA		0.0	0.1		0.1		0.1	
		4.5	I_O=4.0 mA		0.17	0.26		0.33		0.40	
		6.0	I_O=5.2 mA		0.18	0.26		0.33		0.40	
I_I	Input Leakage Current	6.0	V_I = V_{CC} or GND			± 0.1		± 1		± 1	µA
I_{CC}	Quiescent Supply Current	6.0	V_I = V_{CC} or GND			4		40		80	µA

AC ELECTRICAL CHARACTERISTICS (C_L = 50 pF, Input t_r = t_f = 6ns)

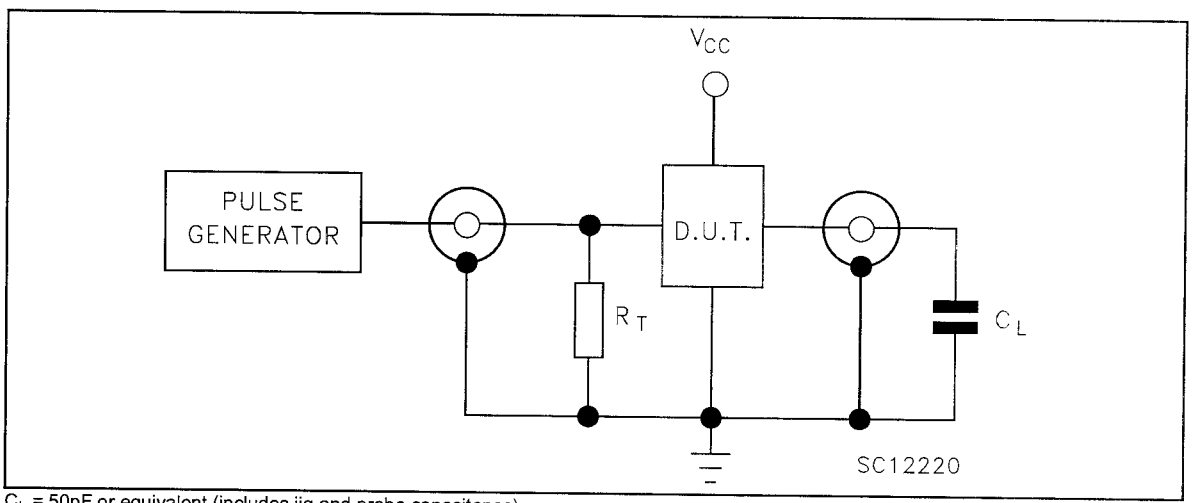
Symbol	Parameter	Test Condition		Value							Unit
		V_{CC} (V)		T_A = 25°C			-40 to 85°C		-55 to 125°C		
				Min.	Typ.	Max.	Min.	Max.	Min.	Max.	
t_{TLH} t_{THL}	Output Transition Time	2.0			30	75		95		110	ns
		4.5			8	15		19		22	
		6.0			7	13		16		19	
t_{PLH} t_{PHL}	Propagation Delay Time (A, B, C - $\overline{Y}$)	2.0			60	125		155		190	ns
		4.5			15	25		31		38	
		6.0			13	21		26		32	
t_{PLH} t_{PHL}	Propagation Delay Time ($\overline{G}$, $\overline{G}$ - $\overline{Y}$)	2.0			56	120		150		180	ns
		4.5			14	24		30		36	
		6.0			12	20		26		31	

CAPACITIVE CHARACTERISTICS

Symbol	Parameter	Test Condition		Value							Unit
		V_{CC} (V)		T_A = 25°C			-40 to 85°C		-55 to 125°C		
				Min.	Typ.	Max.	Min.	Max.	Min.	Max.	
C_{IN}	Input Capacitance	5.0			5	10		10		10	pF
C_{PD}	Power Dissipation Capacitance (note 1)	5.0			47						pF

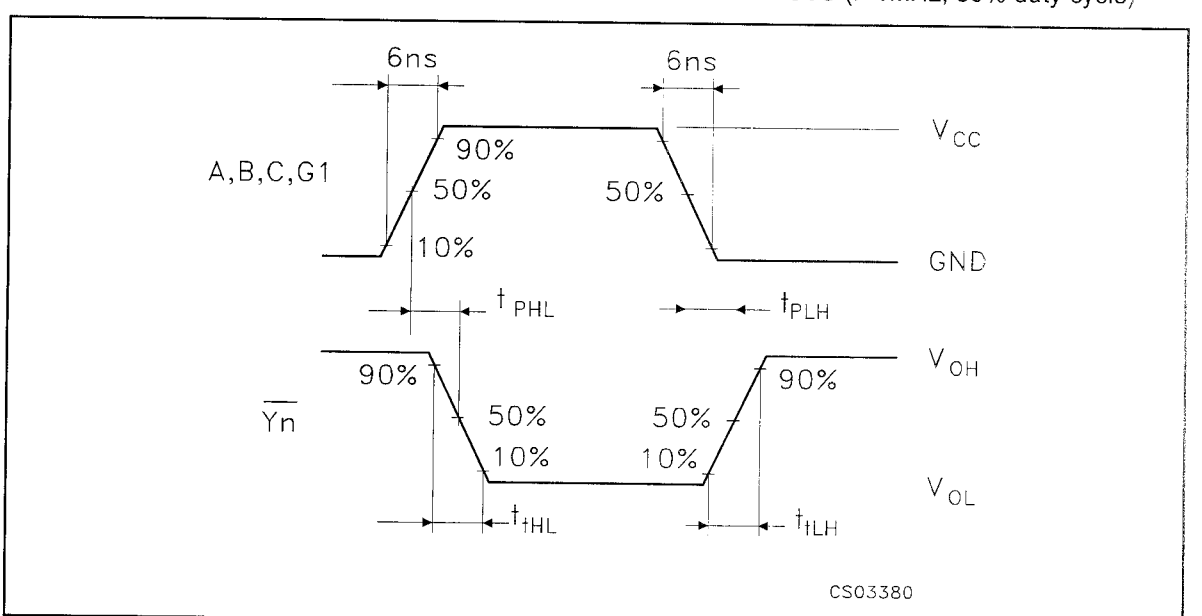
1) C_{PD} is defined as the value of the IC's internal equivalent capacitance which is calculated from the operating current consumption without load. (Refer to Test Circuit). Average operating current can be obtained by the following equation. $I_{CC(opr)} = C_{PD} \times V_{CC} \times f_{IN} + I_{CC}$

TEST CIRCUIT

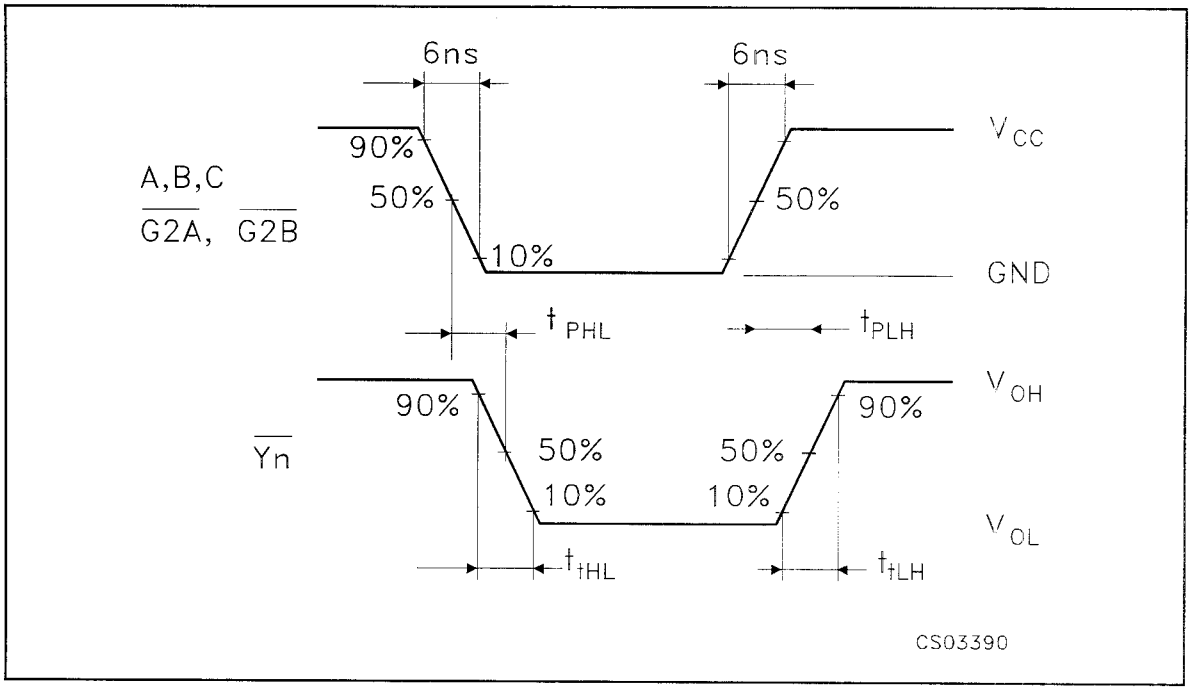


C_L = 50pF or equivalent (includes jig and probe capacitance)
$R_T = Z_{OUT}$ of pulse generator (typically 50Ω)

WAVEFORM 1: PROPAGATION DELAYS FOR INVERTING OUTPUTS (f=1MHz; 50% duty cycle)

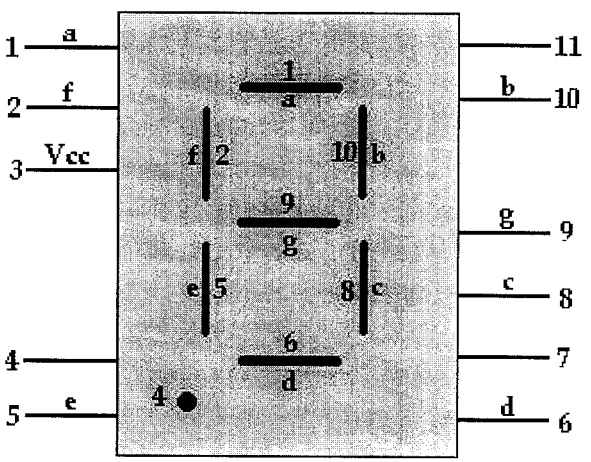


WAVEFORM 2: PROPAGATION DELAYS FOR NON-INVERTING OUTPUTS (f=1MHz; 50% duty cycle)



SEVEN SEGMENT DISPLAY:

PIN DIAGRAM:



FAIRCHILD SEMICONDUCTOR™

October 1988
Revised March 2000

DM74LS47
BCD to 7-Segment Decoder/Driver with Open-Collector Outputs

General Description

The DM74LS47 accepts four lines of BCD (8421) input data, generates their complements internally and decodes the data with seven AND/OR gates having open-collector outputs to drive indicator segments directly. Each segment output is guaranteed to sink 24 mA in the ON (LOW) state and withstand 15V in the OFF (HIGH) state with a maximum leakage current of 250 µA. Auxiliary inputs provided blanking, lamp test and cascadable zero-suppression functions.

Features

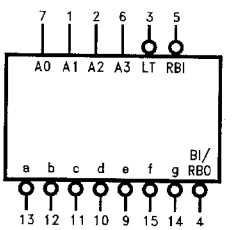
- Open-collector outputs
- Drive indicator segments directly
- Cascadable zero-suppression capability
- Lamp test input

Ordering Code:

Order Number	Package Number	Package Description
DM74LS47M	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150 Narrow
DM74LS47N	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

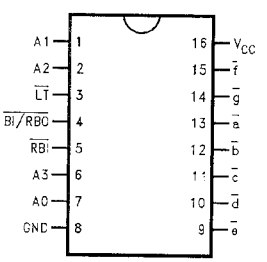
Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Logic Symbol



V_{CC} = Pin 16
GND = Pin 8

Connection Diagram



Pin Descriptions

Pin Names	Description
A0–A3	BCD Inputs
$\overline{RBI}$	Ripple Blanking Input (Active LOW)
$\overline{LT}$	Lamp Test Input (Active LOW)
$\overline{BI}/\overline{RBO}$	Blanking Input (Active LOW) or Ripple Blanking Output (Active LOW)
$\overline{a}$ –$\overline{g}$	Segment Outputs (Active LOW) (Note 1)

Note 1: OC—Open Collector

Truth Table

Decimal or Function	Inputs							Outputs							Note
	$\overline{LT}$	$\overline{RBI}$	A3	A2	A1	A0	$\overline{BI/RBO}$	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$	
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H	(Note 2)
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H	(Note 2)
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L	
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L	
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L	
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L	
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L	
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H	
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L	
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L	
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L	
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L	
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L	
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L	
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L	
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H	
$\overline{BI}$	X	X	X	X	X	X	L	H	H	H	H	H	H	H	(Note 3)
$\overline{RBI}$	H	L	L	L	L	L	L	H	H	H	H	H	H	H	(Note 4)
$\overline{LT}$	L	X	X	X	X	X	H	L	L	L	L	L	L	L	(Note 5)

Note 2: $\overline{BI/RBO}$ is wire-AND logic serving as blanking input ($\overline{BI}$) and/or ripple-blanking output ($\overline{RBO}$). The blanking out ($\overline{BI}$) must be open or held at a HIGH level when output functions 0 through 15 are desired, and ripple-blanking input ($\overline{RBI}$) must be open or at a HIGH level if blanking or a decimal 0 is not desired. X = input may be HIGH or LOW.

Note 3: When a LOW level is applied to the blanking input (forced condition) all segment outputs go to a HIGH level regardless of the state of any other input condition.

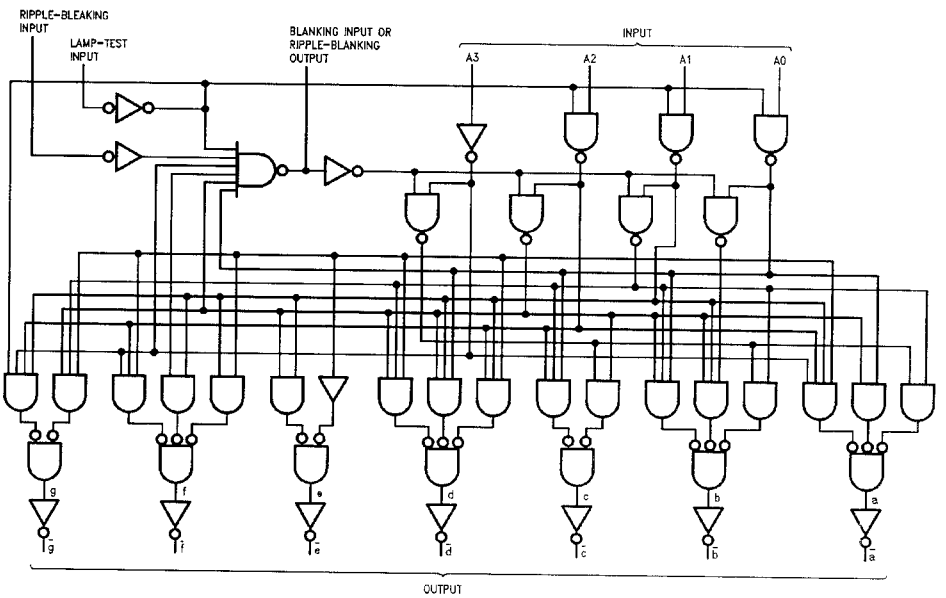
Note 4: When ripple-blanking input ($\overline{RBI}$) and inputs A0, A1, A2 and A3 are LOW level, with the lamp test input at HIGH level, all segment outputs go to a HIGH level and the ripple-blanking output ($\overline{RBO}$) goes to a LOW level (response condition).

Note 5: When the blanking input/ripple-blanking output ($\overline{BI/RBO}$) is OPEN or held at a HIGH level, and a LOW level is applied to lamp test input, all segment outputs go to a LOW level.

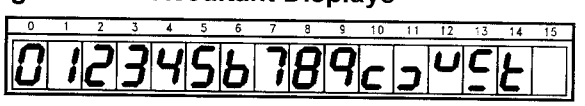
Functional Description

The DM74LS47 decodes the input data in the pattern indicated in the Truth Table and the segment identification illustration. If the input data is decimal zero, a LOW signal applied to the $\overline{RBI}$ blanks the display and causes a multi-digit display. For example, by grounding the $\overline{RBI}$ of the highest order decoder and connecting its $\overline{BI/RBO}$ to $\overline{RBI}$ of the next lowest order decoder, etc., leading zeros will be suppressed. Similarly, by grounding $\overline{RBI}$ of the lowest order decoder and connecting its $\overline{BI/RBO}$ to $\overline{RBI}$ of the next highest order decoder, etc., trailing zeros will be suppressed. Leading and trailing zeros can be suppressed simultaneously by using external gates, i.e.: by driving $\overline{RBI}$ of a intermediate decoder from an OR gate whose inputs are $\overline{BI/RBO}$ of the next highest and lowest order decoders. $\overline{BI/RBO}$ also serves as an unconditional blanking input. The internal NAND gate that generates the $\overline{RBO}$ signal has a resistive pull-up, as opposed to a totem pole, and thus $\overline{BI/RBO}$ can be forced LOW by external means, using wired-collector logic. A LOW signal thus applied to $\overline{BI/RBO}$ turns off all segment outputs. This blanking feature can be used to control display intensity by varying the duty cycle of the blanking signal. A LOW signal applied to $\overline{LT}$ turns on all segment outputs, provided that $\overline{BI/RBO}$ is not forced LOW.

Logic Diagram



Numerical Designations—Resultant Displays



Absolute Maximum Ratings(Note 6)

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note 6: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage	2			V
V_{IL}	LOW Level Input Voltage			0.8	V
I_{OH}	HIGH Level Output Current $\overline{a} - \overline{g}$ @ 15V = V_{OH} (Note 7)			−250	µA
I_{OH}	HIGH Level Output Current $\overline{BI}/\overline{RBO}$			−50	µA
I_{OL}	LOW Level Output Current			24	mA
T_A	Free Air Operating Temperature	0		70	°C

Note 7: OFF-State at $\overline{a}$–$\overline{g}$.

Electrical Characteristics

Over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 8)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, $\overline{BI}/\overline{RBO}$	2.7	3.4		V
I_{OFF}	Output HIGH Current Segment Outputs	V_{CC} = 5.5V, V_O = 15V $\overline{a} - \overline{g}$			250	µA
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IH} = Min, $\overline{a} - \overline{g}$		0.35	0.5	V
		I_{OL} = 3.2 mA, $\overline{BI}/\overline{RBO}$			0.5	
		I_{OL} = 12 mA, $\overline{a} - \overline{g}$		0.25	0.4	
		I_{OL} = 1.6 mA, $\overline{BI}/\overline{RBO}$			0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			100	µA
		V_{CC} = Max, V_I = 10V				
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.4	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 9), I_{OS} at $\overline{BI}/\overline{RBO}$	−0.3		−2.0	mA
I_{CC}	Supply Current	V_{CC} = Max			13	mA

Note 8: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 9: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Switching Characteristics

at V_{CC} = +5.0V, T_A = +25°C

Symbol	Parameter	Conditions	R_L = 665Ω, C_L = 15 pF		Units
			Min	Max	
t_{PLH}	Propagation Delay An to $\overline{a} - \overline{g}$			100	ns
t_{PHL}				100	
t_{PLH}	Propagation Delay $\overline{RBI}$ to $\overline{a} - \overline{g}$ (Note 10)			100	ns
t_{PHL}				100	

Note 10: $\overline{LT}$ = HIGH, A0–A3 = LOW