Wireless Surviellance of Energy

Project Report

Submitted By

Sriram.K.M.

Shanthakumar.R.

Sudhakar.K.

Mohankumar.N.S.

Guided By

Prof . Muthuraman Ramasamy,M.E.,MISTE.,(Ph.D)
Head of the ECE Department

2002 - 2003

In partial fulfillment 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 – 641 006.

Kumaraguru College of Technology

Department of Electronics & Communication Engineering

Coimbatore – 641 006

CERTIFICATE

This is to certify that the contents of the project report entitled

'WIRELESS SURVEILLANCE OF ENERGY'

has been submitted by Mr / Ms . . . .

In partial fulfillment of the requirement for the award of the

Degree of **BACHELOR OF ENGINEERING**

IN ELECTRONICS & COMMUNICATION ENGINEERING

Branch of the Bharathiar University, Coimbatore.

During the academic year 2002 – 2003

. . . . 17/3

Internal Guide

. . . .

Head of the Department

Certified that the candidate with University Register Number . . . .

was examined by us in Project Viva – Voce Examination held on . . . .

. . . . 18/3

Internal Examiner

. . . . 18/3 . . . .

External Examiner

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved Principal, Mr.K.Padmanaban B.Sc. (Engg), M.Tech., Ph.D., for all the facilities provided in carrying out this project work.

We express our gratitude and indebtedness to our respected guide Prof. Muthuraman Ramasamy M.E., FIE. FIETE., MIEEE (USA)., MISTE.,MBMESI.,C.Eng.(I)., Head of the department of Electronics and Communication engineering for his full fledged technical guidance, constant encouragement and suggestions in carrying out this project.

We also take this opportunity to thank our Assistant Professors Mr.K.Ramprakash M.E., and Mr.S.Govindaraju M.E., for sharing their in depth knowledge on the subject and providing us with tips as and when required.

We would also like to thank all the teaching and non-teaching staff for guiding us throughout, without which our project would not have been a great success.

SYNOPSIS

This project is microcontroller based Energy Monitor and transmission of energy through wireless to an EB substation This Project deals with the transmission of the various parameters automatically to the Computer based server through wireless media. The various parameters that would be transmitted are:

- Voltage Reading
- Current Reading
- Power Factor
- Power
- Energy Consumed

The mode of transmission used is wireless, but depending on the economy and various factors it could be done with the help of a telephone line or power line too. Our project uses 89C52 micro controller effectively for reading the above mentioned parameters by efficient programming. Since the core component is the micro controller it is cost effective as well as accurate, which is one of the main criteria for any measuring device. The parameters measured are displayed using LCD display through 8255 PPI. For the conversion of analog signal to digital signal ADC is used.

The measured parameter readings are transmitted using Frequency Shift Keying (FSK) Technique. This FSK technique uses two frequencies of lower range. So in order to transmit without attenuation this frequencies are modulated with high frequency carrier signal. By having server based control the main advantages like 24- Hour Surveillance, automatic fault detection, preventing illicit usage, reduced man power can be achieved.

On the receiver end the received signal is demodulated using a technique which then converted in to 0's and 1's in order to receive at the computer. We have used C as a front-end tool, which displays various parameters. With the help of a computer based server we can calculate the Bills for the used electricity, which could be received like the telephone bills of the present days.

CONTENTS

ACKNOWLEDGEMENT

SYNOPSIS

PAGE NO

1. INTRODUCTION

1.1 NEED FOR THIS PROJECT 2

1.2 OVERVIEW 2

1.3 BLOCK DIAGRAM 4

2. POWER SUPPLY UNIT

2.1 TRANSFORMER 6

2.2 RECTIFIER 6

2.3 FILTER 6

2.4 REGULATOR 7

2.5 CURRENT TRANSFORMER 8

2.6 POTENTIAL TRANSFORMER 9

2.7 DIFFERNCE BETWEEN C.T AND P.T 9

3. PRECISION RECTIFIER AND ZERO CROSSING DETECTOR

3.1 PRECISION RECTIFIER 11

3.2 ZERO CROSSING DETECTOR (ZCD) 13

3.3 ZCD FOR PHASE MEASUREMENT 15

4. MICROCONTROLLER BOARD DETAILS

4.11 MICROCONTROLLER 89C52 18

4.12 PIN DIAGRAM 19

4.13 ARCHIECTURE 20

4.14 MEMORY ORGANISATION 23

4.20 ANALOG TO DIGITAL CONVERTER (ADC0809) 25

4.3 DM74LS138 DECODER 27

4.4 DM7414 HEX INVERTER 27

4.5 DM74LS373 TRANSPARENT LATCH 28

4.6 74LS393 COUNTER 29

4.71 8255-PROGRAMMABLE PERIPHERAL INTERFACE 30

4.72 PIN DIAGRAM 30

4.73 BLOCK DIAGRAM 31

5. AM TRANSMITTER

5.1 FSK GENERATOR (555 TIMER) 34

5.2 AMPLITUDE MODULATION 36

5.3 CD4011 NAND GATE 37

6. AM RECEIVER

6.1 ENVELOPE DETECTOR 38

6.2 PHASE LOCK LOOP (IC 565) 39

7. 89C52 MICROCONTROLLER PROGAMMING 42

8. 'C' CODING 75

9. CONCLUSION 79

10. BIBLIOGRAPHY 81

11. APPENDIX 83

1. INTRODUCTION

1.1 NEED FOR THIS PROJECT

This project is microcontroller based Energy Monitor and transmission of energy through wireless to an EB substation. This project will be very much useful to EB distribution system. This project will not only useful to monitor and transfer energy, and also it can show the electricity theft.

If suppose we implement the project for a complete network in an area, that will be consuming very huge amount and it can calculate the energy transmission and energy consumption. By using the above two values we can calculate the transmission loses we are going to implement an energy meter with wireless transmission for single house.

1.2 OVERVIEW

In a house to calculate the energy we need the following parameters to be measured. They are voltage, current, power factor, power with power and time we can calculate the energy. By using potential transformer we can measure voltage, the potential transformer (P.T.) will give a step down AC output. That AC output will be converted into DC with the help of a rectifier. The rectifier is actually a precision rectifier. The precision rectifier will give you a pure DC output according to the input voltage. The precision rectifier is based on operational amplifiers. Then the converted Analog DC voltage will be given to an ADC.

The current transformer is used to measure the current with the help of the precision rectifier as described earlier. The only difference between voltage measurement and current measurement is we can directly measure the voltage from P.T., to measure the current. The C.T has to connect with a shunt resistor. Then the rectified current output will also be given to the ADC.

The ADC is an eight bit eight channel ADC (ADC0809). This will convert the analog voltage into digital data and those digital data will be fed into the microcontroller.

Then the microcontroller will display the voltage and current output in an LCD display. The accuracy for voltage will be 1V and the current will be 0.1 Amps. If you want to increase the accuracy just we have to increase the number of bits.

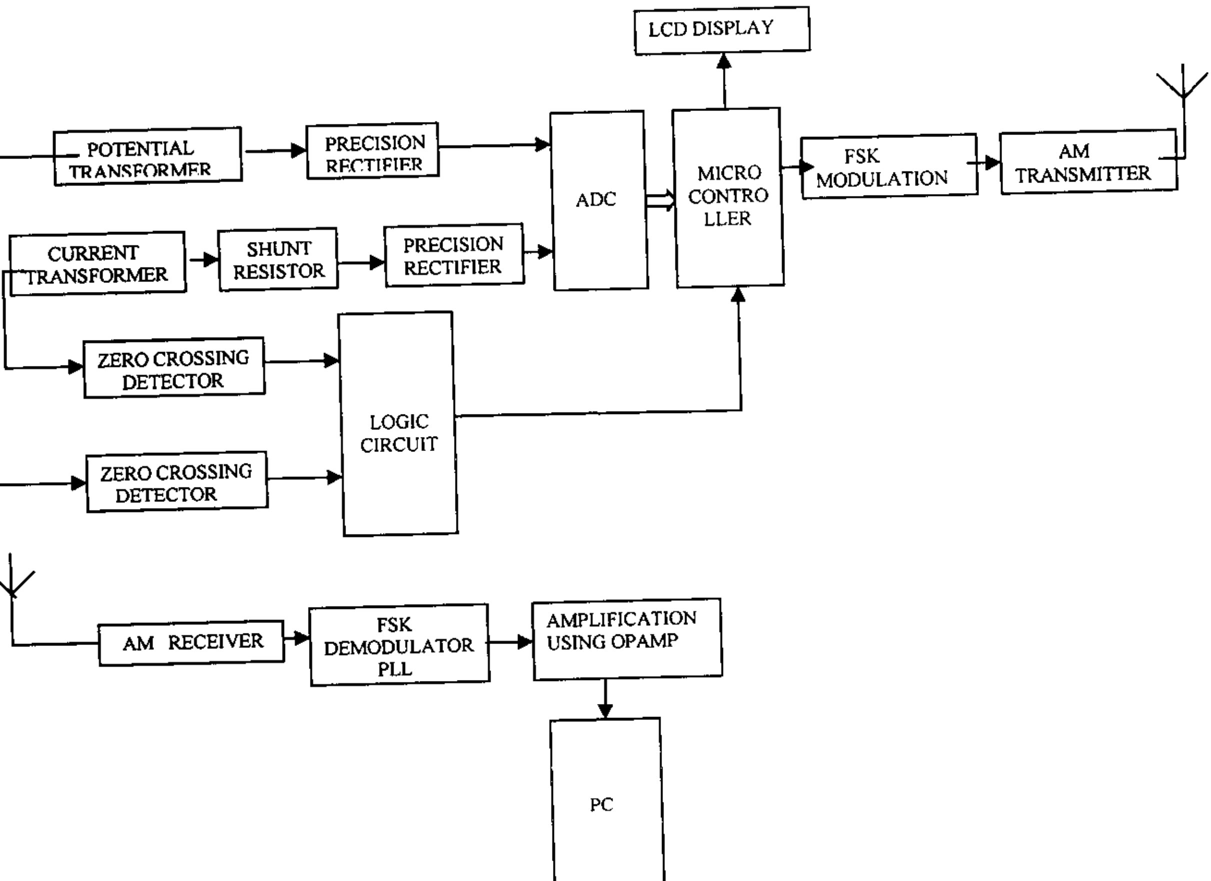
Then the power factor will be measured with the help of both C.T & P.T. The P.T & C.T output will be given to a zero crossing detector and the zero crossing detector output will be given to a logic circuit, that will detect the phase difference between voltage and current waveform. The voltage and current waveform output will be given to a timer program that will calculate the power factor.

After measuring the voltage, current, power factor we can easily calculate the power with the help of a multiplication program. Then by using power and timer we can calculate the energy.

While transmitting it, we have to transmit all the above parameters, that is voltage, current, power, power factor and energy, the transmission will take place through FSK modulation. FSK modulation IC XR2206 will produce two different frequencies for two logic levels and that will be given to an AM transmitter board. The AM output will be transmitted through air.

At the receiver end the AM demodulation take place with the help of diode demodulation technique. Then the AM demodulated output will be given to PLL, the PLL will give '1's and '0's for the frequencies. Then the TTL output will be converted into RS232 output and that will be given to a PC there we have to write a 'C++' program there we can monitor all the measured parameters.

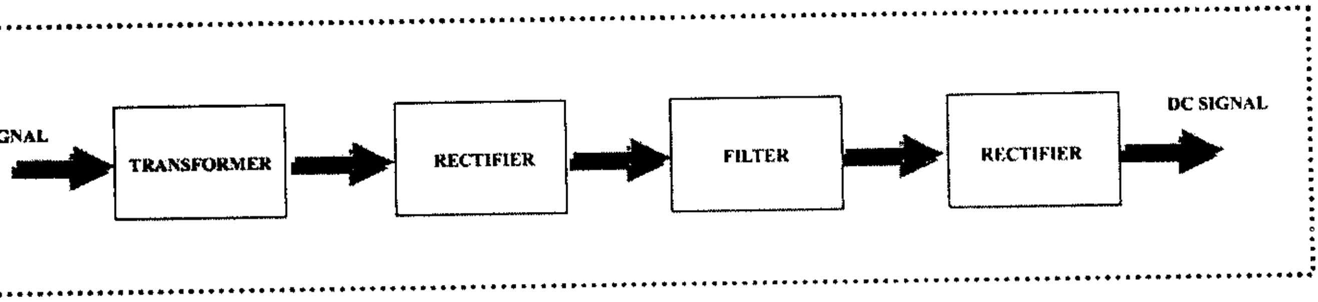
1.3 BLOCK DIAGRAM



2. POWER SUPPLY UNIT

Since all electronic circuits work only with low D.C. voltage we need a power supply unit to provide the appropriate voltage supply. This unit consists of transformer, rectifier, filter and regulator. A.C. voltage typically 230V rms is connected to a transformer which steps that AC voltage down to the level to the desired AC voltage. A diode rectifier then provides a full-wave rectified voltage that is initially filtered by a simple capacitor filter to produce a DC voltage. This resulting DC voltage usually has some ripple or AC voltage variations. A regulator circuit can use this DC input to provide DC voltage that not only has much less ripple voltage but also remains the same DC value even the DC voltage varies some what, or the load connected to the output DC voltages changes.

LOCK DIAGRAM:



2.1 TRANSFORMER

A transformer is a static (or stationary) piece of which electric power in one circuit is transformed into electric power of the same frequency in another circuit. It can raise or lower the voltage in a circuit but with a corresponding decrease or increase in current. It works with the principle of mutual induction. In our project we are using step down transformer for providing a necessary supply for the electronic circuits. In our project we are using a 15-0-15 center tapped transformer.

2.2 RECTIFIER

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

2.3 FILTER

The filter circuit used here is the capacitor filter circuit where a capacitor is connected at the rectifier output, and a DC is obtained across it. The filtered waveform is essentially a DC voltage with negligible ripples, which is ultimately fed to the load.

2.4 REGULATOR

The output voltage from the capacitor is more filtered and finally regulated. The voltage regulator is a device, which maintains the output voltage constant irrespective of the change in supply variations, load variation and temperature changes. Here we use two fixed voltage regulators namely LM 7812, LM 7805 and LM7912. The IC 7812 is a +12V regulator IC 7912 is a -12V regulator and IC 7805 is a +5V regulator.

VOLTAGE & CURRENT TRANSFORMERS

Transformers are used in a.c. systems for the measurement of current, voltage, power and energy. They are also used in connection with measurement of power factor, frequency and for indication of synchronism. The transformer used for measurement of current is called a "Current Transformer" or simply "C.T". Transformers used for voltage measurements are called "Voltage Transformers" or "Potential Transformers" or simply "P.T."

.

2.5 CURRENT TRANSFORMER

The current transformer is used with its primary winding connected in series with line carrying the current to be measured and, therefore, the primary current is dependant upon the load connected to the system and is not determined by the load (burden) connected on the secondary winding of the current transformer. The primary winding consists of very few turns and, therefore, there is no appreciable voltage drop across it. The secondary winding of the current transformer has larger number of turns, the exact number being determined by the turn's ratio. The ammeter, or wattmeter current coil, are connected directly across the secondary winding terminals. Thus a current transformer operates its secondary winding nearly under short circuit conditions. One of the terminals of the secondary winding is earthed so as to protect equipment and personnel in the vicinity In the event of an insulation breakdown in the current transformer.

2.6 POTENTIAL TRANSFORMER

Potential transformers are used to operate voltmeters, the potential coils of wattmeters and relays from high voltage lines. The primary winding of the transformer is connected across the line carrying the voltage to be measured and the voltage circuit is connected across the secondary winding.

The design of potential transformers is quite similar to that of a power transformer but the loading of a potential transformer is always small, sometimes only a few volt-amperes. The secondary winding is designed so that a voltage of 100 to 120 V is delivered to the instrument load. The normal secondary voltage rating is 110 V.

2.7 DIFFERENCE BETWEEN C.T. AND P.T.

There are a few differences in the operation of a current transformer and a potential transformer.

1. The potential transformer may be considered as' parallel' transformer with its secondary winding operating nearly under open circuit conditions whereas the current transformer may be thought as a 'series' transformer under virtual short circuit conditions. Thus the secondary winding of a P.T. can be open-circuited without any damage being caused either to the operator or to the transformer.

2. The primary winding current in a C.T. is independent of the secondary winding circuit conditions while the primary winding current in a P.T. certainly depends upon the secondary circuit burden.

3. In a potential transformer, full line voltage is impressed upon its terminals whereas a C.T. is connected in series with one line and a small voltage exists across its terminals. However, the C.T. carries the full line current.

4. Under normal operation the line voltage is nearly constant and, therefore, the flux density and hence the exciting current of a potential transformer varies only over a restricted range whereas the primary winding current and excitation of a C.T. vary over wide limits in normal operation.

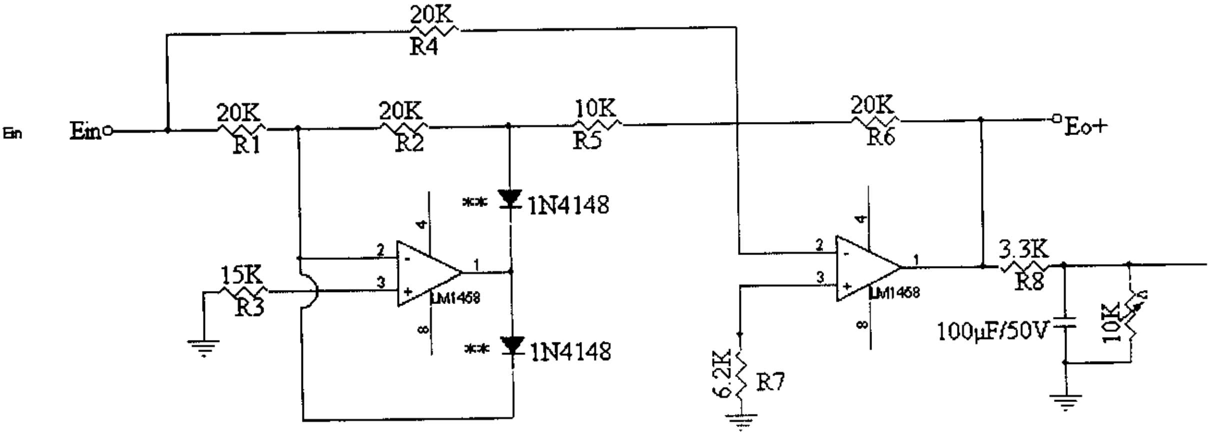
3. PRECISION RECTIFIER AND ZERO CROSSING DETECTOR

3.1 PRECISION RECTIFIER

An absolute-value circuit, or full-wave precision rectifier, can be implemented by summing the output of a half-wave rectifier and its input with the proper phase and amplitude relations. Such a circuit in its basic form is shown in figure. This circuit will be the starting point for a number of other absolute-value circuits, which have evolved from this basic form.

In this circuit given below, A1 is an inverting rectifier similar to the figure. The output from A1 is added to the original input signal in A2 (a summing mixer) with the signal amplitude and phase relations shown. Negative alterations of Ein feeds A2 through 20-kΩ resistor, and E1 feed A2 through a 10-kΩ resistor. The net effect of this scaling is that, for equal amplitudes of Ein and E1, E1will provide twice as much current into the summing point.

This fact is used to advantage here, as the negative alteration of E1 produces twice the input current of that caused by the positive alternation of Ein. This causes a current of precisely half the amplitude, which E1 alone would generate due to the subtraction of Ein. It is the equivalent of having E1 feed through a 20-kΩ input receiver and having Ein non-existent during this half cycle, and it results in a positive going output at A2. During negative alterations of Ein, E1 is absent and Ein produces the alternate positive output swing that, in summation, produces the desired full-wave rectified response. As before, operation with the opposite output polarity is possible by reversing D1 and D2.

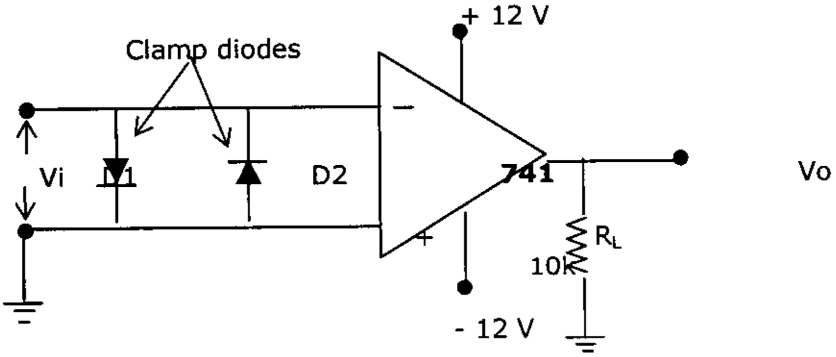


Precision Rectifier Circuit

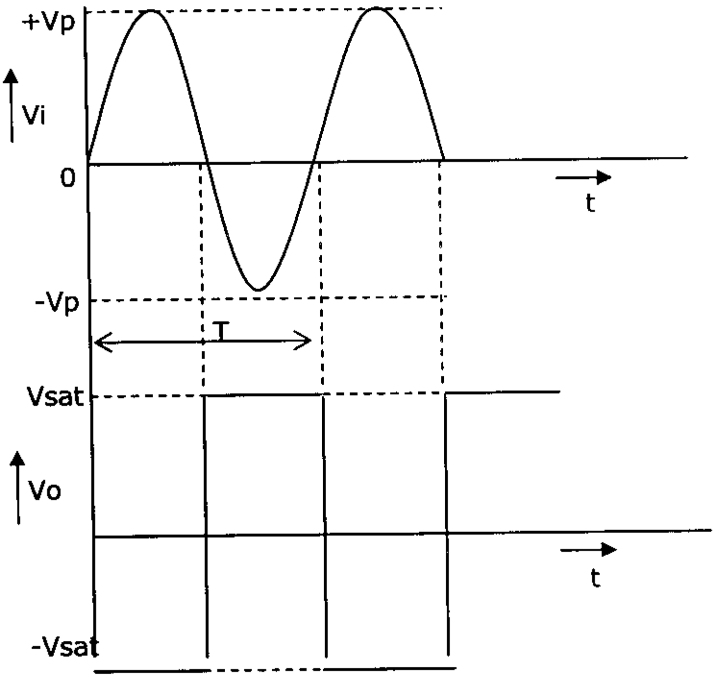
The general -purpose dual-741 type is used in the above circuit. The relationships between resistors for proper circuit operation are noted in the illustration, and may be satisfied best by a single network. Note that resistor R6 can be used as an overall gain trim, or for scaling to net gains of (n) other than unity.

3.2 ZERO CROSSING DETECTOR

An important application of comparator is the zero crossing detector or sine wave to square wave converter. The basic comparator may be used as zero crossing detector provided that V_R is set to zero. With V_i in the form of a sine wave, fig shows the output waveform. It is clear from fig that the output voltage V_o is driven into negative saturation when the input signal Vi passes through zero and goes in positive direction. Similarly when Vi passes through zero and goes in negative direction, the output voltage Vo is driven into positive saturation. This idealized waveform has vertical sides which in reality, should extend over a range of few mV of input voltage Vi.



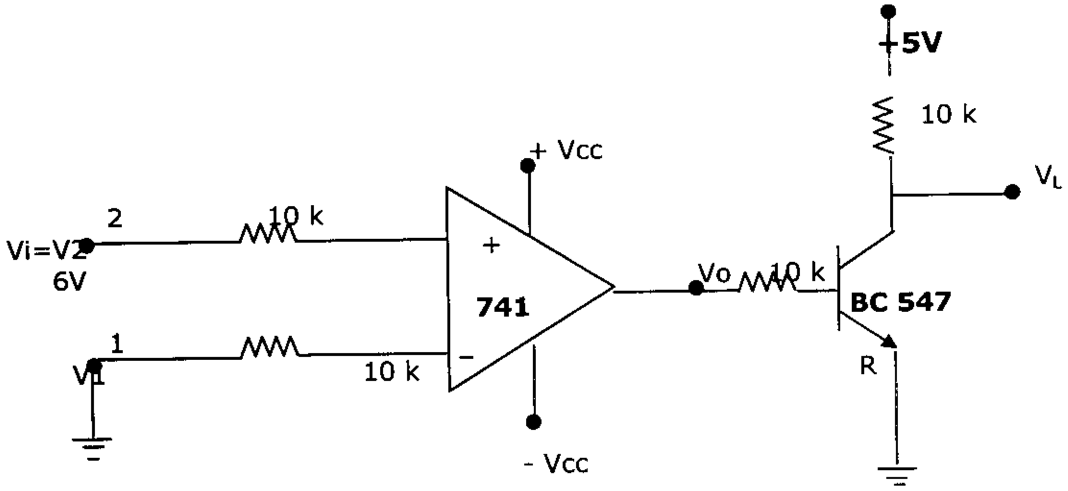
Zero Crossing Detector



Input and output waveforms

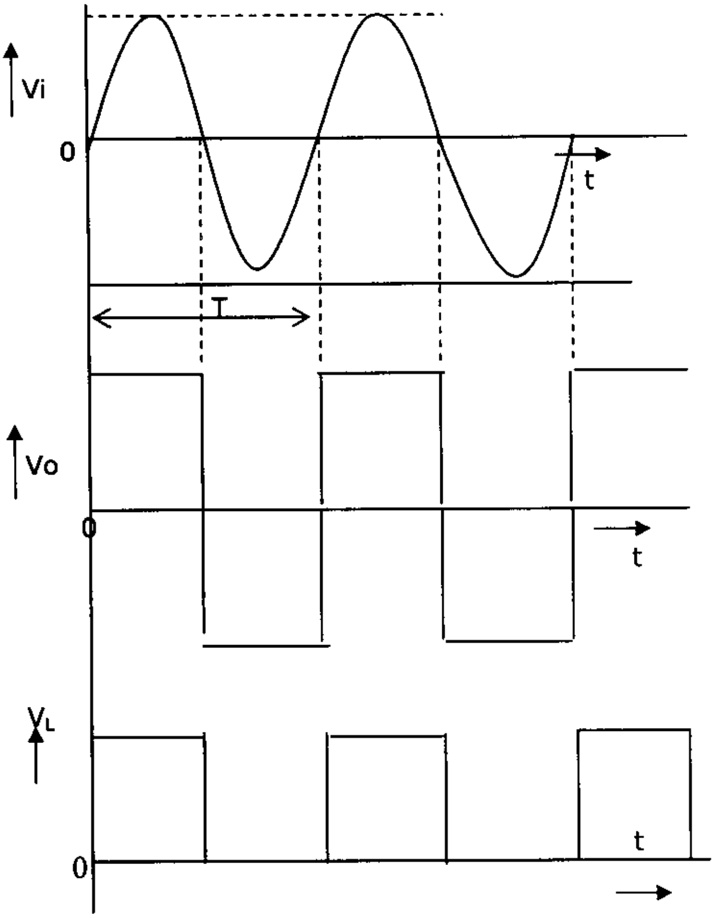
In some cases, the input voltage Vi may be a slowly changing waveform i.e. a signal low frequency. Then it takes Vi more time to cross zero level. Hence Vo may not be able to switch quickly from one saturation level to other saturation level. On the other hand , noise present at the op-amp input may cause Vo to fluctuate between the two saturation levels +Vsat and –Vsat, thus detecting zero crossing for noise voltages also in addition to Vi. Both the above problems may be overcome by using positive or regenerative feedback which causes the output Vo to change faster thereby eliminating the possibility of any false zero crossing due to noise voltages at the Op-amp input.

3.3 ZERO CROSSING DETECTOR FOR PHASE MEASUREMENT



Zero Crossing Detector for phase angle measurement

Circuit of zero crossing detector of the figure is used for measurement of phase angle between two voltages or between voltage and current. In our project this is used to measure the phase angle between voltage and current to calculate the power factor of the circuit. Both the voltage and current are converted into square waves using the above circuit and this is given to the transistor to get only the positive half cycles. Then time interval between the voltage and current wave is measured using an NAND gate (DM 7400) which produces an output of 1 & 0. These outputs are given to 89c52 Microcontroller which with help of timer calculates the time interval that is proportional to the phase difference between the voltage and current.



Pulse Waveforms for phase angle measurement

4.1 MICROCONTROLLER 89C52

4.11 ABOUT THE MICROCONTROLLER

The microcontroller 89C52 is widely used in the electronic and almost all industries. Since the microcontroller controls all the operations it is essential to discuss about its architecture. The special function registers (SFRs) are described below.

ACCUMULATOR

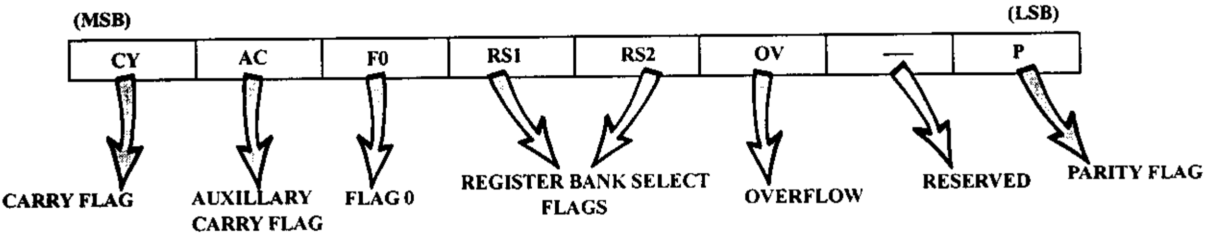
ACC is the accumulator register. The mnemonics for accumulator specific instructions, however, refer to the accumulator simply as **A.** The accumulator is one of the important register.

B REGISTER

The B register is used during multiply and division operations. For other instructions, it can be treated as another scratch pad register.

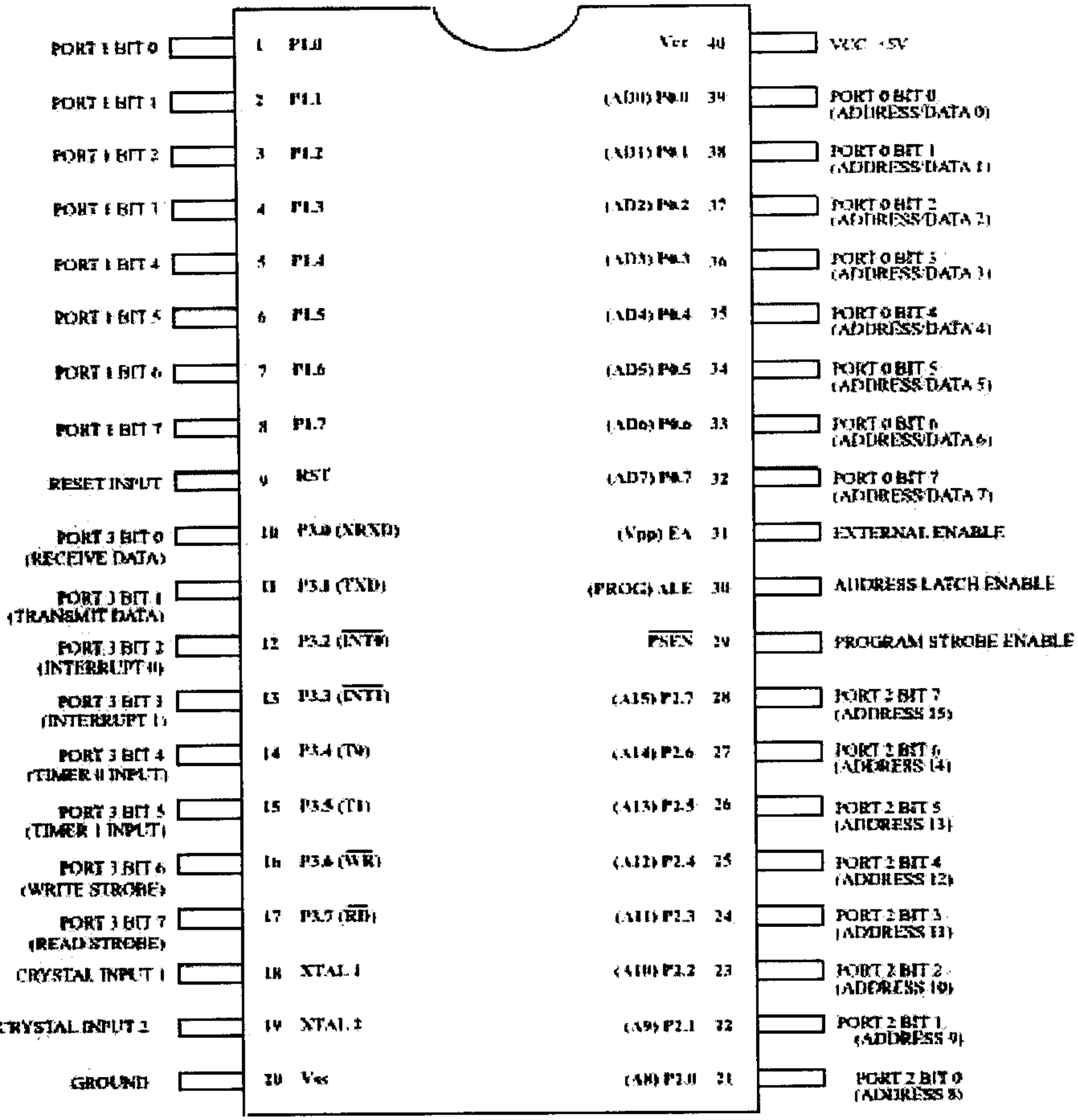
PROGRAM STATUS WORD (PSW)

The program status word register contains the information regarding program status as described in the figure.



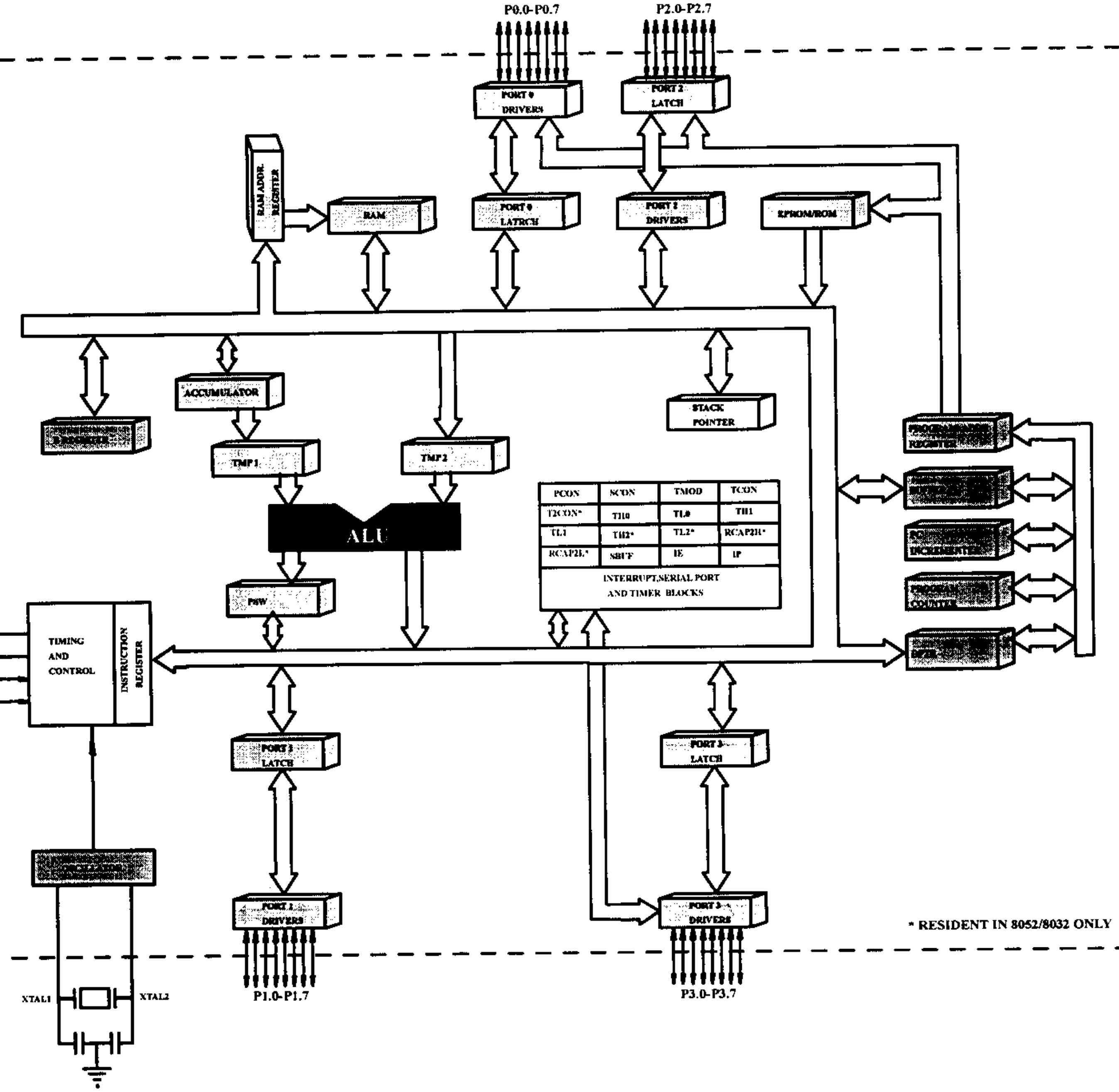
Program Status word

4.12 89C52 PIN DIAGRAM



PORT 1 BIT 0
1 P1.0
PORT 1 BIT 1
2 P1.1
PORT 1 BIT 2
3 P1.2
PORT 1 BIT 3
4 P1.3
PORT 1 BIT 4
5 P1.4
PORT 1 BIT 5
6 P1.5
PORT 1 BIT 6
7 P1.6
PORT 1 BIT 7
8 P1.7
RESET INPUT
9 RST
PORT 3 BIT 0 (RECEIVE DATA)
10 P3.0 (RXD)
PORT 3 BIT 1 (TRANSMIT DATA)
11 P3.1 (TXD)
PORT 3 BIT 2 (INTERRUPT 0)
12 P3.2 ($\overline{INT0}$)
PORT 3 BIT 3 (INTERRUPT 1)
13 P3.3 ($\overline{INT1}$)
PORT 3 BIT 4 (TIMER 0 INPUT)
14 P3.4 (T0)
PORT 3 BIT 5 (TIMER 1 INPUT)
15 P3.5 (T1)
PORT 3 BIT 6 (WRITE STROBE)
16 P3.6 ($\overline{WR}$)
PORT 3 BIT 7 (READ STROBE)
17 P3.7 ($\overline{RD}$)
CRYSTAL INPUT 1
18 XTAL 1
CRYSTAL INPUT 2
19 XTAL 2
GROUND
20 Vss
Vcc 40
VCC +5V
(AD0) P0.0 39
PORT 0 BIT 0 (ADDRESS/DATA 0)
(AD1) P0.1 38
PORT 0 BIT 1 (ADDRESS/DATA 1)
(AD2) P0.2 37
PORT 0 BIT 2 (ADDRESS/DATA 2)
(AD3) P0.3 36
PORT 0 BIT 3 (ADDRESS/DATA 3)
(AD4) P0.4 35
PORT 0 BIT 4 (ADDRESS/DATA 4)
(AD5) P0.5 34
PORT 0 BIT 5 (ADDRESS/DATA 5)
(AD6) P0.6 33
PORT 0 BIT 6 (ADDRESS/DATA 6)
(AD7) P0.7 32
PORT 0 BIT 7 (ADDRESS/DATA 7)
(Vpp) EA 31
EXTERNAL ENABLE
(PROG) ALE 30
ADDRESS LATCH ENABLE
$\overline{PSEN}$ 29
PROGRAM STROBE ENABLE
(A15) P2.7 28
PORT 2 BIT 7 (ADDRESS 15)
(A14) P2.6 27
PORT 2 BIT 6 (ADDRESS 14)
(A13) P2.5 26
PORT 2 BIT 5 (ADDRESS 13)
(A12) P2.4 25
PORT 2 BIT 4 (ADDRESS 12)
(A11) P2.3 24
PORT 2 BIT 3 (ADDRESS 11)
(A10) P2.2 23
PORT 2 BIT 2 (ADDRESS 10)
(A9) P2.1 22
PORT 2 BIT 1 (ADDRESS 9)
(A8) P2.0 21
PORT 2 BIT 0 (ADDRESS 8)

4.13 89C52 ARCHITECTURE



PCON	SCON	TMOD	TCON
T2CON*	TH0	TL0	TH1
TL1	TH2*	TL2*	RCAP2H*
RCAP2L*	SBUF	IE	IP

INTERRUPT,SERIAL PORT AND TIMER BLOCKS

* RESIDENT IN 8052/8032 ONLY

STACK POINTER

The stack pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on chip RAM, the stack pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

DATA POINTER

The data pointer consists of 2 bytes which can be accessed as two separate bytes as DPH and DPL or as a single 16 bit register as a whole. It is being used to hold the 16 bit address to refer any data in the data memory.

PORTS

P0, P1, P2 and P3 are the special function registers (SFR) which are having the RAM address of 80H, 90H, A0H and B0H respectively. These registers can be bit addressable also.

SERIAL DATA BUFFER

The serial data buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the data buffer where it is held for serial transmission. When data is moved from SBUF, it comes from the receiver.

TIMER/COUNTER REGISTERS

The microcontroller 89C52 has two timer registers namely T0 and T1 which are 16 bit up counters. The counters are being used to monitor both external and internal events and to provide accurate delays. These registers can be accessed as single byte registers also. (TH0, TL0, TH1 and TL1).

CONTROL REGISTERS

Special function registers IP, IE, TMOD, TCON, SCON and PCON content control and status bit for the interrupt system, the timer/counter, and the serial ports.

SERIAL INTERFACE

The serial port is full duplex, meaning it can transmit and receive simultaneously. It is also receive buffered, meaning that it can commence the reception of second byte before a previously received byte has been read from the receive register. The serial port receive and transmit register are both accessed at special function register SBUF. Writing from SBUF loads the transmit register, and reading SBUF accesses a physically separate register. The serial port can operate in following four modes as shown in figure which can be controlled by SCON register.

SCON bit 7 *SCON bit 6*

SM0	SM1	MODE	DESCRIPTION
0	0	0	Shift register : baud = f/12
1	1	1	8 bit UART : baud = variable
0	0	2	9 bit UART : baud = f/32 or f/64
1	1	3	9 bit UART : baud = variable

Four serial Modes Of 89c52

INTERRUPTS

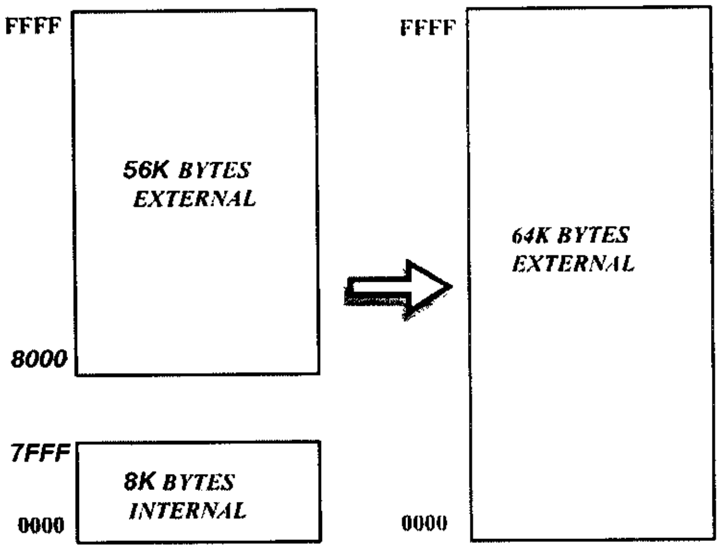
The 89c52 microcontroller consists of five interrupt sources in which three are internal and two are external. The two external interrupts are INT0 and INT1 which can be level activated or transition activated, depending on bits of IT0 and IT1 in register TCON. The other three interrupts, which are internal, are the Timer1, Timer2 interrupts and the serial port interrupt.

The Timer0 and Timer1 interrupts are generated by TF0 and TF1, which are set by a rollover in their respective timer/counter registers. The serial port interrupt is generated by the logical OR of RI and TI.

4.14 MEMORY ORGANISATION

PROGRAM MEMORY

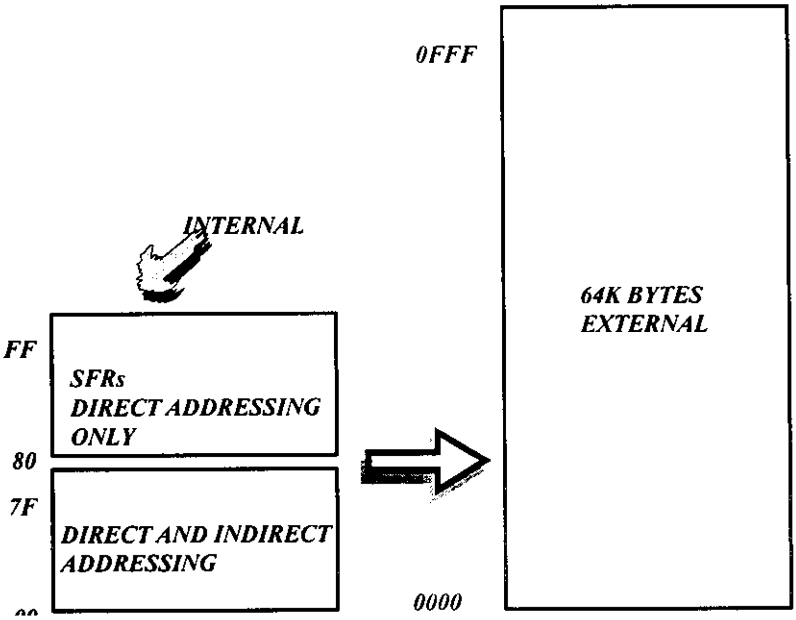
The 89c52 has separate address spaces for program and data memory. The program memory can be up to 64K bytes long. The lower 8K can reside on-chip. The figure shows the map of the 89c52 program memory.



DATA MEMORY

The 89c52 has 128 bytes of on-chip RAM, plus a number of Special Function Registers (SFRs). The lower 128 bytes of RAM which can be accessed by both direct and indirect addressing can be divided into three segments as,

1. Register Banks 0-3, 2. Bit Addressable Area, 3. Scratch Pad Area.



4.2 ANALOG TO DIGITAL CONVERTER (ADC0809)

The ADC0809 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 8-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique.. The 8-channel multiplexer can directly access any of 8-single-ended analog signals. The device eliminates the need for external zero and full-scale adjustments. Easy interfacing to microprocessors is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATE outputs. The ADC0809 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications from process and machine control to consumer and automotive applications.

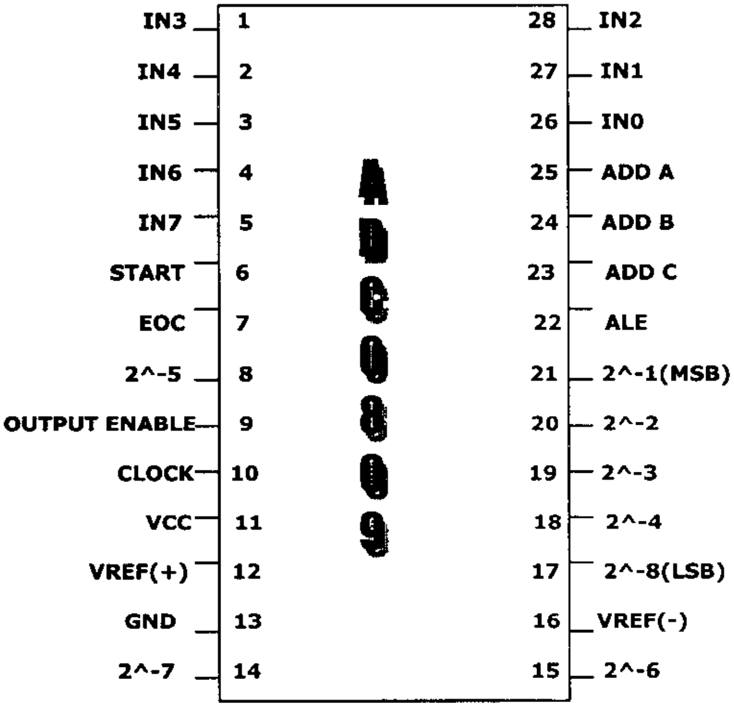
The device contains an 8-channel single-ended analog signal multiplexer. A particular input channel is selected by using the address decoder. Table below shows the input states for the address lines to select any channel. The address is latched into the decoder on the low-to-high transition of the address latch enable signal.

SELECTED ANALOG CHANNEL	ADDRESS LINES		
	C	B	A
IN0	L	L	L
IN1	L	L	H
IN2	L	H	L
IN3	L	H	H
IN4	H	L	L
IN5	H	L	H
IN6	H	H	L
IN7	H	H	H

In our project the analog inputs namely Voltage and Current are to be converted in to digital outputs in order to access by the 89c52 microcontroller. At regular intervals one of the inputs is selected and converted to digital 8-Bit output data. The converted digital output is given to 89c52 through port 0.

The clock at which this ADC0809 is to be operated is 750 KHz. This clock signal comes from the external crystal oscillator operating at a frequency of 12 MHz. This 12 MHz clock is used for 89c52 microcontroller. By using counter this clock frequency of 12 MHz is divided to obtain 750 KHz and given as clock input for ADC0809. During each clock period the conversion take place.

Pin details of ADC0809 is shown below



Pin diagram of ADC0809

4.3 DM74LS138 DECODER

These Schottky-clamped circuits are designed to be used in high performance memory-decoding or data-routing applications, requiring very short propagation delay times. In high-performance memory systems these decoders can be used to minimize the effects of system decoding. When used with high-speed memories, the delay times of these decoders are usually less than the typical access time of the memory. This means that the effective system delay introduced by the decoder is negligible. The LS138 decodes one-of-eight lines; based upon the conditions at the three binary select inputs and the three enable inputs.

The decoder is used to select 8255-PPI and ADC0809 depending upon the address lines input of 89c52. By default the output of the decoder is high. As the chip is selected the output turns low which selects either ADC or 8255.

4.4 DM7414- HEX INVERTER WITH SCHMITT TRIGGER INPUTS

This device contains six independent gates each of which performs the logic INVERT function. Each input has hysteresis which increases the noise immunity and transforms a slowly changing input signal to a fast changing, jitter free output.

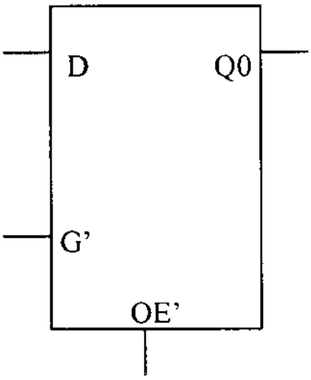
The function is given by Y=A′

Input **A**	**Output** **Y**
L H	H L

H=High Logic Level
L = Low Logic Level

4.5 DM74LS373-TRANSPARENT LATCH

These 8-bit registers feature 3-STATE outputs designed specifically for driving highly capacitive or relatively low-impedance loads. They are particularly attractive for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers. The eight latches of the DM54/74LS373 are transparent D-type latches meaning that while the enable (G) is high the Q outputs will follow the data (D) inputs. When the enable is taken low the output will be latched at the level of the data that was set up.



D-Flip flop

Function Table

Output Control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q0
H	X	X	Z

H=High Level (Steady State), L = Low Level (Steady State), X = Don't Care

Z = High Impedance State, Q 0 = The level of the output before steady-state input conditions were established.

4.6 74LS393 COUNTER

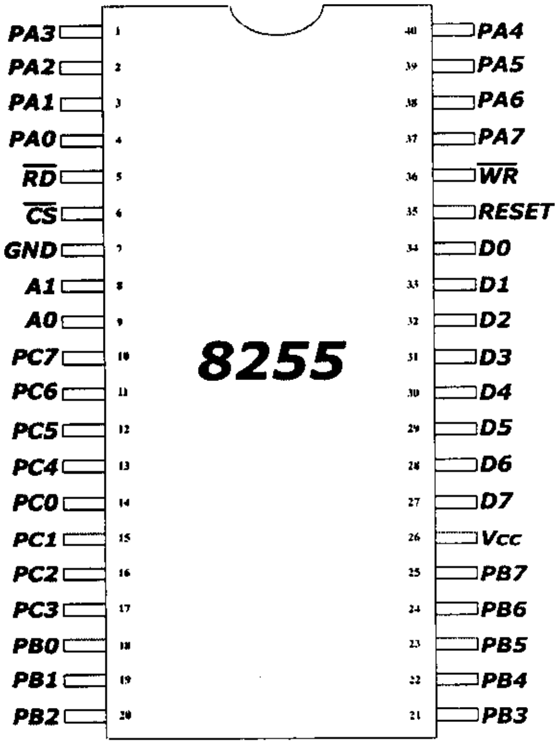
The **74LS39** is actually dual 4-bit binary ripple counter. It consists of two 4-bit binary counters. There are separately two master resets to clear each 4-bit counter separately. Each counter has separate clock so that they can operate independently. The counters are triggered by a HIGH-to-LOW transition of the clock inputs.

We are using a master crystal of generating 12MHz signal for the microcontroller and our ADC operates at the 750 KHz. There is a requirement of generating 750 KHz frequency signal for ADC using the same crystal. This can be achieved by applying the master crystal signal to this counter and by loading the counter with proper count value before ADC begin operations.

4.6 8255-PROGRAMMABLE PERIPHERAL INTERFACE

4.61 DESCRIPTION

The 8255 is general purpose I/O interfacing device. It provides 24 I/O lines organized as three 8-bit I/O ports labeled A, B, and C. It is a very versatile device in the means that this can be programmed to look like three simple I/O ports (mode0), two handshaking I/O ports (mode1), or a bidirectional I/O port with five handshaking signals (mode2). There is also a bit reset/set mode that allows individual bits of port C to be set or reset for control purposes. Pin definitions and a block diagram are provided in the following figures.



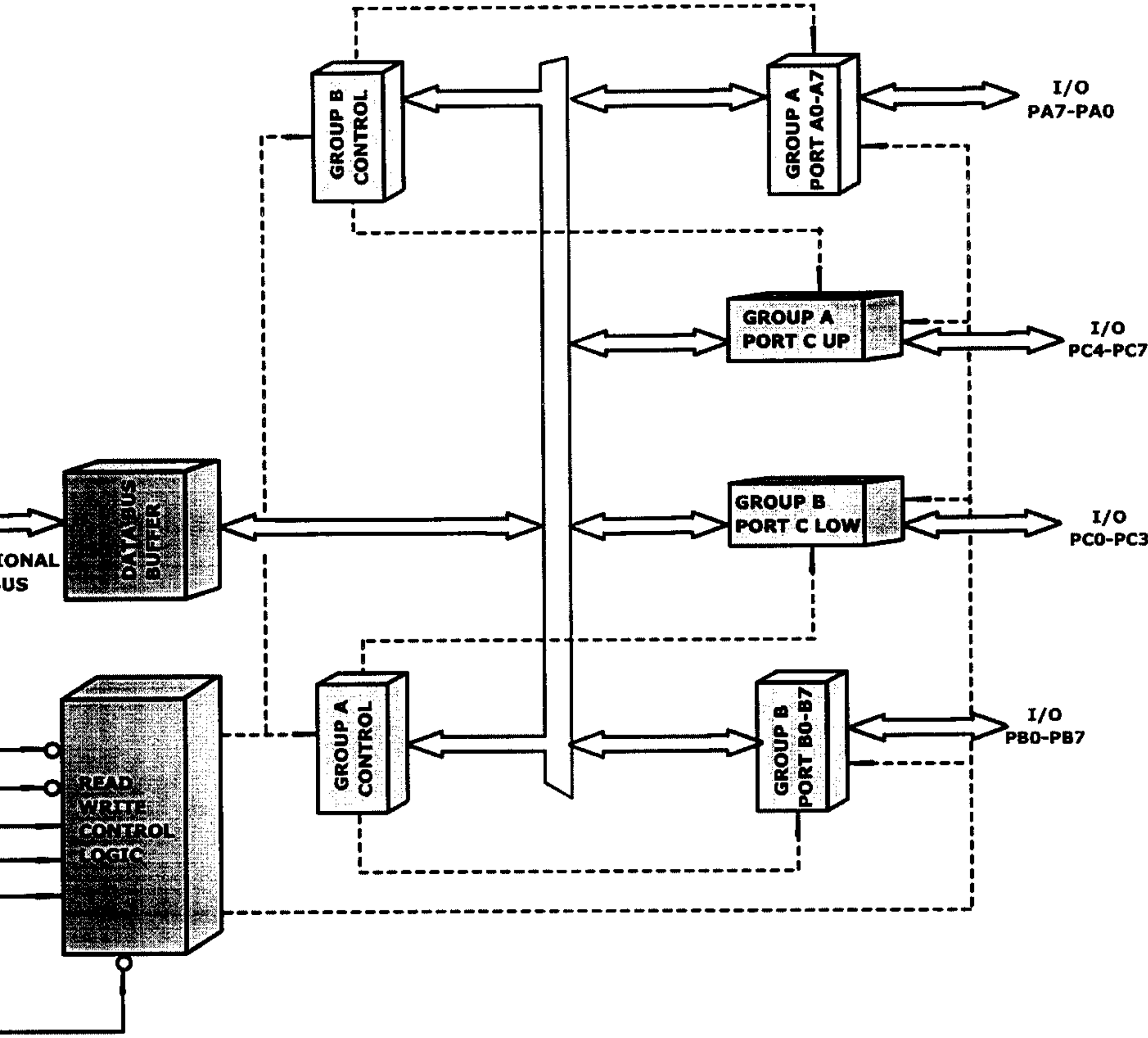
PIN NAMES

D_7-D_0	***DATA BUS (BIDIRECTIONAL)***
RESET	***RESET INPUT***
CS	***CHIP SELECT***
RD	***READ INPUT***
WR	***WRITE INPUT***
A0,A1	***PORT ADDRESS***
PA7-PA0	***PORT A (BIT)***
PB7-PB0	***PORT B (BIT)***
PC7-PC0	***PORT C (BIT)***
Vcc	***+5 VOLTS***
GND	***0 VOLTS***

4.62 PIN DIAGRAM

PIN DETAILS

4.63 BLOCK DIAGRAM



CONTROL WORD

As mentioned above the programmable peripheral interface 8255 can be operated in three different modes as well as bit SET/RESET mode. The mode of operation can be controlled by a word, which is going to be stored in control register of the 8255, called as control word. The format of the control word is shown below.

MODE 0 (BASIC I/O)

When there is no need for conditional or handshaking I/O mode 0 can be selected. In this mode of operation all the ports are simply treated as I/O ports.

MODE 1 (STROBED I/O)

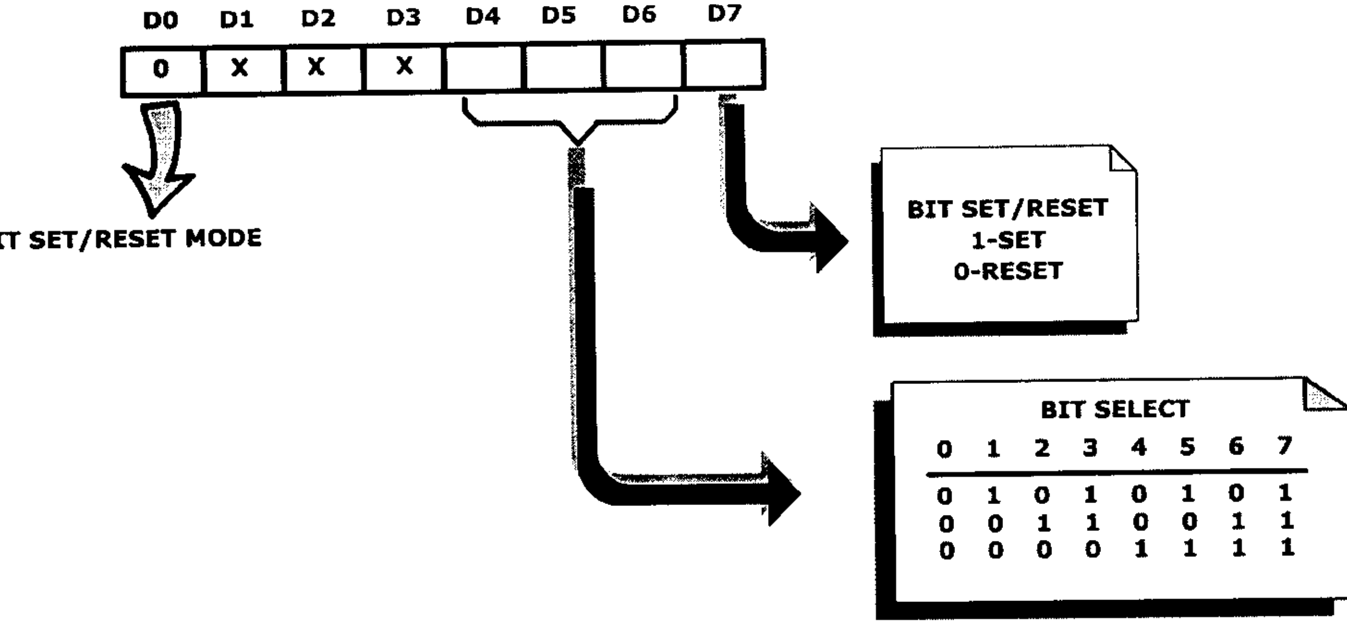
This mode of operation is mainly intended for handshaking and interrupt driven I/O interfaces. Here, ports A and B are programmed as data ports and pot C is programmed to carry status signals.

MODE 2 (STROBED BIDIRECTIONAL I/O)

When operated in mode 2, port A of the 8255 becomes a bidirectional data port supported by five handshaking signals. Port B can operate in mode 0 or mode 1.

BIT SET/RESET MODE

When bit 7 of the 8255 control word is a 0, the bit set/reset mode is active. In this mode any one bit of port C can be set to a logic 1 or reset to a logic 0. Only one bit can be set or reset at a time. This feature of 8255 can be taken advantage of to generate strobe signals.



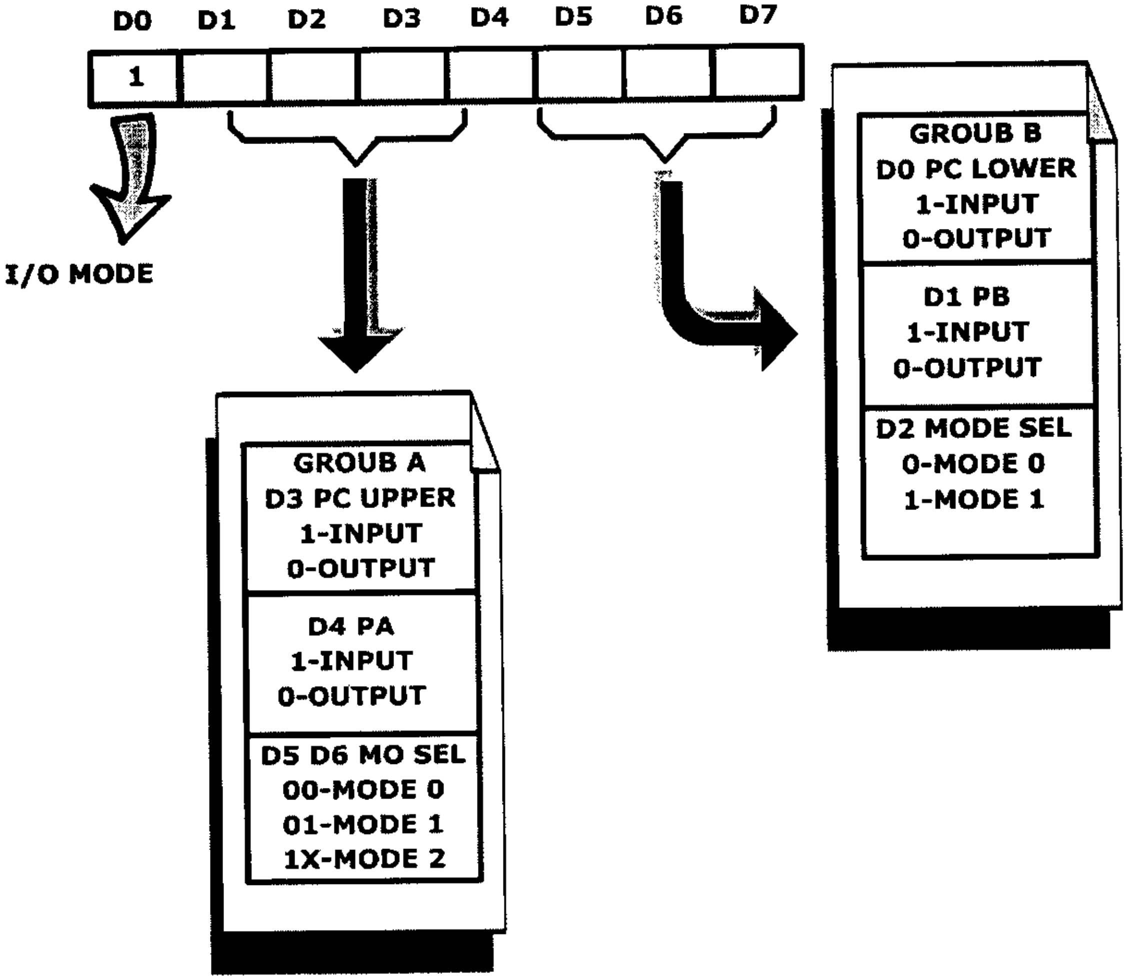
D0	D1	D2	D3	D4	D5	D6	D7
0	X	X	X				

IT SET/RESET MODE

BIT SET/RESET
1-SET
0-RESET

BIT SELECT

0	1	2	3	4	5	6	7
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	0	0	0	1	1	1	1



D0	D1	D2	D3	D4	D5	D6	D7
1							

I/O MODE

GROUB A
D3 PC UPPER
1-INPUT
0-OUTPUT

D4 PA
1-INPUT
0-OUTPUT

D5 D6 MO SEL
00-MODE 0
01-MODE 1
1X-MODE 2

GROUB B
D0 PC LOWER
1-INPUT
0-OUTPUT

D1 PB
1-INPUT
0-OUTPUT

D2 MODE SEL
0-MODE 0
1-MODE 1

5. AM TRANSMITTER

5.1 FSK GENERATOR

In digital data communication, binary code is transmitted by shifting a carrier frequency between two preset frequencies. This type of transmission is called frequency shift keying technique. A 555 timer in astable mode can be used to generate FSK signals. When the input is high, transistor Q is cutoff and 555 timer works in the normal astable mode of operation. The frequency of the output waveform is given by

$$F0 = 1.45/(RA+2RB)\ C$$

When the input is low Q goes on and connects the resistance RC across Ra. The output frequency is given by

$$F0 = 1.45/\ ((RA||RC)+2RB)\ C$$

The resistance is used to get the desired output frequency.

Frequency shift keying (FSK) is the mostly used method for transmitting digital data over telecommunications links. In order to use FSK a modulator-demodulator (modem) is needed to translate digital 1's and 0's into their respective frequencies and back again.

In FSK modulation, the carrier frequency is shifted in steps (or) levels corresponding to the levels of the digital modulating signal. In case of binary signal, two carrier frequencies are used, one was corresponding to binary '0' and another to binary '1'.

TIMER

The 555 timer used in the above circuit is highly stable for generating time delays or oscillations. A single 555 timer can provide a time delay ranging from microseconds to hours where as counter timer can have maximum timing range of days. The 555 timer can be used with supply voltage in the range of +5volts to +18volts and can drive load up to 200milli ampere. It is compatible with both TTL and CMOS logic circuits. Because of wide range of supply voltage the 555timers are versatile and easy to use in various applications. The various applications include oscillator, pulse generator, ramp generator, square wave generator, monoshot multivibrator, burglar alarm etc. In our circuit, astable mode of 555 timers is used.

MODULATION

The purpose of a communication is the source and user being physically separate from each other. To do this, the transmitter modifies the message signal into a suitable form for transmission over the channel. This modification is achieved by a process known as modulation, which involves varying some parameters of a carrier wave in accordance with the message signal. The receiver recreates the original message signal from a degraded version of the transmitted signal after propagation through the channel.

A process known as demodulation, which is a reverse process of modulation used in the transmitter, achieves the recreation. However owing to the unavoidable presence of noise and distortion in the received signal, we find that the receiver cannot recreate the original message signal exactly. The type of modulation scheme used influences the resulting degradation in the overall system performance. Specifically we find that some modulation schemes are less sensitive to the effects of noise and distortion than others.

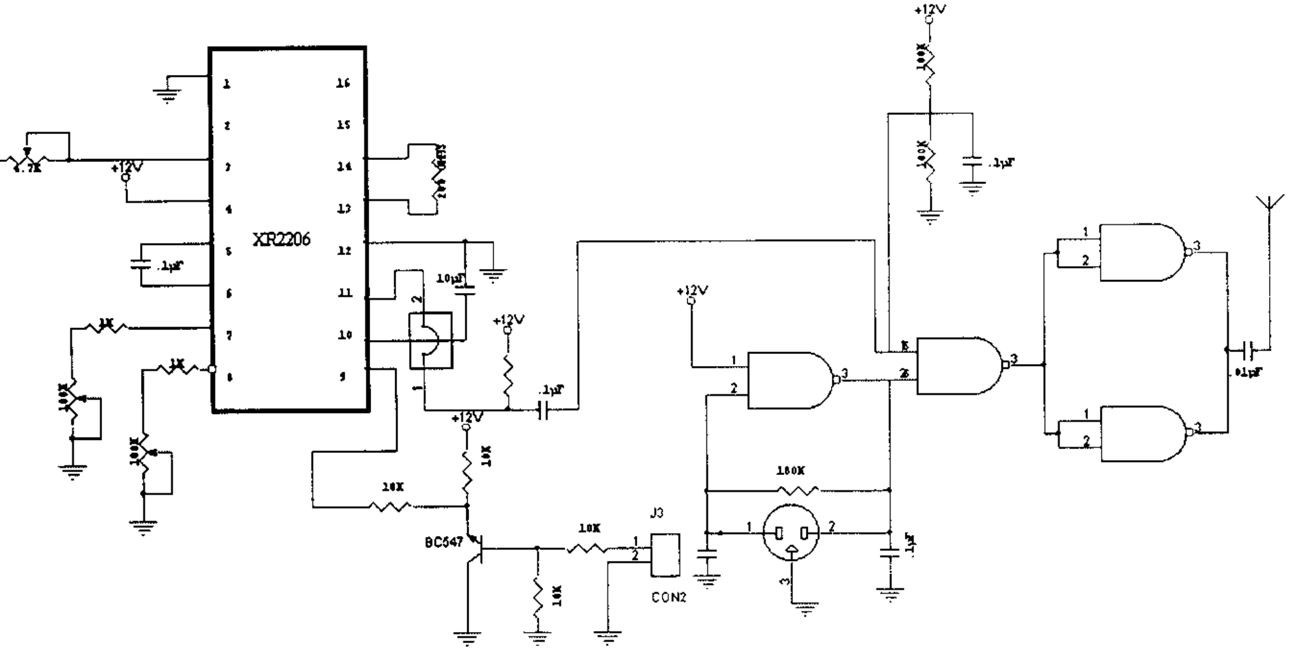
5.2 AMPLITUDE MODULATION

The amplitude of the carrier wave is varied in accordance with the amplitude of the modulating signal. Consider a sinusoidal carrier wave c (t)_ defined by,

$$C\ (t) = Ac\ Cos\ (2*Pi*Fc*t+\phi),$$

where Ac is the carrier amplitude and Fc is the carrier frequency. We have assumed that the phase of the carrier wave is zero for specification of the message. The source of the carrier wave c (t) is physically independent of the source responsible for generating m (t). An amplitude-modulated wave may thus describe, in its most general form, as a function of time.

In the circuit shown, schmitt trigger NAND gate CD4011 is used. The carrier frequency is generated using the ceramic filter of value 10.7 MHz. The resistor R1 and R2 provide the necessary biasing. The modulated output is than transmitted through the antenna after passing through a capacitor. The antenna used is of aerial type. In our project the transmitted frequency is 5.5 MHz. The maximum distance to Which we can transmit is around 30ft to 40ft. the distance to which it can be transmitted can be improved further by providing some amplification at the output of the amplitude modulation circuit.



AM Transmitter

4.3 CD4011-CMOS NAND GATES

The **XR2206** IC generates two frequencies namely 1200 Hz and 1000 Hz representing the digital data 1 and 0 respectively. These generated frequencies get attenuated while transmitting. In order to avoid that, we are padding up those frequencies with the high frequency carrier of 5.5 MHz. This transmission is carried out using the **CD4011- NAND GATES** where the inputs are 5.5MHz carrier and one of the two low frequency signals. The presence of logic 1 is represented as 5.5MHz carrier output and logic 0 as zero Hz signal.

6.AM RECEIVER

The encoded message has to now be decoded. The encoded message is received by receiving antenna and than given to demodulator circuit. This is then given to a DFSK circuit where 0's and 1's are separated out by using carrier at two different frequencies. Then the output is given to a P.C where message is decoded using the same algorithm as in encrypment process and the original message is taken out.

6.1 ENVELOPE DETECTOR

An envelope detector is a simple and yet highly effective device that is well suited for the demodulation of a narrow band AM wave, for which the percentage modulation is less than 100%. Ideally, an envelope detector produces an output signal that follows the envelope of the input signal waveform.

Envelope detector consists of a diode and a resistor-capacitor filter. The operation is as follows. On the positive half-cycle of the input signal, the diode is forward-biased and the capacitor C charges up rapidly to the peak value of the input signal. When the input signal falls below this value, the diode becomes reverse biased and the capacitor C discharges slowly to the load resistor RI. The discharging process continues until the next positive half cycle. When the input signal becomes greater than the voltage across the capacitor, the diode conducts again and the process is repeated. Thus the demodulation is carried out. In our circuit we are using 0A79 diode.

In digital data communication and computer peripheral, binary data is transmitted by means of a carrier frequency, which is shifted between two preset frequencies. This type of data transmission is called frequency shift keying (FSK) technique. The binary data can be retrieved using FSK demodulators at the receiving end.

The 565 PLL is a very useful as a FSK demodulator. As the signal appears at the input, the loop locks to the input frequency and tracks it between the two frequencies with a corresponding and the DC shift at the output. A three-stage filter removed the carrier component and the output signal is made logic compatible by the voltage comparator.

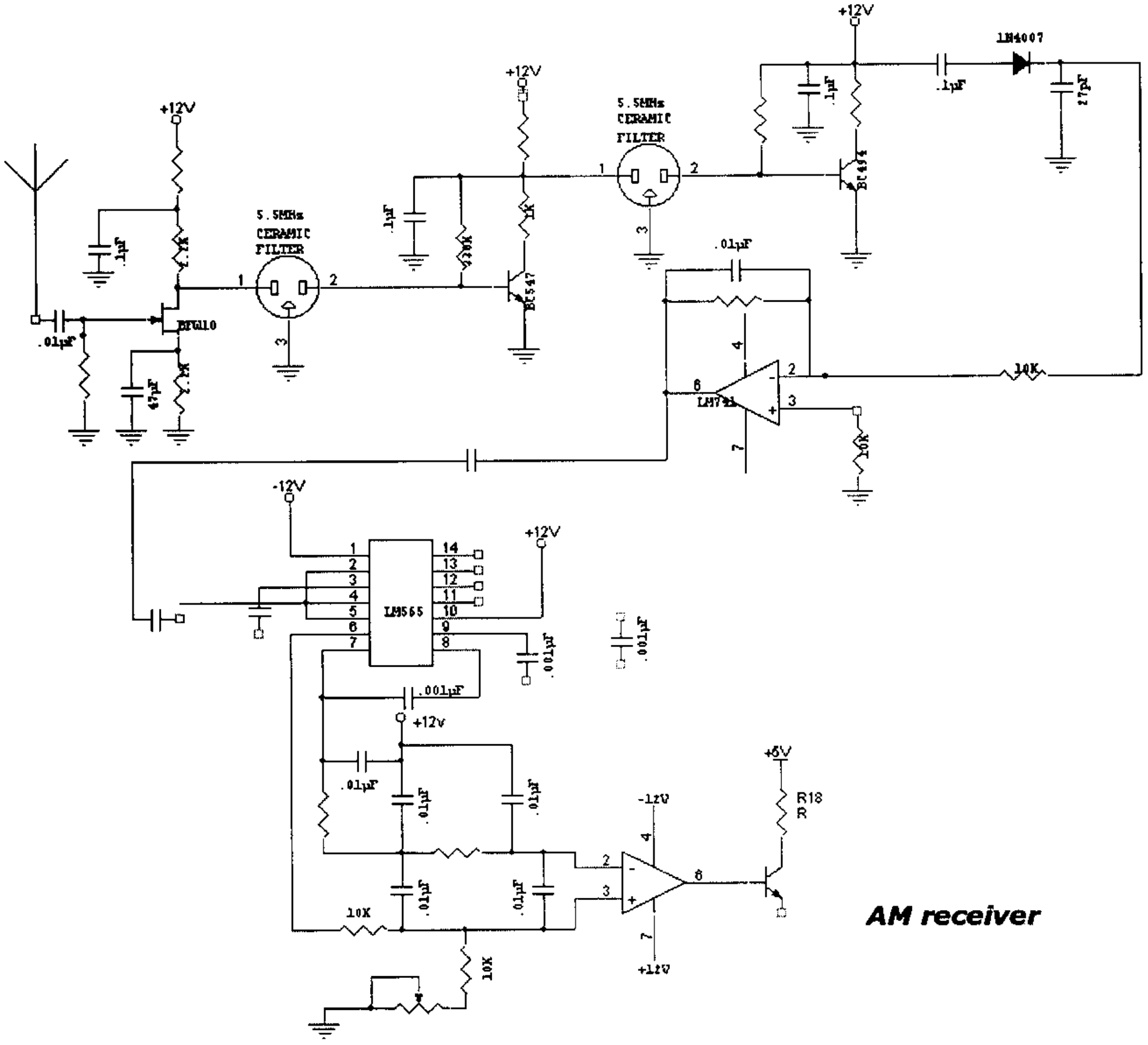
6.2 IC PLL 565

The phase detector of this PLL is comprised of differential amplifier pairs provided with current sink bias source. The output voltage phase detector is limited by diodes to maximum of +0.7V.

This limiting action helps to minimize the effect of high amplitude noise pulses and other transient effects on the operation of the PLL. The phase detector has a balanced output and is supplied to the differential amplifier pair, which serves an amplifier stage in amplifying the phase detector; output a single ended output is taken from this stage from across the load resistor R1 and connected internally to the VCO.

Connection of an external capacitor C between Pin 7 and ground will produce a first order low-pass (lag) network. A capacitor C and a resistor R2 connected in series between pin 7 and ground will result in lag lead network.

The VCO consists of a voltage controlled current source, which supplies equal magnitude of charging and discharging currents to an externally connected (pin 9) timing capacitor C0. A timing resistor is connected between pin8 and positive power supply. The rest of the VCO circuit is Schmitt trigger with a differential amplifier output circuit. This controls the turn-on and turn-off for the switching action of the current source for the charging and discharging.



AM receiver

MICROCONTROLLER PROGRAMMING

```
; #pragma src;
; #include<reg51.h>
; #include<math.h>
; void htdt();
; void dist();
; void cal();
; void disr();
; void htdp();
; void htdp1();
; void ser_init();
; void ser_out();
; void ser_out0();
; void ser_out1();
; void ser_out2();
; //void serout();
; void lcd_init();
; void lcd_dis(unsigned char*,unsigned char);
; void del();
; void delay();
; void read();
; void write();
; void adc0();
; void htd0();
; void dis0();
; void adc1();
; void htd1();
; void dis1();
; void power();
; void htd2();
; void dis2();
; void powf();
; void htd3();
; void dis3();
; sbit pul=P3^2;
; pdata unsigned char ch0 _at_ 0x08;
; pdata unsigned char ch1 _at_ 0x09;
; pdata unsigned char soc _at_ 0x10;
; pdata unsigned char porta _at_ 0x18;
; pdata unsigned char portb _at_ 0x19;
; pdata unsigned char portc _at_ 0x1a;
; pdata unsigned char cwr _at_ 0x1b;
; idata unsigned char a0,b0,c0,d0,a1,b1,c1,d1,a2,c2,e2,f2,g2,h2,i2,j2;
; idata unsigned char i,j,x,dat0,dat1,ser_data;
; idata unsigned int s,b2,d2;
; unsigned int mask,mask2;
; unsigned char mask1,data2;
; //float pow_f,pow,dat;
; unsigned int  pow_f,pow,dat;
; idata unsigned int countr=0,ix,b1p,d1p,f1p,h1p,bp,dp,fp,hp,reg=0,tt;
; unsigned long av;
; //float ttemp1,ttemp2,count1,energy=0,date;
; unsigned int ttemp1,ttemp2,count1,energy=0,date;
; idata unsigned char a1p,c1p,e1p,g1p,ap,cp,ep,gp,l;
; idata unsigned int be,de,fe;
; idata unsigned char ae,ce,ge,he,ee;
; idata unsigned int bt,dt,ft;
; idata unsigned char at,ct,gt,ht,et;
;
; main()

                        RSEG  ?PR?main?KCT_PF
main:
                        USING         0
                                      ; SOURCE LINE # 59
; {
                                      ; SOURCE LINE # 60
; lcd_init();
                                      ; SOURCE LINE # 61
                        LCALL         lcd_init
; porta=0x80;
                                      ; SOURCE LINE # 62
; read();
```

```
                                        ; SOURCE LINE # 63
                          LCALL         L?0059
; delay();
                                        ; SOURCE LINE # 64
                          LCALL         delay
; lcd_dis("   PARAMETER    ",16);
                                        ; SOURCE LINE # 65
                          MOV           R2,#HIGH (?SC_0)
                          MOV           R1,#LOW (?SC_0)
                          LCALL         L?0062
; porta=0xc0;
                                        ; SOURCE LINE # 66
                          MOV           R0,#LOW (porta)
                          MOV           A,#0C0H
                          MOVX          @R0,A
; read();
                                        ; SOURCE LINE # 67
                          LCALL         read
; delay();
                                        ; SOURCE LINE # 68
                          LCALL         delay
; lcd_dis("    DISPLAY     ",16);
                                        ; SOURCE LINE # 69
                          MOV           R3,#0FFH
                          MOV           R2,#HIGH (?SC_17)
                          MOV           R1,#LOW (?SC_17)
                          MOV           R5,#010H
                          LCALL         _lcd_dis
; del();
                                        ; SOURCE LINE # 70
                          LCALL         del
; del();
                                        ; SOURCE LINE # 71
                          LCALL         del
; del();
                                        ; SOURCE LINE # 72
                          LCALL         del
; del();
                                        ; SOURCE LINE # 73
                          LCALL         del
; del();
                                        ; SOURCE LINE # 74
                          LCALL         del
; del();
                                        ; SOURCE LINE # 75
                          LCALL         del
; porta=0x80;
                                        ; SOURCE LINE # 76
; read();
                                        ; SOURCE LINE # 77
                          LCALL         L?0059
; delay();
                                        ; SOURCE LINE # 78
                          LCALL         delay
; lcd_dis("                ",16);
                                        ; SOURCE LINE # 79
                          LCALL         L?0061
; porta=0xc0;
                                        ; SOURCE LINE # 80
                          MOV           R0,#LOW (porta)
                          MOV           A,#0C0H
                          MOVX          @R0,A
; read();
                                        ; SOURCE LINE # 81
                          LCALL         read
; delay();
                                        ; SOURCE LINE # 82
                          LCALL         delay
; lcd_dis("                ",16);
                                        ; SOURCE LINE # 83
                          LCALL         L?0061
; EA=1;
                                        ; SOURCE LINE # 84
                          SETB          EA
; TH0=0x00;
                                        ; SOURCE LINE # 85
                          CLR             A
                          MOV             TH0,A
; TL0=0x00;
                                          ; SOURCE LINE # 86
                          MOV             TL0,A
; TMOD=0x01;
                                          ; SOURCE LINE # 87
                          MOV             TMOD,#01H
; ET0=1;
                                          ; SOURCE LINE # 88
                          SETB            ET0
; TR0=1;
                                          ; SOURCE LINE # 89
                          SETB            TR0
?C0001:
;
; while(1)
                                          ; SOURCE LINE # 91
; {
                                          ; SOURCE LINE # 92
; ser_init();
                                          ; SOURCE LINE # 93
                          LCALL           ser_init
; ser_data=0xff;
                                          ; SOURCE LINE # 94
                          MOV             R0,#LOW (ser_data)
                          MOV             @R0,#0FFH
; ser_out();
                                          ; SOURCE LINE # 95
                          LCALL           ser_out
;
; for(ix=1;ix<=50;ix++)
                                          ; SOURCE LINE # 97
                          MOV             R0,#LOW (ix)
                          MOV             @R0,#00H
                          INC             R0
                          MOV             @R0,#01H
?C0003:
; {
                                          ; SOURCE LINE # 98
?C0006:
; while(pul==0);
                                          ; SOURCE LINE # 99
                          JNB             pul,?C0006
?C0008:
; while(pul==1);
                                          ; SOURCE LINE # 100
                          JB    pul,?C0008
?C0010:
; while(pul==0);
                                          ; SOURCE LINE # 101
                          JNB             pul,?C0010
;
; bb:
                                          ; SOURCE LINE # 103
?main?bb:
; if(pul==1)
                                          ; SOURCE LINE # 104
                          JNB             pul,?C0005
; {
                                          ; SOURCE LINE # 105
; countr=countr+1;
                                          ; SOURCE LINE # 106
                          MOV             R0,#LOW (countr+01H)
                          INC             @R0
                          MOV             A,@R0
                          DEC             R0
                          JNZ             ?main?bb
                          INC             @R0
?C0054:
; goto bb;
                                          ; SOURCE LINE # 107
                          SJMP            ?main?bb
; }
                                          ; SOURCE LINE # 108
; }
```

```
                                ; SOURCE LINE # 109
?C0005:
                    MOV         R0,#LOW (ix+01H)
                    INC         @R0
                    MOV         A,@R0
                    DEC         R0
                    JNZ         ?C0055
                    INC         @R0
?C0055:
                    MOV         R0,#LOW (ix+01H)
                    MOV         A,@R0
                    XRL         A,#033H
                    DEC         R0
                    ORL         A,@R0
                    JNZ         ?C0003
?C0004:
; count1=countr/50;
                                ; SOURCE LINE # 110
                    MOV         R0,#LOW (countr)
                    MOV         A,@R0
                    MOV         R6,A
                    INC         R0
                    MOV         A,@R0
                    MOV         R7,A
                    MOV         R4,#00H
                    MOV         R5,#032H
                    LCALL       ?C?UIDIV
                    MOV         count1,R6
                    MOV         count1+01H,R7
; cal();
                                ; SOURCE LINE # 111
                    LCALL       cal
; if(ttemp1>10000)
                                ; SOURCE LINE # 112
                    SETB        C
                    MOV         A,ttemp1+01H
                    SUBB        A,#010H
                    MOV         A,ttemp1
                    SUBB        A,#027H
                    JC   ?C0014
; {
                                ; SOURCE LINE # 113
; ttemp1=10000;
                                ; SOURCE LINE # 114
                    MOV         ttemp1,#027H
                    MOV         ttemp1+01H,#010H
; ttemp2=1;
                                ; SOURCE LINE # 115
                    MOV         ttemp2,#00H
                    MOV         ttemp2+01H,#01H
; }
                                ; SOURCE LINE # 116
?C0014:
; if(ttemp1<2000)
                                ; SOURCE LINE # 117
                    CLR         C
                    MOV         A,ttemp1+01H
                    SUBB        A,#0D0H
                    MOV         A,ttemp1
                    SUBB        A,#07H
                    JNC         ?C0015
; {
                                ; SOURCE LINE # 118
; ttemp1=10000;
                                ; SOURCE LINE # 119
                    MOV         ttemp1,#027H
                    MOV         ttemp1+01H,#010H
; ttemp2=1;
                                ; SOURCE LINE # 120
                    MOV         ttemp2,#00H
                    MOV         ttemp2+01H,#01H
; }
                                ; SOURCE LINE # 121
?C0015:
; tt=ttemp1;
                                ; SOURCE LINE # 122
```

```
                        MOV         R0,#LOW (tt)
                        MOV         @R0,ttemp1
                        INC         R0
                        MOV         @R0,ttemp1+01H
; htdp();
                                    ; SOURCE LINE # 123
                        LCALL       htdp
; disr();
                                    ; SOURCE LINE # 124
                        LCALL       disr
; mask=tt;
                                    ; SOURCE LINE # 125
                        MOV         R0,#LOW (tt)
                        MOV         A,@R0
                        MOV         mask,A
                        INC         R0
                        MOV         A,@R0
                        MOV         mask+01H,A
; ser_out2();
                                    ; SOURCE LINE # 126
                        LCALL       ser_out2

;
;  adc0();
                                    ; SOURCE LINE # 128
                        LCALL       adc0
;  htd0();
                                    ; SOURCE LINE # 129
                        LCALL       htd0
;  dis0();
                                    ; SOURCE LINE # 130
                        LCALL       dis0
;  del();
                                    ; SOURCE LINE # 131
                        LCALL       del
;  ser_out0();
                                    ; SOURCE LINE # 132
                        LCALL       ser_out0

;
;  adc1();
                                    ; SOURCE LINE # 134
                        LCALL       adc1
;  htd1();
                                    ; SOURCE LINE # 135
                        LCALL       htd1
;  dis1();
                                    ; SOURCE LINE # 136
                        LCALL       dis1
;  del();
                                    ; SOURCE LINE # 137
                        LCALL       del
;  ser_out1();
                                    ; SOURCE LINE # 138
                        LCALL       ser_out1

;
;  del();
                                    ; SOURCE LINE # 140
                        LCALL       del
;  del();
                                    ; SOURCE LINE # 141
                        LCALL       del
;  del();
                                    ; SOURCE LINE # 142
                        LCALL       del
;  del();
                                    ; SOURCE LINE # 143
                        LCALL       del
;  del();
                                    ; SOURCE LINE # 144
                        LCALL       del
;  del();
                                    ; SOURCE LINE # 145
                        LCALL       del
;  del();
                                    ; SOURCE LINE # 146
                        LCALL       del
;  del();
```

```
                                   ; SOURCE LINE # 147
                       LCALL       del
;
; power();
                                   ; SOURCE LINE # 149
                       LCALL       power
; powf();
                                   ; SOURCE LINE # 150
                       LCALL       powf
; htd2();
                                   ; SOURCE LINE # 151
                       LCALL       htd2
; dis3();
                                   ; SOURCE LINE # 152
                       LCALL       dis3
; mask=dat;
                                   ; SOURCE LINE # 153
                       MOV         mask,dat
                       MOV         mask+01H,dat+01H
; ser_out2();
                                   ; SOURCE LINE # 154
                       LCALL       ser_out2
;
; date=energy;
                                   ; SOURCE LINE # 156
                       MOV         date,energy
                       MOV         date+01H,energy+01H
; htd3();
                                   ; SOURCE LINE # 157
                       LCALL       htd3
; dis2();
                                   ; SOURCE LINE # 158
                       LCALL       dis2
; mask=date;
                                   ; SOURCE LINE # 159
                       MOV         mask,date
                       MOV         mask+01H,date+01H
; ser_out2();
                                   ; SOURCE LINE # 160
                       LCALL       ser_out2
; del();
                                   ; SOURCE LINE # 161
                       LCALL       del
; del();
                                   ; SOURCE LINE # 162
                       LCALL       del
; del();
                                   ; SOURCE LINE # 163
                       LCALL       del
; del();
                                   ; SOURCE LINE # 164
                       LCALL       del
; del();
                                   ; SOURCE LINE # 165
                       LCALL       del
;
; }
                                   ; SOURCE LINE # 167
                       LJMP        ?C0001
; END OF main

CSEG                   AT   0000BH
                       LJMP        timer0

; }
;
; void timer0(void) interrupt 1

                       RSEG  ?PR?timer0?KCT_PF
                       USING       0
timer0:
                       PUSH        ACC
                       PUSH        B
                       PUSH        PSW
                       MOV         PSW,#00H
                       PUSH        AR0
```

```
                                PUSH            AR4
                                PUSH            AR5
                                PUSH            AR6
                                PUSH            AR7
                                USING           0
                                                ; SOURCE LINE # 170
; {
;  unsigned char reg1;
;  TR0=0;
                                                ; SOURCE LINE # 173
                                CLR             TR0
;  reg1++;
                                                ; SOURCE LINE # 174
                                INC             reg1?140
;  if(reg1==0x10)
                                                ; SOURCE LINE # 175
                                MOV             A,reg1?140
                                CJNE            A,#010H,?C0017
;  {
                                                ; SOURCE LINE # 176
;   reg1=0;
                                                ; SOURCE LINE # 177
                                MOV             reg1?140,#00H
;   energy=energy+(pow_f/3600);
                                                ; SOURCE LINE # 178
                                MOV             R6,pow_f
                                MOV             R7,pow_f+01H
                                MOV             R4,#0EH
                                MOV             R5,#010H
                                LCALL           ?C?UIDIV
                                MOV             A,R7
                                ADD             A,energy+01H
                                MOV             energy+01H,A
                                MOV             A,R6
                                ADDC            A,energy
                                MOV             energy,A
;  }
                                                ; SOURCE LINE # 179
?C0017:
;  TH0=0x00;
                                                ; SOURCE LINE # 180
                                MOV             TH0,#00H
;  TL0=0x00;
                                                ; SOURCE LINE # 181
                                MOV             TL0,#00H
;  TR0=1;
                                                ; SOURCE LINE # 182
                                SETB            TR0
; }
                                                ; SOURCE LINE # 183
                                POP             AR7
                                POP             AR6
                                POP             AR5
                                POP             AR4
                                POP             AR0
                                POP             PSW
                                POP             B
                                POP             ACC
                                RETI
; END OF timer0

;
;
; void ser_init()

                                RSEG  ?PR?ser_init?KCT_PF
ser_init:
                                USING           0
                                                ; SOURCE LINE # 186
; {
                                                ; SOURCE LINE # 187
;  TH1=0x72;
                                                ; SOURCE LINE # 188
                                MOV             TH1,#072H
;  TMOD=0x20;
                                                ; SOURCE LINE # 189
```

```
                        MOV         TMOD,#020H
; TR1=1;
                                    ; SOURCE LINE # 190
                        SETB        TR1
; delay();
                                    ; SOURCE LINE # 191
                        LCALL       delay
; SCON=0x40;
                                    ; SOURCE LINE # 192
                        MOV         SCON,#040H
; }
                                    ; SOURCE LINE # 193
                        RET
; END OF ser_init

;
; void ser_out()

                        RSEG  ?PR?ser_out?KCT_PF
ser_out:
                        USING       0
                                    ; SOURCE LINE # 195
; {
                                    ; SOURCE LINE # 196
; SBUF=ser_data;
                                    ; SOURCE LINE # 197
                        MOV         R0,#LOW (ser_data)
                        MOV         A,@R0
                        MOV         SBUF,A
; delay();
                                    ; SOURCE LINE # 198
                        LCALL       delay
; SCON=0x40;
                                    ; SOURCE LINE # 199
                        MOV         SCON,#040H
; }
                                    ; SOURCE LINE # 200
                        RET
; END OF ser_out

;
; void adc0()

                        RSEG  ?PR?adc0?KCT_PF
adc0:
                        USING       0
                                    ; SOURCE LINE # 202
; {
                                    ; SOURCE LINE # 203
; x=ch0;
                                    ; SOURCE LINE # 204
                        MOV         R0,#LOW (ch0)
                        MOVX        A,@R0
                        MOV         R0,#LOW (x)
                        MOV         @R0,A
; delay();
                                    ; SOURCE LINE # 205
                        LCALL       delay
; dat0=soc;
                                    ; SOURCE LINE # 206
                        MOV         R0,#LOW (soc)
                        MOVX        A,@R0
                        MOV         R0,#LOW (dat0)
                        MOV         @R0,A
; }
                                    ; SOURCE LINE # 207
                        RET
; END OF adc0

;
; void htd0()

                        RSEG  ?PR?htd0?KCT_PF
htd0:
                        USING       0
                                    ; SOURCE LINE # 209
; {
                                   ; SOURCE LINE # 210
;  a0=dat0/0x64;
                                   ; SOURCE LINE # 211
                       MOV         R0,#LOW (dat0)
                       MOV         A,@R0
                       MOV         R7,A
                       MOV         B,#064H
                       DIV         AB
                       MOV         R0,#LOW (a0)
                       MOV         @R0,A
;  b0=dat0%0x64;
                                   ; SOURCE LINE # 212
                       MOV         A,R7
                       MOV         B,#064H
                       DIV         AB
                       MOV         R0,#LOW (b0)
                       MOV         @R0,B
;  c0=b0/0x0a;
                                   ; SOURCE LINE # 213
                       MOV         A,@R0
                       MOV         R7,A
                       MOV         B,#0AH
                       DIV         AB
                       MOV         R0,#LOW (c0)
                       MOV         @R0,A
;  d0=b0%0x0a;
                                   ; SOURCE LINE # 214
                       MOV         A,R7
                       MOV         B,#0AH
                       DIV         AB
                       MOV         R0,#LOW (d0)
                       MOV         @R0,B
; }
                                   ; SOURCE LINE # 215
                       RET
; END OF htd0

;
; void dis0()

                       RSEG  ?PR?dis0?KCT_PF
dis0:
                       USING       0
                                   ; SOURCE LINE # 217
; {
                                   ; SOURCE LINE # 218
;  porta=0x89;
                                   ; SOURCE LINE # 219
                       MOV         R0,#LOW (porta)
                       MOV         A,#089H
                       MOVX        @R0,A
;  read();
                                   ; SOURCE LINE # 220
                       LCALL       read
;  delay();
                                   ; SOURCE LINE # 221
                       LCALL       delay
;  porta='V';
                                   ; SOURCE LINE # 222
                       MOV         R0,#LOW (porta)
                       MOV         A,#056H
                       MOVX        @R0,A
;  write();
                                   ; SOURCE LINE # 223
                       LCALL       write
;  delay();
                                   ; SOURCE LINE # 224
                       LCALL       delay
;  porta=':';
                                   ; SOURCE LINE # 225
                       MOV         R0,#LOW (porta)
                       MOV         A,#03AH
                       MOVX        @R0,A
;  write();
                                   ; SOURCE LINE # 226
```

```
                        LCALL       write
; delay();
                                    ; SOURCE LINE # 227
                        LCALL       delay
; porta=a0+0x30;
                                    ; SOURCE LINE # 228
                        MOV         R0,#LOW (a0)
                        MOV         A,@R0
                        ADD         A,#030H
                        MOV         R1,#LOW (porta)
                        MOVX        @R1,A
; write();
                                    ; SOURCE LINE # 229
                        LCALL       write
; delay();
                                    ; SOURCE LINE # 230
                        LCALL       delay
; porta=c0+0x30;
                                    ; SOURCE LINE # 231
                        MOV         R0,#LOW (c0)
                        MOV         A,@R0
                        ADD         A,#030H
                        MOV         R1,#LOW (porta)
                        MOVX        @R1,A
; write();
                                    ; SOURCE LINE # 232
                        LCALL       write
; delay();
                                    ; SOURCE LINE # 233
                        LCALL       delay
; porta=d0+0x30;
                                    ; SOURCE LINE # 234
                        MOV         R0,#LOW (d0)
                        MOV         A,@R0
                        ADD         A,#030H
                        MOV         R1,#LOW (porta)
                        MOVX        @R1,A
; write();
                                    ; SOURCE LINE # 235
                        LCALL       write
; delay();
                                    ; SOURCE LINE # 236
                        LCALL       delay
; porta=' ';
                                    ; SOURCE LINE # 237
; write();
                                    ; SOURCE LINE # 238
                        LCALL       L?0057
; delay();
                                    ; SOURCE LINE # 239
                        LCALL       delay
; porta=' ';
                                    ; SOURCE LINE # 240
; write();
                                    ; SOURCE LINE # 241
                        LCALL       L?0057
; delay();
                                    ; SOURCE LINE # 242
                        LCALL       delay
; porta=' ';
                                    ; SOURCE LINE # 243
; write();
                                    ; SOURCE LINE # 244
                        LCALL       L?0058
; delay();
                                    ; SOURCE LINE # 245
                        LCALL       delay
; porta=' ';
                                    ; SOURCE LINE # 246
; write();
                                    ; SOURCE LINE # 247
                        LCALL       L?0058
; delay();
                                    ; SOURCE LINE # 248
                        LCALL       delay
; porta=' ';
```

```
                                        ; SOURCE LINE # 249
;  write();
                                        ; SOURCE LINE # 250
                          LCALL         L?0058
;  delay();
                                        ; SOURCE LINE # 251
                          LCALL         delay
;  porta=' ';
                                        ; SOURCE LINE # 252
;  write();
                                        ; SOURCE LINE # 253
                          LCALL         L?0058
;  delay();
                                        ; SOURCE LINE # 254
                          LCALL         delay
;  porta=' ';
                                        ; SOURCE LINE # 255
;  write();
                                        ; SOURCE LINE # 256
                          LCALL         L?0058
;  delay();
                                        ; SOURCE LINE # 257
                          LJMP          delay
; END OF dis0

; }
;
; void ser_out0()

                          RSEG  ?PR?ser_out0?KCT_PF
ser_out0:
                          USING         0
                                        ; SOURCE LINE # 260
; {
                                        ; SOURCE LINE # 261
;  ser_data=dat0;
                                        ; SOURCE LINE # 262
                          MOV           R0,#LOW (dat0)
                          MOV           A,@R0
                          MOV           R0,#LOW (ser_data)
                          MOV           @R0,A
;  ser_out();
                                        ; SOURCE LINE # 263
                          LJMP          ser_out
; END OF ser_out0

; }
;
; void adc1()

                          RSEG  ?PR?adc1?KCT_PF
adc1:
                          USING         0
                                        ; SOURCE LINE # 266
; {
                                        ; SOURCE LINE # 267
;  x=ch1;
                                        ; SOURCE LINE # 268
                          MOV           R0,#LOW (ch1)
                          MOVX          A,@R0
                          MOV           R0,#LOW (x)
                          MOV           @R0,A
;  delay();
                                        ; SOURCE LINE # 269
                          LCALL         delay
;  dat1=soc;
                                        ; SOURCE LINE # 270
                          MOV           R0,#LOW (soc)
                          MOVX          A,@R0
                          MOV           R0,#LOW (dat1)
                          MOV           @R0,A
; }
                                        ; SOURCE LINE # 271
                          RET
; END OF adc1
```

```
;
; void htd1()

                        RSEG  ?PR?htd1?KCL_PF
htd1:
                        USING       0
                                    ; SOURCE LINE # 273
; {
                                    ; SOURCE LINE # 274
;  a1=dat1/0x64;
                                    ; SOURCE LINE # 275
                        MOV         R0,#LOW (dat1)
                        MOV         A,@R0
                        MOV         R7,A
                        MOV         B,#064H
                        DIV         AB
                        MOV         R0,#LOW (a1)
                        MOV         @R0,A
;  b1=dat1%0x64;
                                    ; SOURCE LINE # 276
                        MOV         A,R7
                        MOV         B,#064H
                        DIV         AB
                        MOV         R0,#LOW (b1)
                        MOV         @R0,B
;  c1=b1/0x0a;
                                    ; SOURCE LINE # 277
                        MOV         A,@R0
                        MOV         R7,A
                        MOV         B,#0AH
                        DIV         AB
                        MOV         R0,#LOW (c1)
                        MOV         @R0,A
;  d1=b1%0x0a;
                                    ; SOURCE LINE # 278
                        MOV         A,R7
                        MOV         B,#0AH
                        DIV         AB
                        MOV         R0,#LOW (d1)
                        MOV         @R0,B
; }
                                    ; SOURCE LINE # 279
                        RET
; END OF htd1

;
; void dis1()

                        RSEG  ?PR?dis1?KCL_PF
dis1:
                        USING       0
                                    ; SOURCE LINE # 281
; {
                                    ; SOURCE LINE # 282
;  porta=0xc0;
                                    ; SOURCE LINE # 283
                        MOV         R0,#LOW (porta)
                        MOV         A,#0C0H
                        MOVX        @R0,A
;  read();
                                    ; SOURCE LINE # 284
                        LCALL       read
;  delay();
                                    ; SOURCE LINE # 285
                        LCALL       delay
;  porta='I';
                                    ; SOURCE LINE # 286
                        MOV         R0,#LOW (porta)
                        MOV         A,#049H
                        MOVX        @R0,A
;  write();
                                    ; SOURCE LINE # 287
                        LCALL       write
;  delay();
                                    ; SOURCE LINE # 288
                        LCALL       delay
```

```
;  porta=':';
                                    ; SOURCE LINE # 289
                        MOV         R0,#LOW (porta)
                        MOV         A,#03AH
                        MOVX        @R0,A
;  write();
                                    ; SOURCE LINE # 290
                        LCALL       write
;  delay();
                                    ; SOURCE LINE # 291
                        LCALL       delay
;  porta=a1+0x30;
                                    ; SOURCE LINE # 292
                        MOV         R0,#LOW (a1)
                        MOV         A,@R0
                        ADD         A,#030H
                        MOV         R1,#LOW (porta)
                        MOVX        @R1,A
;  write();
                                    ; SOURCE LINE # 293
                        LCALL       write
;  delay();
                                    ; SOURCE LINE # 294
                        LCALL       delay
;  porta=c1+0x30;
                                    ; SOURCE LINE # 295
                        MOV         R0,#LOW (c1)
                        MOV         A,@R0
                        ADD         A,#030H
                        MOV         R1,#LOW (porta)
                        MOVX        @R1,A
;  write();
                                    ; SOURCE LINE # 296
                        LCALL       write
;  delay();
                                    ; SOURCE LINE # 297
                        LCALL       delay
;  porta='.';
                                    ; SOURCE LINE # 298
                        MOV         R0,#LOW (porta)
                        MOV         A,#02EH
                        MOVX        @R0,A
;  write();
                                    ; SOURCE LINE # 299
                        LCALL       write
;  delay();
                                    ; SOURCE LINE # 300
                        LCALL       delay
;  porta=d1+0x30;
                                    ; SOURCE LINE # 301
                        MOV         R0,#LOW (d1)
                        MOV         A,@R0
                        ADD         A,#030H
                        MOV         R1,#LOW (porta)
                        MOVX        @R1,A
;  write();
                                    ; SOURCE LINE # 302
                        LCALL       write
;  delay();
                                    ; SOURCE LINE # 303
                        LCALL       delay
;  porta=' ';
                                    ; SOURCE LINE # 304
;  write();
                                    ; SOURCE LINE # 305
                        LCALL       L?0058
;  delay();
                                    ; SOURCE LINE # 306
                        LCALL       delay
;  porta=' ';
                                    ; SOURCE LINE # 307
;  write();
                                    ; SOURCE LINE # 308
                        LCALL       L?0058
;  delay();
                                    ; SOURCE LINE # 309
                        LCALL       delay
; porta=' ';
                                    ; SOURCE LINE # 310
; write();
                                    ; SOURCE LINE # 311
                        LCALL       L?0058
; delay();
                                    ; SOURCE LINE # 312
                        LCALL       delay
; porta=' ';
                                    ; SOURCE LINE # 313
; write();
                                    ; SOURCE LINE # 314
                        LCALL       L?0058
; delay();
                                    ; SOURCE LINE # 315
                        LCALL       delay
; porta=' ';
                                    ; SOURCE LINE # 316
; write();
                                    ; SOURCE LINE # 317
                        LCALL       L?0058
; delay();
                                    ; SOURCE LINE # 318
                        LCALL       delay
; porta=' ';
                                    ; SOURCE LINE # 319
; write();
                                    ; SOURCE LINE # 320
                        LCALL       L?0058
; delay();
                                    ; SOURCE LINE # 321
                        LCALL       delay
; porta=' ';
                                    ; SOURCE LINE # 322
; write();
                                    ; SOURCE LINE # 323
                        LCALL       L?0058
; delay();
                                    ; SOURCE LINE # 324
                        LCALL       delay
; porta=' ';
                                    ; SOURCE LINE # 325
; write();
                                    ; SOURCE LINE # 326
                        LCALL       L?0058
; delay();
                                    ; SOURCE LINE # 327
                        LJMP        delay
; END OF dis1

; }
;
; void ser_out1()

                        RSEG  ?PR?ser_out1?KCT_PF
ser_out1:
                        USING       0
                                    ; SOURCE LINE # 330
; {
                                    ; SOURCE LINE # 331
; ser_data=dat1;
                                    ; SOURCE LINE # 332
                        MOV         R0,#LOW (dat1)
                        MOV         A,@R0
                        MOV         R0,#LOW (ser_data)
                        MOV         @R0,A
; ser_out();
                                    ; SOURCE LINE # 333
                        LJMP        ser_out
; END OF ser_out1

; }
;
; void power()
```

```
                        RSEG  ?PR?power?KCT_PF
power:
                        USING           0
                                        ; SOURCE LINE # 336
; {
                                        ; SOURCE LINE # 337
; float volt,cur;
;  volt=dat0;
                                        ; SOURCE LINE # 339
                        MOV             R0,#LOW (dat0)
                        MOV             A,@R0
                        MOV             R4,A
                        CLR             A
                        LCALL           ?C?FCASTC
                        MOV             volt?1241+03H,R7
                        MOV             volt?1241+02H,R6
                        MOV             volt?1241+01H,R5
                        MOV             volt?1241,R4
;  cur=(float)dat1/10;
                                        ; SOURCE LINE # 340
                        MOV             R0,#LOW (dat1)
                        MOV             A,@R0
                        MOV             R4,A
                        CLR             A
                        LCALL           ?C?FCASTC
                        CLR             A
                        MOV             R3,A
                        MOV             R2,A
                        MOV             R1,#020H
                        MOV             R0,#041H
                        LCALL           ?C?FPDIV
;---- Variable 'cur?1242' assigned to Register 'R4/R5/R6/R7' ----
;  pow=volt*cur;
                                        ; SOURCE LINE # 341
                        MOV             R3,volt?1241+03H
                        MOV             R2,volt?1241+02H
                        MOV             R1,volt?1241+01H
                        MOV             R0,volt?1241
                        LCALL           ?C?FPMUL
                        LCALL           ?C?CASTF
                        MOV             pow,R6
                        MOV             pow+01H,R7
;  dat=pow*100;
                                        ; SOURCE LINE # 342
                        MOV             R4,#00H
                        MOV             R5,#064H
                        LCALL           ?C?IMUL
                        MOV             dat,R6
                        MOV             dat+01H,R7
; }
                                        ; SOURCE LINE # 343
                        RET
; END OF power

;
; void htd2()

                        RSEG  ?PR?htd2?KCT_PF
htd2:
                        USING           0
                                        ; SOURCE LINE # 345
; {
                                        ; SOURCE LINE # 346
;  a2=dat/0x2710;
                                        ; SOURCE LINE # 347
                        MOV             R6,dat
                        MOV             R7,dat+01H
                        MOV             R4,#027H
                        MOV             R5,#010H
                        LCALL           ?C?UIDIV
                        MOV             R0,#LOW (a2)
                        MOV             @R0,AR7
;  b2=(unsigned long)dat%0x2710;
                                        ; SOURCE LINE # 348
                        MOV             R6,dat
                        MOV             R7,dat+01H
```

```
                    CLR       A
                    MOV       R4,A
                    MOV       R5,A
                    MOV       R3,#010H
                    MOV       R2,#027H
                    MOV       R1,A
                    MOV       R0,A
                    LCALL     ?C?UIDIV
                    MOV       R6,AR2
                    MOV       R7,AR3
                    MOV       R0,#LOW (b2)
                    MOV       @R0,AR6
                    INC       R0
                    MOV       @R0,AR7
; c2=b2/0x3e8;
                              ; SOURCE LINE # 349
                    DEC       R0
                    MOV       A,@R0
                    MOV       R2,A
                    INC       R0
                    MOV       A,@R0
                    MOV       R3,A
                    MOV       R4,#03H
                    MOV       R5,#0E8H
                    MOV       R7,A
                    MOV       R6,AR2
                    LCALL     ?C?UIDIV
                    MOV       R0,#LOW (c2)
                    MOV       @R0,AR7
; d2=b2%0x3e8;
                              ; SOURCE LINE # 350
                    MOV       R4,#03H
                    MOV       R5,#0E8H
                    MOV       R7,AR3
                    MOV       R6,AR2
                    LCALL     ?C?UIDIV
                    MOV       R0,#LOW (d2)
                    MOV       @R0,AR4
                    INC       R0
                    MOV       @R0,AR5
; e2=d2/0x64;
                              ; SOURCE LINE # 351
                    DEC       R0
                    MOV       A,@R0
                    MOV       R2,A
                    INC       R0
                    MOV       A,@R0
                    MOV       R3,A
                    MOV       R4,#00H
                    MOV       R5,#064H
                    MOV       R7,A
                    MOV       R6,AR2
                    LCALL     ?C?UIDIV
                    MOV       R0,#LOW (e2)
                    MOV       @R0,AR7
; f2=d2%0x64;
                              ; SOURCE LINE # 352
                    MOV       R4,#00H
                    MOV       R5,#064H
                    MOV       R7,AR3
                    MOV       R6,AR2
                    LCALL     ?C?UIDIV
                    MOV       R0,#LOW (f2)
                    MOV       @R0,AR5
; g2=f2/0x0a;
                              ; SOURCE LINE # 353
                    MOV       A,@R0
                    MOV       R7,A
                    MOV       B,#0AH
                    DIV       AB
                    INC       R0
                    MOV       @R0,A
; h2=f2%0x0a;
                              ; SOURCE LINE # 354
                    MOV       A,R7
                    MOV       B,#0AH
```

```
                        DIV             AB
                        INC             R0
                        MOV             @R0,B
; }
                                        ; SOURCE LINE # 355
                        RET
; END OF htd2

;
; void htd3()

                        RSEG  ?PR?htd3?KCF_PF
htd3:
                        USING           0
                                        ; SOURCE LINE # 357
; {
                                        ; SOURCE LINE # 358
;  ae=date/0x2710;
                                        ; SOURCE LINE # 359
                        MOV             R6,date
                        MOV             R7,date+01H
                        MOV             R4,#027H
                        MOV             R5,#010H
                        LCALL           ?C?UIDIV
                        MOV             R0,#LOW (ae)
                        MOV             @R0,AR7
;  be=(unsigned long)date%0x2710;
                                        ; SOURCE LINE # 360
                        MOV             R6,date
                        MOV             R7,date+01H
                        CLR             A
                        MOV             R4,A
                        MOV             R5,A
                        MOV             R3,#010H
                        MOV             R2,#027H
                        MOV             R1,A
                        MOV             R0,A
                        LCALL           ?C?ULDIV
                        MOV             R6,AR2
                        MOV             R7,AR3
                        MOV             R0,#LOW (be)
                        MOV             @R0,AR6
                        INC             R0
                        MOV             @R0,AR7
;  ce=be/0x3e8;
                                        ; SOURCE LINE # 361
                        DEC             R0
                        MOV             A,@R0
                        MOV             R2,A
                        INC             R0
                        MOV             A,@R0
                        MOV             R3,A
                        MOV             R4,#03H
                        MOV             R5,#0E8H
                        MOV             R7,A
                        MOV             R6,AR2
                        LCALL           ?C?UIDIV
                        MOV             R0,#LOW (ce)
                        MOV             @R0,AR7
;  de=be%0x3e8;
                                        ; SOURCE LINE # 362
                        MOV             R4,#03H
                        MOV             R5,#0E8H
                        MOV             R7,AR3
                        MOV             R6,AR2
                        LCALL           ?C?UIDIV
                        MOV             R0,#LOW (de)
                        MOV             @R0,AR4
                        INC             R0
                        MOV             @R0,AR5
;  ee=de/0x64;
                                        ; SOURCE LINE # 363
                        DEC             R0
                        MOV             A,@R0
                        MOV             R2,A
                        INC             R0
                        MOV         A,@R0
                        MOV         R3,A
                        MOV         R4,#00H
                        MOV         R5,#064H
                        MOV         R7,A
                        MOV         R6,AR2
                        LCALL       ?C?UIDIV
                        MOV         R0,#LOW (ec)
                        MOV         @R0,AR7
; fe=de%0x64;
                                    ; SOURCE LINE # 364
                        MOV         R4,#00H
                        MOV         R5,#064H
                        MOV         R7,AR3
                        MOV         R6,AR2
                        LCALL       ?C?UIDIV
                        MOV         R0,#LOW (fe)
                        MOV         @R0,AR4
                        INC         R0
                        MOV         @R0,AR5
; ge=fe/0x0a;
                                    ; SOURCE LINE # 365
                        DEC         R0
                        MOV         A,@R0
                        MOV         R2,A
                        INC         R0
                        MOV         A,@R0
                        MOV         R3,A
                        MOV         R4,#00H
                        MOV         R5,#0AH
                        MOV         R7,A
                        MOV         R6,AR2
                        LCALL       ?C?UIDIV
                        MOV         R0,#LOW (ge?)
                        MOV         @R0,AR7
; he=fe%0x0a;
                                    ; SOURCE LINE # 366
                        MOV         R4,#00H
                        MOV         R5,#0AH
                        MOV         R7,AR3
                        MOV         R6,AR2
                        LCALL       ?C?UIDIV
                        MOV         R0,#LOW (he)
                        MOV         @R0,AR5
; }
                                    ; SOURCE LINE # 367
                        RET

; END OF htd3

;
; void dis2()

                        RSEG  ?PR?dis2?KCT_PF

dis2:
                        USING       0
                                    ; SOURCE LINE # 369
; {
                                    ; SOURCE LINE # 370
; porta=0xc0;
                                    ; SOURCE LINE # 371
                        MOV         R0,#LOW (porta)
                        MOV         A,#0C0H
                        MOVX        @R0,A
; read();
                                    ; SOURCE LINE # 372
                        LCALL       read
; delay();
                                    ; SOURCE LINE # 373
                        LCALL       delay
; porta='E';
                                    ; SOURCE LINE # 374
                        MOV         R0,#LOW (porta)
                        MOV         A,#045H
                        MOVX        @R0,A
; write();
                                    ; SOURCE LINE # 375
                        LCALL       write
; delay();
                                    ; SOURCE LINE # 376
                        LCALL       delay
; porta=':';
                                    ; SOURCE LINE # 377
                        MOV         R0,#LOW (porta)
                        MOV         A,#03AH
                        MOVX        @R0,A
; write();
                                    ; SOURCE LINE # 378
                        LCALL       write
; delay();
                                    ; SOURCE LINE # 379
                        LCALL       delay
; porta=ae+0x30;
                                    ; SOURCE LINE # 380
                        MOV         R0,#LOW (ae)
                        MOV         A,@R0
                        ADD         A,#030H
                        MOV         R1,#LOW (porta)
                        MOVX        @R1,A
; write();
                                    ; SOURCE LINE # 381
                        LCALL       write
; delay();
                                    ; SOURCE LINE # 382
                        LCALL       delay
; porta=ce+0x30;
                                    ; SOURCE LINE # 383
                        MOV         R0,#LOW (ce)
                        MOV         A,@R0
                        ADD         A,#030H
                        MOV         R1,#LOW (porta)
                        MOVX        @R1,A
; write();
                                    ; SOURCE LINE # 384
                        LCALL       write
; delay();
                                    ; SOURCE LINE # 385
                        LCALL       delay
; porta=ee+0x30;
                                    ; SOURCE LINE # 386
                        MOV         R0,#LOW (ee)
                        MOV         A,@R0
                        ADD         A,#030H
                        MOV         R1,#LOW (porta)
                        MOVX        @R1,A
; write();
                                    ; SOURCE LINE # 387
                        LCALL       write
; delay();
                                    ; SOURCE LINE # 388
                        LCALL       delay
; porta='.';
                                    ; SOURCE LINE # 389
                        MOV         R0,#LOW (porta)
                        MOV         A,#02EH
                        MOVX        @R0,A
; write();
                                    ; SOURCE LINE # 390
                        LCALL       write
; delay();
                                    ; SOURCE LINE # 391
                        LCALL       delay
; porta=ge+0x30;
                                    ; SOURCE LINE # 392
                        MOV         R0,#LOW (ge?)
                        MOV         A,@R0
                        ADD         A,#030H
                        MOV         R1,#LOW (porta)
                        MOVX        @R1,A
; write();
                                    ; SOURCE LINE # 393
                        LCALL       write
; delay();
```

```
                                ; SOURCE LINE # 394
                    LCALL       delay
;  porta=he+0x30;
                                ; SOURCE LINE # 395
                    MOV         R0,#LOW (he)
                    MOV         A,@R0
                    ADD         A,#030H
                    MOV         R1,#LOW (porta)
                    MOVX        @R1,A
;  write();
                                ; SOURCE LINE # 396
                    LCALL       write
;  delay();
                                ; SOURCE LINE # 397
                    LCALL       delay
;  porta=' ';
                                ; SOURCE LINE # 398
;  write();
                                ; SOURCE LINE # 399
                    LCALL       L?0058
;  delay();
                                ; SOURCE LINE # 400
                    LCALL       delay
;  porta=' ';
                                ; SOURCE LINE # 401
;  write();
                                ; SOURCE LINE # 402
                    LCALL       L?0058
;  delay();
                                ; SOURCE LINE # 403
                    LCALL       delay
;  porta=' ';
                                ; SOURCE LINE # 404
;  write();
                                ; SOURCE LINE # 405
                    LCALL       L?0058
;  delay();
                                ; SOURCE LINE # 406
                    LCALL       delay
;  porta=' ';
                                ; SOURCE LINE # 407
;  write();
                                ; SOURCE LINE # 408
                    LCALL       L?0058
;  delay();
                                ; SOURCE LINE # 409
                    LCALL       delay
;  porta=' ';
                                ; SOURCE LINE # 410
;  write();
                                ; SOURCE LINE # 411
                    LCALL       L?0058
;  delay();
                                ; SOURCE LINE # 412
                    LCALL       delay
;  porta=' ';
                                ; SOURCE LINE # 413
;  write();
                                ; SOURCE LINE # 414
                    LCALL       L?0058
;  delay();
                                ; SOURCE LINE # 415
                    LCALL       delay
;  porta=' ';
                                ; SOURCE LINE # 416
;  write();
                                ; SOURCE LINE # 417
                    LCALL       L?0058
;  delay();
                                ; SOURCE LINE # 418
                    LCALL       delay
;  porta=' ';
                                ; SOURCE LINE # 419
;  write();
                                ; SOURCE LINE # 420
                    LCALL       L?0058
```

```
; delay();
                                        ; SOURCE LINE # 421
                        LJMP            delay
; END OF dis2

; }
;
; void ser_out2()

                        RSEG  ?PR?ser_out2?KCT_PF
ser_out2:
                        USING           0
                                        ; SOURCE LINE # 424
; {
                                        ; SOURCE LINE # 425
; mask1=(unsigned char)mask;
                                        ; SOURCE LINE # 426
                        MOV             mask1,mask+01H
; ser_data=mask1;
                                        ; SOURCE LINE # 427
                        MOV             R0,#LOW (ser_data)
                        MOV             @R0,mask1
; ser_out();
                                        ; SOURCE LINE # 428
                        LCALL           ser_out
; del();
                                        ; SOURCE LINE # 429
                        LCALL           del
; del();
                                        ; SOURCE LINE # 430
                        LCALL           del
; del();
                                        ; SOURCE LINE # 431
                        LCALL           del
; mask2=mask&0xff00;
                                        ; SOURCE LINE # 432
                        MOV             mask2,mask
                        MOV             mask2+01H,#00H
; mask2=mask2>>8;
                                        ; SOURCE LINE # 433
                        MOV             A,mask2
                        MOV             mask2+01H,A
                        MOV             mask2,#00H
; data2=mask2;
                                        ; SOURCE LINE # 434
                        MOV             data2,mask2+01H
; ser_data=data2;
                                        ; SOURCE LINE # 435
                        MOV             R0,#LOW (ser_data)
                        MOV             @R0,data2
; ser_out();
                                        ; SOURCE LINE # 436
                        LCALL           ser_out
; del();
                                        ; SOURCE LINE # 437
                        LCALL           del
; del();
                                        ; SOURCE LINE # 438
                        LCALL           del
; del();
                                        ; SOURCE LINE # 439
                        LJMP            del
; END OF ser_out2

; }
;
;
; void powf()

                        RSEG  ?PR?powf?KCT_PF
powf:
                        USING           0
                                        ; SOURCE LINE # 443
; {
                                        ; SOURCE LINE # 444
;  pow_f=pow*ttemp2;
```

```
                                ; SOURCE LINE # 445
                  MOV           R4,ttemp2
                  MOV           R5,ttemp2+01H
                  MOV           R6,pow
                  MOV           R7,pow+01H
                  LCALL         ?C?IMUL
                  MOV           pow_f,R6
                  MOV           pow_f+01H,R7
; dat=pow_f*100;
                                ; SOURCE LINE # 446
                  MOV           R4,#00H
                  MOV           R5,#064H
                  LCALL         ?C?IMUL
                  MOV           dat,R6
                  MOV           dat+01H,R7
; }
                                ; SOURCE LINE # 447
                  RET
; END OF powf

;
; void dis3()

                  RSEG  ?PR?dis3?KCT_PF
dis3:
                  USING         0
                                ; SOURCE LINE # 449
; {
                                ; SOURCE LINE # 450
; porta=0x80;
                                ; SOURCE LINE # 451
; read();
                                ; SOURCE LINE # 452
                  LCALL         L?0060
; delay();
                                ; SOURCE LINE # 453
                  LCALL         delay
; porta='P';
                                ; SOURCE LINE # 454
                  MOV           R0,#LOW (porta)
                  MOV           A,#050H
                  MOVX          @R0,A
; write();
                                ; SOURCE LINE # 455
                  LCALL         write
; delay();
                                ; SOURCE LINE # 456
                  LCALL         delay
; porta=':';
                                ; SOURCE LINE # 457
                  MOV           R0,#LOW (porta)
                  MOV           A,#03AH
                  MOVX          @R0,A
; write();
                                ; SOURCE LINE # 458
                  LCALL         write
; delay();
                                ; SOURCE LINE # 459
                  LCALL         delay
; porta=a2+0x30;
                                ; SOURCE LINE # 460
                  MOV           R0,#LOW (a2)
                  MOV           A,@R0
                  ADD           A,#030H
                  MOV           R1,#LOW (porta)
                  MOVX          @R1,A
; write();
                                ; SOURCE LINE # 461
                  LCALL         write
; delay();
                                ; SOURCE LINE # 462
                  LCALL         delay
; porta=c2+0x30;
                                ; SOURCE LINE # 463
                  MOV           R0,#LOW (c2)
                  MOV           A,@R0
```

```
                    ADD        A,#030H
                    MOV        R1,#LOW (porta)
                    MOVX       @R1,A
; write();
                               ; SOURCE LINE # 464
                    LCALL      write
; delay();
                               ; SOURCE LINE # 465
                    LCALL      delay
; porta=e2+0x30;
                               ; SOURCE LINE # 466
                    MOV        R0,#LOW (e2)
                    MOV        A,@R0
                    ADD        A,#030H
                    MOV        R1,#LOW (porta)
                    MOVX       @R1,A
; write();
                               ; SOURCE LINE # 467
                    LCALL      write
; delay();
                               ; SOURCE LINE # 468
                    LCALL      delay
; porta='.';
                               ; SOURCE LINE # 469
                    MOV        R0,#LOW (porta)
                    MOV        A,#02EH
                    MOVX       @R0,A
; write();
                               ; SOURCE LINE # 470
                    LCALL      write
; delay();
                               ; SOURCE LINE # 471
                    LCALL      delay
; porta=g2+0x30;
                               ; SOURCE LINE # 472
                    MOV        R0,#LOW (g2)
                    MOV        A,@R0
                    ADD        A,#030H
                    MOV        R1,#LOW (porta)
                    MOVX       @R1,A
; write();
                               ; SOURCE LINE # 473
                    LCALL      write
; delay();
                               ; SOURCE LINE # 474
                    LCALL      delay
; porta=h2+0x30;
                               ; SOURCE LINE # 475
                    MOV        R0,#LOW (h2)
                    MOV        A,@R0
                    ADD        A,#030H
                    MOV        R1,#LOW (porta)
                    MOVX       @R1,A
; write();
                               ; SOURCE LINE # 476
                    LCALL      write
; delay();
                               ; SOURCE LINE # 477
                    LCALL      delay
; porta=' ';
                               ; SOURCE LINE # 478
; write();
                               ; SOURCE LINE # 479
                    LCALL      L?0058
; delay();
                               ; SOURCE LINE # 480
                    LCALL      delay
; porta=' ';
                               ; SOURCE LINE # 481
; write();
                               ; SOURCE LINE # 482
                    LCALL      L?0058
; delay();
                               ; SOURCE LINE # 483
                    LCALL      delay
; porta=' ';
```

```
                                  ; SOURCE LINE # 484
; write();
                                  ; SOURCE LINE # 485
                LCALL             L?0058
; delay();
                                  ; SOURCE LINE # 486
                LCALL             delay
; porta=' ';
                                  ; SOURCE LINE # 487
; write();
                                  ; SOURCE LINE # 488
                LCALL             L?0058
; delay();
                                  ; SOURCE LINE # 489
                LCALL             delay
; porta=' ';
                                  ; SOURCE LINE # 490
; write();
                                  ; SOURCE LINE # 491
                LCALL             L?0058
; delay();
                                  ; SOURCE LINE # 492
                LCALL             delay
; porta=' ';
                                  ; SOURCE LINE # 493
; write();
                                  ; SOURCE LINE # 494
                LCALL             L?0058
; delay();
                                  ; SOURCE LINE # 495
                LCALL             delay
; porta=' ';
                                  ; SOURCE LINE # 496
; write();
                                  ; SOURCE LINE # 497
                LCALL             L?0058
; delay();
                                  ; SOURCE LINE # 498
                LCALL             delay
; porta=' ';
                                  ; SOURCE LINE # 499
; write();
                                  ; SOURCE LINE # 500
                LCALL             L?0058
; delay();
                                  ; SOURCE LINE # 501
                LJMP              delay
; END OF dis3

; }
;
;
; void lcd_init()

                RSEG  ?PR?lcd_init?KCT_PF
lcd_init:
                USING             0
                                  ; SOURCE LINE # 505
; {
                                  ; SOURCE LINE # 506
; cwr=0x80;
                                  ; SOURCE LINE # 507
                MOV               R0,#LOW (cwr)
                MOV               A,#080H
                MOVX              @R0,A
; porta=0x38;
                                  ; SOURCE LINE # 508
                MOV               R0,#LOW (porta)
                MOV               A,#038H
                MOVX              @R0,A
; read();
                                  ; SOURCE LINE # 509
                LCALL             read
; delay();
                                  ; SOURCE LINE # 510
                LCALL             delay
```

```
; porta=0x08;
                                ; SOURCE LINE # 511
                MOV             R0,#LOW (porta)
                MOV             A,#08H
                MOVX            @R0,A
; read();
                                ; SOURCE LINE # 512
                LCALL           read
; delay();
                                ; SOURCE LINE # 513
                LCALL           delay
; porta=0x01;
                                ; SOURCE LINE # 514
                MOV             R0,#LOW (porta)
                MOV             A,#01H
                MOVX            @R0,A
; read();
                                ; SOURCE LINE # 515
                LCALL           read
; delay();
                                ; SOURCE LINE # 516
                LCALL           delay
; porta=0x06;
                                ; SOURCE LINE # 517
                MOV             R0,#LOW (porta)
                MOV             A,#06H
                MOVX            @R0,A
; read();
                                ; SOURCE LINE # 518
                LCALL           read
; delay();
                                ; SOURCE LINE # 519
                LCALL           delay
; porta=0x0c;
                                ; SOURCE LINE # 520
                MOV             R0,#LOW (porta)
                MOV             A,#0CH
                MOVX            @R0,A
; read();
                                ; SOURCE LINE # 521
                LCALL           read
; delay();
                                ; SOURCE LINE # 522
                LCALL           delay
; porta=0x45;
                                ; SOURCE LINE # 523
                MOV             R0,#LOW (porta)
                MOV             A,#045H
                MOVX            @R0,A
; read();
                                ; SOURCE LINE # 524
                LCALL           read
; delay();
                                ; SOURCE LINE # 525
                LJMP            delay
; END OF lcd_init

; }
;
; void lcd_dis(unsigned char *mess,unsigned char n)

                RSEG  ?PR?_lcd_dis?KCT_PF
L?0061:
                USING           0
                MOV             R2,#HIGH (?SC_34)
                MOV             R1,#LOW (?SC_34)
L?0062:
                MOV             R3,#0FFH
                MOV             R5,#010H
_lcd_dis:
                USING           0
                                ; SOURCE LINE # 528
                MOV             mess??043,R3
                MOV             mess??043+01H,R2
                MOV             mess??043+02H,R1
                MOV             n?2044,R5
```

```
; {
                                    ; SOURCE LINE # 529
;  for(i=0;i<n;i++)
                                    ; SOURCE LINE # 530
                        CLR         A
                        MOV         R0,#LOW (i)
                        MOV         @R0,A
?C0037:
                        MOV         R0,#LOW (i)
                        MOV         A,@R0
                        MOV         R7,A
                        CLR         C
                        SUBB        A,n?2044
                        JNC         ?C0040
; {
                                    ; SOURCE LINE # 531
;  porta=mess[i];
                                    ; SOURCE LINE # 532
                        MOV         R3,mess?2043
                        MOV         R2,mess?2043+01H
                        MOV         R1,mess?2043+02H
                        MOV         DPL,R7
                        MOV         DPH,#00H
                        LCALL       ?C?CLDOPTR
                        MOV         R0,#LOW (porta)
                        MOVX        @R0,A
;  write();
                                    ; SOURCE LINE # 533
                        LCALL       write
;  delay();
                                    ; SOURCE LINE # 534
                        LCALL       delay
;  }
                                    ; SOURCE LINE # 535
                        MOV         R0,#LOW (i)
                        INC         @R0
                        SJMP        ?C0037
; }
                                    ; SOURCE LINE # 536
?C0040:
                        RET
; END OF _lcd_dis

;
; void read()

                        RSEG  ?PR?read?KCT PF
L?0059:
                        USING       0
L?0060:
                        MOV         R0,#LOW (porta)
                        MOV         A,#080H
                        MOVX        @R0,A
read:
                        USING       0
                                    ; SOURCE LINE # 538
; {
                                    ; SOURCE LINE # 539
;  portc=0x04;
                                    ; SOURCE LINE # 540
                        MOV         R0,#LOW (portc)
                        MOV         A,#04H
                        MOVX        @R0,A
;  delay();
                                    ; SOURCE LINE # 541
                        LCALL       delay
;  portc=0x00;
                                    ; SOURCE LINE # 542
                        CLR         A
                        MOV         R0,#LOW (portc)
                        MOVX        @R0,A
;  delay();
                                    ; SOURCE LINE # 543
                        LJMP        delay
; END OF read
; }
;
; void write()

                        RSEG  ?PR?write?KCT_PF
L?0057:
                        USING       0
L?0058:
                        MOV         R0,#LOW (porta)
                        MOV         A,#020H
                        MOVX        @R0,A
write:
                        USING       0
                                    ; SOURCE LINE # 546
; {
                                    ; SOURCE LINE # 547
;  portc=0x05;
                                    ; SOURCE LINE # 548
                        MOV         R0,#LOW (portc)
                        MOV         A,#05H
                        MOVX        @R0,A
;  delay();
                                    ; SOURCE LINE # 549
                        LCALL       delay
;  portc=0x01;
                                    ; SOURCE LINE # 550
                        MOV         R0,#LOW (portc)
                        MOV         A,#01H
                        MOVX        @R0,A
;  delay();
                                    ; SOURCE LINE # 551
                        LJMP        delay
; END OF write

; }
;
; void delay()

                        RSEG  ?PR?delay?KCT_PF
delay:
                        USING       0
                                    ; SOURCE LINE # 554
; {
                                    ; SOURCE LINE # 555
;  for(j=0x00;j<=0xfe;j++)
                                    ; SOURCE LINE # 556
                        CLR         A
                        MOV         R0,#LOW (j)
                        MOV         @R0,A
?C0043:
;  {}
                                    ; SOURCE LINE # 557
                        MOV         R0,#LOW (j)
                        INC         @R0
                        MOV         A,@R0
                        SETB        C
                        SUBB        A,#0FEH
                        JC   ?C0043
; }
                                    ; SOURCE LINE # 558
?C0046:
                        RET
; END OF delay

;
; void del()

                        RSEG  ?PR?del?KCT_PF
del:
                        USING       0
                                    ; SOURCE LINE # 560
; {
                                    ; SOURCE LINE # 561
;  for(s=0;s<=30000;s++)
                                    ; SOURCE LINE # 562
                        CLR         A
                        MOV             R0,#LOW (s)
                        MOV             @R0,A
                        INC             R0
                        MOV             @R0,A
?C0047:
; {}
                                        ; SOURCE LINE # 563
                        MOV             R0,#LOW (s+01H)
                        INC             @R0
                        MOV             A,@R0
                        DEC             R0
                        JNZ             ?C0056
                        INC             @R0
?C0056:
                        MOV             R0,#LOW (s)
                        CJNE            @R0,#075H,?C0047
                        INC             R0
                        CJNE            @R0,#031H,?C0047
; }
                                        ; SOURCE LINE # 564
?C0050:
                        RET
; END OF del

;
; void cal()

                        RSEG  ?PR?cal?KCT_PF
cal:
                        USING           0
                                        ; SOURCE LINE # 566
; {
                                        ; SOURCE LINE # 567
; reg=count1;
                                        ; SOURCE LINE # 568
                        MOV             R0,#LOW (reg)
                        MOV             @R0,count1
                        INC             R0
                        MOV             @R0,count1+01H
; count1=count1*216;
                                        ; SOURCE LINE # 569
                        MOV             R6,count1
                        MOV             R7,count1+01H
                        MOV             R4,#00H
                        MOV             R5,#0D8H
                        LCALL           ?C?IMUL
                        MOV             count1,R6
                        MOV             count1+01H,R7
; count1=count1/1000;
                                        ; SOURCE LINE # 570
                        MOV             R4,#03H
                        MOV             R5,#0E8H
                        LCALL           ?C?UIDIV
                        MOV             count1,R6
                        MOV             count1+01H,R7
; ttemp2=((count1*2*3.14)/180);
                                        ; SOURCE LINE # 571
                        MOV             A,count1+01H
                        ADD             A,ACC
                        MOV             R5,A
                        MOV             A,count1
                        RLC             A
                        MOV             R4,A
                        CLR             A
                        LCALL           ?C?FCASTI
                        MOV             R3,#0C3H
                        MOV             R2,#0F5H
                        MOV             R1,#048H
                        MOV             R0,#040H
                        LCALL           ?C?FPMUL
                        CLR             A
                        MOV             R3,A
                        MOV             R2,A
                        MOV             R1,#034H
                        MOV             R0,#043H
                        LCALL           ?C?FPDIV
```

```
                    LCALL       ?C?CASTF
                    MOV         ttemp2,R6
                    MOV         ttemp2+01H,R7
; ttemp2=cos(ttemp2);
                                ; SOURCE LINE # 572
                    MOV         R4,ttemp2
                    MOV         R5,ttemp2+01H
                    CLR         A
                    LCALL       ?C?FCASTI
                    LCALL       _cos
                    LCALL       ?C?CASTF
                    MOV         ttemp2,R6
                    MOV         ttemp2+01H,R7
; ttemp1=ttemp2*10000;
                                ; SOURCE LINE # 573
                    MOV         R4,#027H
                    MOV         R5,#010H
                    LCALL       ?C?IMUL
                    MOV         ttemp1,R6
                    MOV         ttemp1+01H,R7
; }
                                ; SOURCE LINE # 574
                    RET
; END OF cal

;
; void htdp()

                    RSEG  ?PR?htdp?KCTPF
htdp:
                    USING       0
                                ; SOURCE LINE # 576
; {
                                ; SOURCE LINE # 577
;  ap=tt/0x2710;
                                ; SOURCE LINE # 578
                    MOV         R0,#LOW (tt)
                    MOV         A,@R0
                    MOV         R2,A
                    INC         R0
                    MOV         A,@R0
                    MOV         R3,A
                    MOV         R4,#027H
                    MOV         R5,#010H
                    MOV         R7,A
                    MOV         R6,AR2
                    LCALL       ?C?UIDIV
                    MOV         R0,#LOW (ap)
                    MOV         @R0,AR7
;  bp=tt%0x2710;
                                ; SOURCE LINE # 579
                    MOV         R4,#027H
                    MOV         R5,#010H
                    MOV         R7,AR3
                    MOV         R6,AR2
                    LCALL       ?C?UIDIV
                    MOV         R0,#LOW (bp)
                    MOV         @R0,AR4
                    INC         R0
                    MOV         @R0,AR5
;  cp=bp/0x3e8;
                                ; SOURCE LINE # 580
                    DEC         R0
                    MOV         A,@R0
                    MOV         R2,A
                    INC         R0
                    MOV         A,@R0
                    MOV         R3,A
                    MOV         R4,#03H
                    MOV         R5,#0E8H
                    MOV         R7,A
                    MOV         R6,AR2
                    LCALL       ?C?UIDIV
                    MOV         R0,#LOW (cp)
                    MOV         @R0,AR7
;  dp=bp%0x3e8;
```

```
                                        ; SOURCE LINE # 581
                    MOV         R4,#03H
                    MOV         R5,#0E8H
                    MOV         R7,AR3
                    MOV         R6,AR2
                    LCALL       ?C?UIDIV
                    MOV         R0,#LOW (dp)
                    MOV         @R0,AR4
                    INC         R0
                    MOV         @R0,AR5
;  ep=dp/0x64;
                                        ; SOURCE LINE # 582
                    DEC         R0
                    MOV         A,@R0
                    MOV         R2,A
                    INC         R0
                    MOV         A,@R0
                    MOV         R3,A
                    MOV         R4,#00H
                    MOV         R5,#064H
                    MOV         R7,A
                    MOV         R6,AR2
                    LCALL       ?C?UIDIV
                    MOV         R0,#LOW (ep)
                    MOV         @R0,AR7
;  fp=dp%0x64;
                                        ; SOURCE LINE # 583
                    MOV         R4,#00H
                    MOV         R5,#064H
                    MOV         R7,AR3
                    MOV         R6,AR2
                    LCALL       ?C?UIDIV
                    MOV         R0,#LOW (fp)
                    MOV         @R0,AR4
                    INC         R0
                    MOV         @R0,AR5
;  gp=fp/0x0a;
                                        ; SOURCE LINE # 584
                    DEC         R0
                    MOV         A,@R0
                    MOV         R2,A
                    INC         R0
                    MOV         A,@R0
                    MOV         R3,A
                    MOV         R4,#00H
                    MOV         R5,#0AH
                    MOV         R7,A
                    MOV         R6,AR2
                    LCALL       ?C?UIDIV
                    MOV         R0,#LOW (gp)
                    MOV         @R0,AR7
;  hp=fp%0x0a;
                                        ; SOURCE LINE # 585
                    MOV         R4,#00H
                    MOV         R5,#0AH
                    MOV         R7,AR3
                    MOV         R6,AR2
                    LCALL       ?C?UIDIV
                    MOV         R0,#LOW (hp)
                    MOV         @R0,AR4
                    INC         R0
                    MOV         @R0,AR5
; }
                                        ; SOURCE LINE # 586
                    RET
; END OF htdp

;
;
; void disr()

                    RSEG  ?PR?disr?KCT PF
disr:
                    USING       0
                                        ; SOURCE LINE # 589
; {
```

```
                                        ; SOURCE LINE # 590
; porta=0x80;
                                        ; SOURCE LINE # 591
; read();
                                        ; SOURCE LINE # 592
                        LCALL           L?0060
; delay();
                                        ; SOURCE LINE # 593
                        LCALL           delay
; porta=0xec;
                                        ; SOURCE LINE # 594
                        MOV             R0,#LOW (porta)
                        MOV             A,#0ECH
                        MOVX            @R0,A
; write();
                                        ; SOURCE LINE # 595
                        LCALL           write
; delay();
                                        ; SOURCE LINE # 596
                        LCALL           delay
; porta=':';
                                        ; SOURCE LINE # 597
                        MOV             R0,#LOW (porta)
                        MOV             A,#03AH
                        MOVX            @R0,A
; write();
                                        ; SOURCE LINE # 598
                        LCALL           write
; delay();
                                        ; SOURCE LINE # 599
                        LCALL           delay
; porta=ap+0x30;
                                        ; SOURCE LINE # 600
                        MOV             R0,#LOW (ap)
                        MOV             A,@R0
                        ADD             A,#030H
                        MOV             R1,#LOW (porta)
                        MOVX            @R1,A
; write();
                                        ; SOURCE LINE # 601
                        LCALL           write
; delay();
                                        ; SOURCE LINE # 602
                        LCALL           delay
; porta='.';
                                        ; SOURCE LINE # 603
                        MOV             R0,#LOW (porta)
                        MOV             A,#02EH
                        MOVX            @R0,A
; write();
                                        ; SOURCE LINE # 604
                        LCALL           write
; delay();
                                        ; SOURCE LINE # 605
                        LCALL           delay
; porta=cp+0x30;
                                        ; SOURCE LINE # 606
                        MOV             R0,#LOW (cp)
                        MOV             A,@R0
                        ADD             A,#030H
                        MOV             R1,#LOW (porta)
                        MOVX            @R1,A
; write();
                                        ; SOURCE LINE # 607
                        LCALL           write
; delay();
                                        ; SOURCE LINE # 608
                        LCALL           delay
; porta=ep+0x30;
                                        ; SOURCE LINE # 609
                        MOV             R0,#LOW (ep)
                        MOV             A,@R0
                        ADD             A,#030H
                        MOV             R1,#LOW (porta)
                        MOVX            @R1,A
; write();
```

```
                                              ; SOURCE LINE # 610
                              LCALL          write
; delay();
                                              ; SOURCE LINE # 611
                              LCALL          delay
; porta=gp+0x30;
                                              ; SOURCE LINE # 612
                              MOV            R0,#LOW (gp)
                              MOV            A,@R0
                              ADD            A,#030H
                              MOV            R1,#LOW (porta)
                              MOVX           @R1,A
; write();
                                              ; SOURCE LINE # 613
                              LCALL          write
; delay();
                                              ; SOURCE LINE # 614
                              LCALL          delay
; porta=hp+0x30;
                                              ; SOURCE LINE # 615
                              MOV            R0,#LOW (hp+01H)
                              MOV            A,@R0
                              ADD            A,#030H
                              MOV            R0,#LOW (porta)
                              MOVX           @R0,A
; write();
                                              ; SOURCE LINE # 616
                              LCALL          write
; delay();
                                              ; SOURCE LINE # 617
                              LCALL          delay
; porta=' ';
                                              ; SOURCE LINE # 618
; write();
                                              ; SOURCE LINE # 619
                              LCALL          L?0058
; delay();
                                              ; SOURCE LINE # 620
                              LJMP           delay
; END OF disr

                              END
```

Here we use 'C' language as a front end to display the various parameter readings that have been transmitted. By using 'C' the ports can be easily accessed by IN and OUT instructions and hence gets the use in this project. Once the initial test data is checked and the program starts reading the data (parameters) using interrupt command. Simultaneously the data are stored in a file and can be viewed in future when needed. So we use port accessing commands and file storage related commands in our 'C' programming. The coding is as follows:

```
**************************************************

   Generates Interrupt 14 to Check Serial Port

   If any new data in serial port the first bit

   of 'ah' register will go high.

**************************************************

#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <process.h>
void main()
{
  int ah, al;
  int f;
  int hour, min, sec;
  int year, month, day;
  struct time ti;
  struct date d;
  float temp=0.0;
  float pfr, vol, ct, powr, ene, ene1;
  //float energy=0.0;
  char x;
  unsigned char a, ch;
  FILE *stream,*str,*kct;
  unsigned char pf=0,volt=0,cur=0,pow=0,ener=0,tariff=0;
  double long pf1;
  unsigned char t1=0,t2=0,tt1=0,tt2=0,ttt1=0,ttt2=0,tttt1=0,tttt2=0;
  int i=0,j=0,no=0;
  int t;
  int flag=0;
  int count=0;
  clrscr();
  f=system("mode com1 110 n 8 1");
  gotoxy(28,3);
  printf("WIRELESS SURVEILLANCE OF ENERGY");
aa:
  gotoxy(25,5);
  printf("Power Factor %6.5f",(t2*256+t1)/10000.0);
  gotoxy(25,6);
  printf("Voltage      %d",volt);
  gotoxy(25,7);
  printf("Current      %2.1f",cur/10.0);
  gotoxy(25,8);
  printf("Power       %5.2f",(tt2*256+tt1)/100.0);
  gotoxy(25,9);
  printf("Energy      %5.2f",(ttt2*256+ttt1)/100.0);
  _AH=0x03;
  _DX=0x00;
  geninterrupt(0x14);
  ah=_AH;
  al=_AL;
  if( (ah & 0x01) == 0x01)
  {
  a=inportb(0x3f8);     //   reads data from serial port
  }
  if(a!=0xff)
  goto aa;
  i=0;
bb:
  _AH=0x03;
  _DX=0x00;
  geninterrupt(0x14);
  ah=_AH;
  al=_AL;
  if( (ah & 0x01) == 0x01)
  {
  i=i+1;
  a=inportb(0x3f8);
  if(i==1)
  {
  t1=a;
  }
  if(i==2)
  {
  t2=a;
  }
  if(i==3)
  {
  volt=a;
  }
  if(i==4)
  {
  cur=a;
  }
  if(i==5)
  {
  tt1=a;
  }
  if(i==6)
  {
  tt2=a;
  }
  if(i==7)
  {
  ttt1=a;
  }
  if(i==8)
  {
  ttt2=a;
  }
  }
  if(i<8)
  goto bb;
  str = fopen("sri.dat", "r+");
  ch=fgetc(str);
  if(ch!='\n')
  {
  fscanf(str,"%f",&pfr);
  fscanf(str,"%f",&vol);
  fscanf(str,"%f",&ct);
  fscanf(str,"%f",&powr);
  fscanf(str,"%f",&ene);
  }
  fclose(str);
  str = fopen("sri.dat", "w+");
  ene=ene-temp+(ttt2*256+ttt1)/100.0;
  temp=(ttt2*256+ttt1)/100.0;
  fprintf(str,"%6.5f %d %2.1f %5.2f
%5.2f",(t2*256+t1)/10000.0,volt,cur/10.0,(tt2*256+tt1)/100.0,ene);
  fclose(str);
  str = fopen("sri.dat", "r+");
  //ch=fgetc(str);
  //if(ch!='\n')
  //{
  fscanf(str,"%f",&pfr);
  fscanf(str,"%f",&vol);
  fscanf(str,"%f",&ct);
  fscanf(str,"%f",&powr);
  fscanf(str,"%f",&ene);
  //}
  fclose(str);
  stream = fopen("result.dat", "w+");
  fprintf(stream,"%f\n",pfr);
  fprintf(stream,"%f\n",vol);
  fprintf(stream,"%f\n",ct);
  fprintf(stream,"%f\n",powr);
  fprintf(stream,"%f\n",ene);
  fclose(stream);
  if(volt==0 && i==0)
  {
  kct = fopen("log.dat", "a+");
  getdate(&d);
  year=d.da_year;
  day=d.da_day;
  month=d.da_mon;
  gettime(&ti);
  hour=ti.ti_hour;
  min=ti.ti_min;
  sec=ti.ti_sec;
  fprintf(kct,"POWER SHUT DOWN TIME %d : %d : % d  DATE   %d : %d : %d
\n",hour,min,sec,day,month,year);
  fclose(kct);
  }
  goto aa;
}
```

9. CONCLUSION

This project is microcontroller based Energy Monitor and transmission of energy through wireless to an EB substation. This project can be implemented for a complete network in an area, that will be consuming very huge amount and it can calculate the energy transmission and energy consumption. By using the above two values we can calculate the transmission loses as well.

This project design is cheap, efficient and showed an excellent performance. This project is very flexible and it can also be implemented using telephone or power lines depending on various criteria.

The implementation of the project will be an excellent way of reducing the man power involved in the field work today. Also this extends 24 – Hour surveillance which is a standout quality. Hence such qualities ensure that this is a practically feasible project.

10. BIBLIOGRAPHY

ROY CHOUDHARY – "LINEAR INTEGRATED CIRCUITS"
PUBLISHED BY NEW AGE INTERNATIONAL

DALLAS SEMICONDUCTORS – "SYSTEM EXTENSION DATABOOK" 1995

YASHWANTH KANITKAR – "LET US C"
PUBLISHED BY TATA MCGRAW HILL 2ND EDITION

WWW.ATMEL.COM

POWER IC'S DATABOOK
PUBLISHED BY NATIONAL SEMICONDUCTORS

ANOLOG MODULATION TECHNIQUES
PUBLISHED BY RAJ PAPERS

FAIRCHILD
SEMICONDUCTOR ™

March 1998

DM74LS373/DM74LS374 3-STATE Octal D-Type Transparent Latches and Edge-Triggered Flip-Flops

General Description

These 8-bit registers feature totem-pole 3-STATE outputs designed specifically for driving highly-capacitive or relatively low-impedance loads. The high-impedance state and increased high-logic level drive provide these registers with the capability of being connected directly to and driving the bus lines in a bus-organized system without need for interface or pull-up components. They are particularly attractive for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

The eight latches of the DM54/74LS373 are transparent D-type latches meaning that while the enable (G) is high the Q outputs will follow the data (D) inputs. When the enable is taken low the output will be latched at the level of the data that was set up.

The eight flip-flops of the DM54/74LS374 are edge-triggered D-type flip flops. On the positive transition of the clock, the Q outputs will be set to the logic states that were set up at the D inputs.

A buffered output control input can be used to place the eight outputs in either a normal logic state (high or low logic levels) or a high-impedance state. In the high-impedance state the outputs neither load nor drive the bus lines significantly.

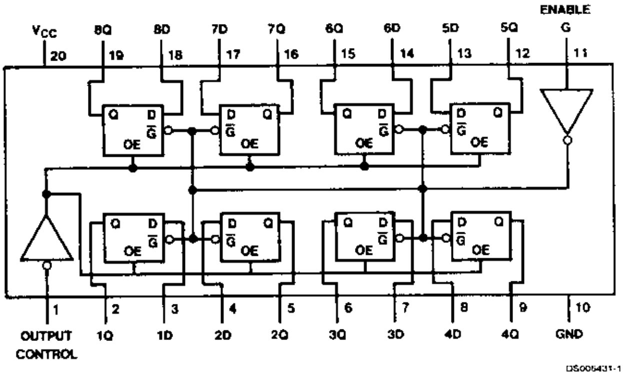
The output control does not affect the internal operation of the latches or flip-flops. That is, the old data can be retained or new data can be entered even while the outputs are off.

Features

- Choice of 8 latches or 8 D-type flip-flops in a single package
- 3-STATE bus-driving outputs
- Full parallel-access for loading
- Buffered control inputs
- P-N-P inputs reduce D-C loading on data lines

Connection Diagrams

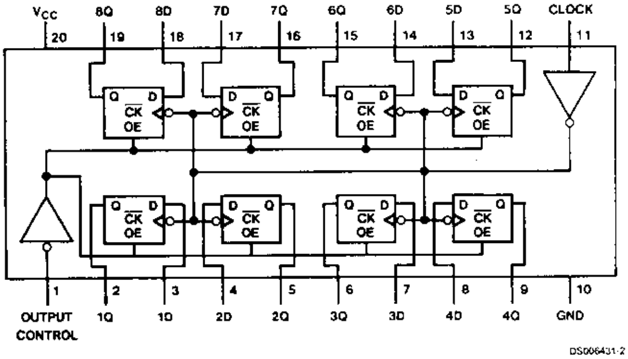
Dual-In-Line Packages
'LS373



Order Number DM54LS373J, DM54LS373W, DM74LS373N or DM74LS373WM
See Package Number J20A, M20B, N20A or W20A

Connection Diagrams (Continued)

'LS374



DS006431-2

Order Number DM54LS374J, DM54LS374W, DM74LS374WM or DM74LS374N
See Package Number J20A, M20B, N20A or W20A

Function Tables

DM54/74LS373

Output Control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

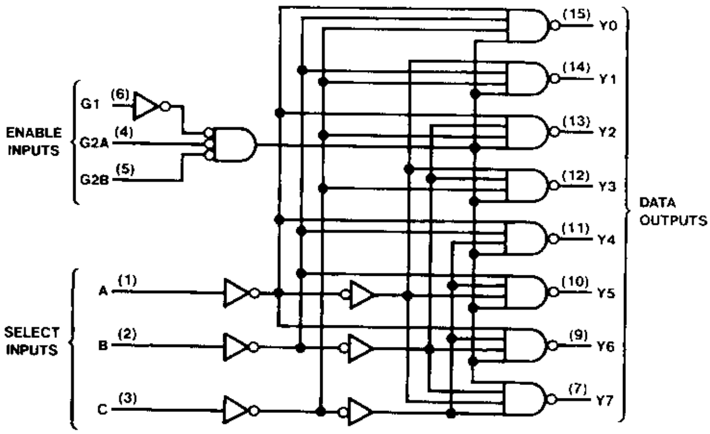
H = High Level (Steady State), L = Low Level (Steady State), X = Don't Care
↑ = Transition from low-to-high level, Z = High Impedance State
Q_0 = The level of the output before steady-state input conditions were established.

DM54/74LS374

Output Control	Clock	D	Output
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

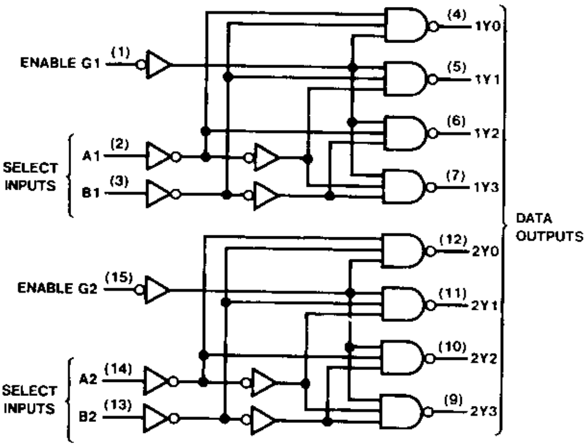
Logic Diagrams

LS138



DS006391-3

LS139



DS006391-4

DM7414 Hex Inverter with Schmitt Trigger Inputs

FAIRCHILD
SEMICONDUCTOR™

March 1998

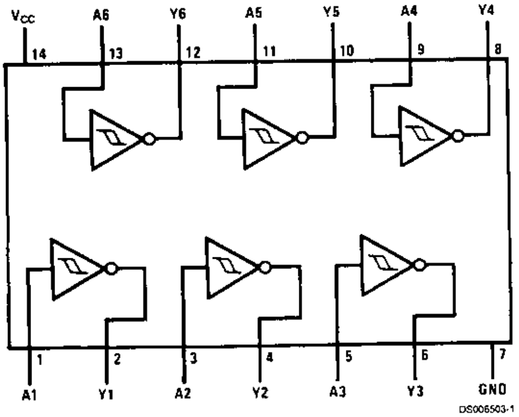
DM7414
Hex Inverter with Schmitt Trigger Inputs

General Description

This device contains six independent gates each of which performs the logic INVERT function. Each input has hysteresis which increases the noise immunity and transforms a slowly changing input signal to a fast changing, jitter free output.

Connection Diagram

Dual-In-Line Package



Order Number DM5414J, DM5414W or DM7414N
See Package Number J14A, N14A or W14B

Function Table

Y = $\overline{A}$

Input	Output
A	Y
L	H
H	L

H = High Logic Level
L = Low Logic Level

FAIRCHILD
SEMICONDUCTOR™

March 1998

DM74LS138, DM74LS139 Decoders/Demultiplexers

General Description

These Schottky-clamped circuits are designed to be used in high-performance memory-decoding or data-routing applications, requiring very short propagation delay times. In high-performance memory systems these decoders can be used to minimize the effects of system decoding. When used with high-speed memories, the delay times of these decoders are usually less than the typical access time of the memory. This means that the effective system delay introduced by the decoder is negligible.

The LS138 decodes one-of-eight lines, based upon the conditions at the three binary select inputs and the three enable inputs. Two active-low and one active-high enable inputs reduce the need for external gates or inverters when expanding. A 24-line decoder can be implemented with no external inverters, and a 32-line decoder requires only one inverter. An enable input can be used as a data input for demultiplexing applications.

The LS139 comprises two separate two-line-to-four-line decoders in a single package. The active-low enable input can be used as a data line in demultiplexing applications.

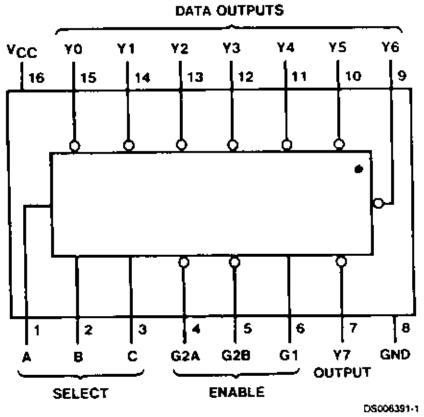
All of these decoders/demultiplexers feature fully buffered inputs, presenting only one normalized load to its driving circuit. All inputs are clamped with high-performance Schottky diodes to suppress line-ringing and simplify system design.

Features

- Designed specifically for high speed:
  Memory decoders
  Data transmission systems
- LS138 3-to-8-line decoders incorporates 3 enable inputs to simplify cascading and/or data reception
- LS139 contains two fully independent 2-to-4-line decoders/demultiplexers
- Schottky clamped for high performance
- Typical propagation delay (3 levels of logic)
  LS138 21 ns
  LS139 21 ns
- Typical power dissipation
  LS138 32 mW
  LS139 34 mW
- Alternate Military/Aerospace devices (54LS138, 54LS139) are available. Contact a Fairchild Semiconductor Sales Office/Distributor for specifications.

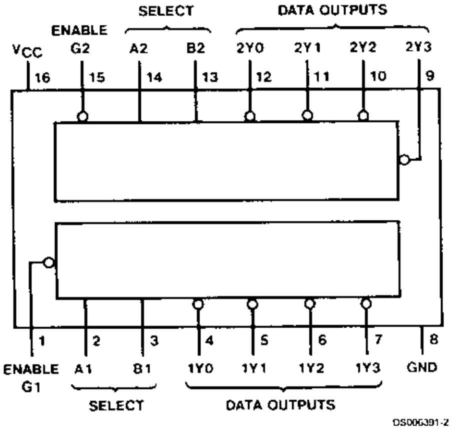
Connection Diagrams

Dual-In-Line Package



Order Number 54LS138DMQB, 54LS138FMQB, 54LS138LMQB, DM54LS138J, DM54LS138W, DM74LS138M or DM74LS138N
See Package Number E20A, J16A, M16A, N16E or W16A
Dual-In-Line Package

Dual-In-Line Package



Order Number 54LS139DMQB, 54LS139FMQB, 54LS139LMQB, DM54LS139J, DM54LS139W, DM74LS139M or DM74LS139N
See Package Number E20A, J16A, M16A, N16E or W16A

'LS139 Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	From (Input) To (Output)	R_L = 2 kΩ				Units
			C_L = 15 pF		C_L = 50 pF		
			Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time Low to High Level Output	Select to Output		18		27	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Select to Output		27		40	ns
t_{PLH}	Propagation Delay Time Low to High Level Output	Enable to Output		18		27	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Enable to Output		24		40	ns

Function Tables

LS138

Inputs					Outputs							
Enable		Select										
G1	G2 (Note 8)	C	B	A	YO	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

H = High Level, L = Low Level, X = Don't Care

Note 8: G2 = G2A + G2B

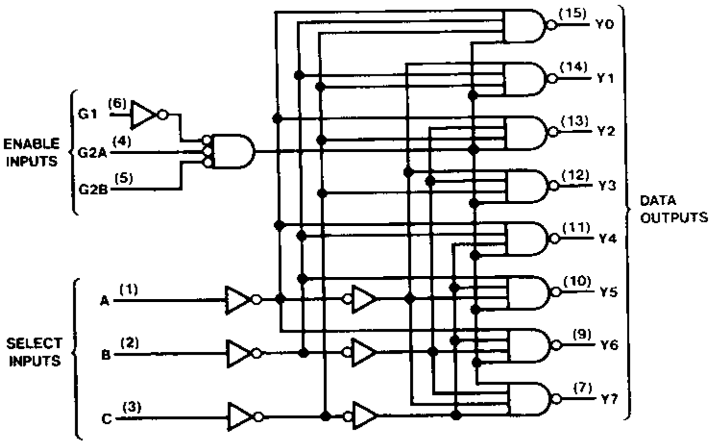
LS139

Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = High Level, L = Low Level, X = Don't Care

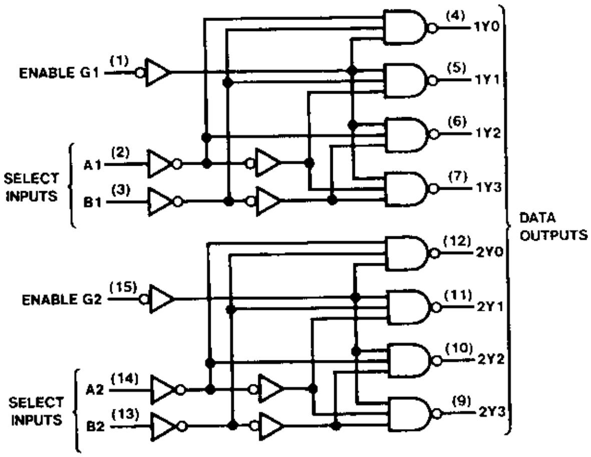
Logic Diagrams

LS138



DS006391-3

LS139



DS006391-4

TEXAS
STRUMENTS

heet acquired from Harris Semiconductor
021

CD4011B, CD4012B, CD4023B Types

OS NAND GATES

Voltage Types (20-Volt Rating)

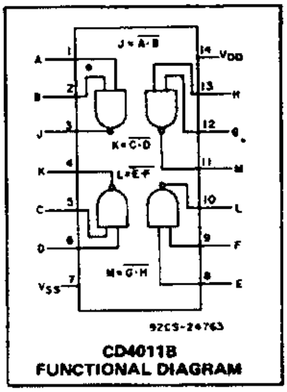
2 Input – CD4011B
4 Input – CD4012B
e 3 Input – CD4023B

CD4011B, CD4012B, and CD4023B
D gates provide the system designer
direct implementation of the NAND
ion and supplement the existing family
MOS gates. All inputs and outputs are
red.

CD4011B, CD4012B, and CD4023B
are supplied in 14-lead hermetic dual-
e ceramic packages (D and F suffixes),
ad dual-in-line plastic packages (E suf-
nd in chip form (H suffix).

Features:

- Propagation delay time = 60 ns (typ.) at C_L = 50 pF, V_{DD} = 10 V
- Buffered inputs and outputs
- Standardized symmetrical output characteristics
- Maximum input current of 1 μA at 18 V over full package temperature range; 100 nA at 18 V and 25°C
- 100% tested for quiescent current at 20 V
- 5-V, 10-V, and 15-V parametric ratings
- Noise margin (over full package temperature range:
  1 V at V_{DD} = 5 V
  2 V at V_{DD} = 10 V
  2.5 V at V_{DD} = 15 V
- Meets all requirements of JEDEC Tentative Standard No. 13B, "Standard Specifications for Description of "B" Series CMOS Devices"


J = $\overline{A \cdot B}$
K = $\overline{C \cdot D}$
L = $\overline{E \cdot F}$
M = $\overline{G \cdot H}$
A
B
J
K
C
D
V_{SS}
V_{DD}
H
G
M
L
F
E
92CS-24763

CD4011B
FUNCTIONAL DIAGRAM

UM RATINGS, *Absolute-Maximum Values:*

PPLY-VOLTAGE RANGE, (V_{DD})

ages referenced to V_{SS} Terminal) .... -0.5V to +20V

VOLTAGE RANGE, ALL INPUTS .... -0.5V to V_{DD} +0.5V

PUT CURRENT, ANY ONE INPUT .... ±10mA

R DISSIPATION PER PACKAGE (P_D):

T_A = -55°C to +100°C 500mW

T_A = +100°C to +125°C .... Derate Linearity at 12mW/°C to 200mW

E DISSIPATION PER OUTPUT TRANSISTOR

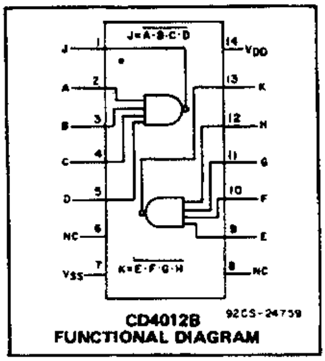
T_A = FULL PACKAGE-TEMPERATURE RANGE (All Package Types) 100mW

ATING-TEMPERATURE RANGE (T_A) .... -55°C to +125°C

AGE TEMPERATURE RANGE (T_{stg}) .... -65°C to +150°C

TEMPERATURE (DURING SOLDERING):

istance 1/16 ± 1/32 inch (1.59 ± 0.79mm) from case for 10s max .... +265°C

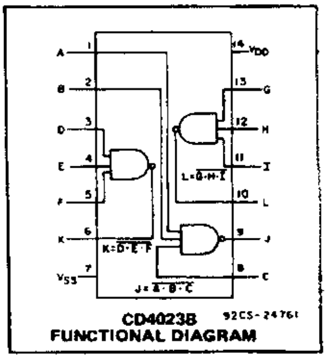

J = $\overline{A \cdot B \cdot C \cdot D}$
K = $\overline{E \cdot F \cdot G \cdot H}$
J
A
B
C
D
NC
V_{SS}
V_{DD}
K
H
G
F
E
NC
92CS-24759

CD4012B
FUNCTIONAL DIAGRAM

OMMENDED OPERATING CONDITIONS

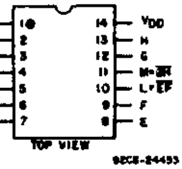
maximum reliability, nominal operating conditions should be selected so that ation is always within the following ranges:

CHARACTERISTIC	LIMITS		UNITS
	MIN.	MAX.	
pply-Voltage Range (For T_A = Full Package Temperature Range)	3	18	V

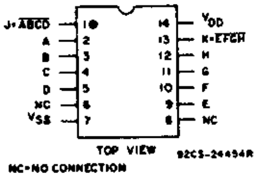

A
B
D
E
F
K
V_{SS}
V_{DD}
G
H
I
L
J
C
L = $\overline{G \cdot H \cdot I}$
K = $\overline{D \cdot E \cdot F}$
J = $\overline{A \cdot B \cdot C}$
92CS-24761

CD4023B
FUNCTIONAL DIAGRAM

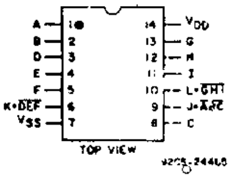
TERMINAL ASSIGNMENTS



CD4011B



CD4012B



CD4023B

FAIRCHILD
SEMICONDUCTOR™

March 1998

DM7400
Quad 2-Input NAND Gates

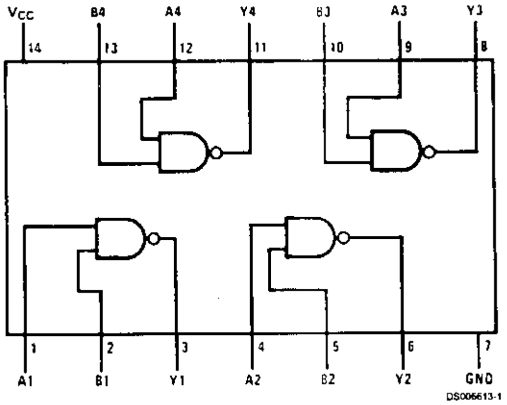
General Description

This device contains four independent gates each of which performs the logic NAND function.

Features

- Alternate Military/Aerospace device (5400) is available. Contact a Fairchild Semiconductor Sales Office/Distributor for specifications.

Connection Diagram



Order Number 5400DMQB, 5400FMQB, DM5400J, DM5400W or DM7400N
See Package Number J14A, N14A or W14B

Function Table

$Y = \overline{AB}$

Inputs		Output
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

H = High Logic Level
L = Low Logic Level

National Semiconductor

May 2000

LM78XX Series Voltage Regulators

General Description

The LM78XX series of three terminal regulators is available with several fixed output voltages making them useful in a wide range of applications. One of these is local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow these regulators to be used in logic systems, instrumentation, HiFi, and other solid state electronic equipment. Although designed primarily as fixed voltage regulators these devices can be used with external components to obtain adjustable voltages and currents.

The LM78XX series is available in an aluminum TO-3 package which will allow over 1.0A load current if adequate heat sinking is provided. Current limiting is included to limit the peak output current to a safe value. Safe area protection for the output transistor is provided to limit internal power dissipation. If internal power dissipation becomes too high for the heat sinking provided, the thermal shutdown circuit takes over preventing the IC from overheating.

Considerable effort was expanded to make the LM78XX series of regulators easy to use and minimize the number of external components. It is not necessary to bypass the output, although this does improve transient response. Input bypassing is needed only if the regulator is located far from the filter capacitor of the power supply.

For output voltage other than 5V, 12V and 15V the LM117 series provides an output voltage range from 1.2V to 57V.

Features

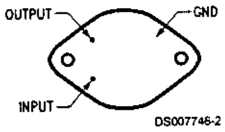
- Output current in excess of 1A
- Internal thermal overload protection
- No external components required
- Output transistor safe area protection
- Internal short circuit current limit
- Available in the aluminum TO-3 package

Voltage Range

LM7805C	5V
LM7812C	12V
LM7815C	15V

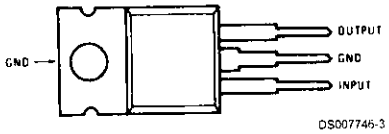
Connection Diagrams

Metal Can Package
TO-3 (K)
Aluminum



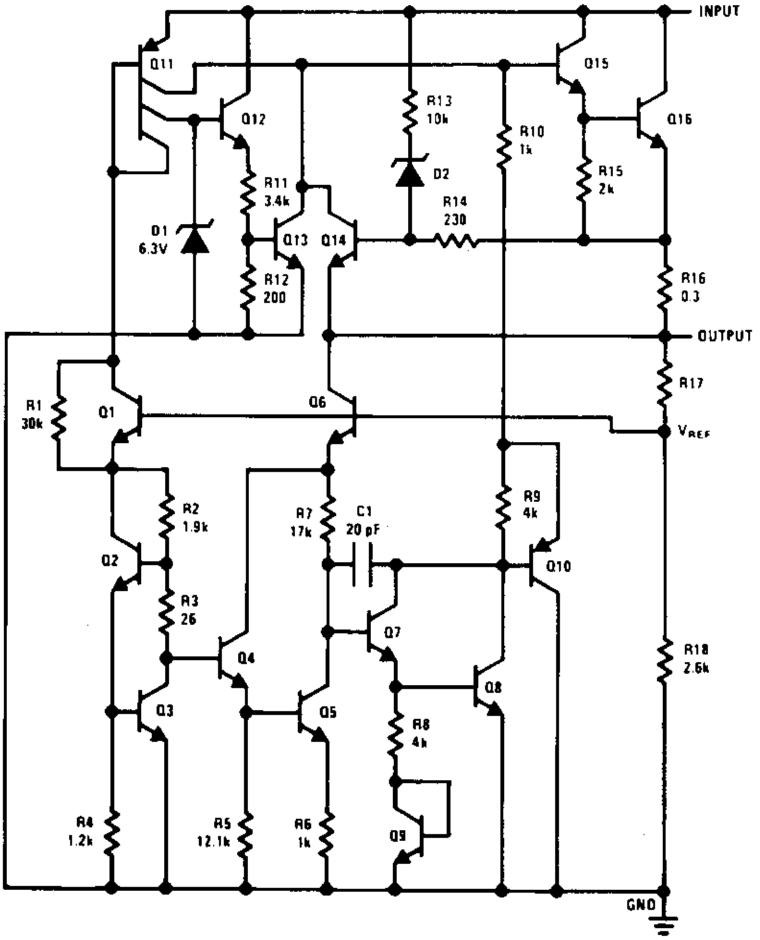
Bottom View
Order Number LM7805CK,
LM7812CK or LM7815CK
See NS Package Number KC02A

Plastic Package
TO-220 (T)



Top View
Order Number LM7805CT,
LM7812CT or LM7815CT
See NS Package Number T03B

Schematic



DS007746-1

Order this document by MC7900/D

MOTOROLA

MC7900 Series

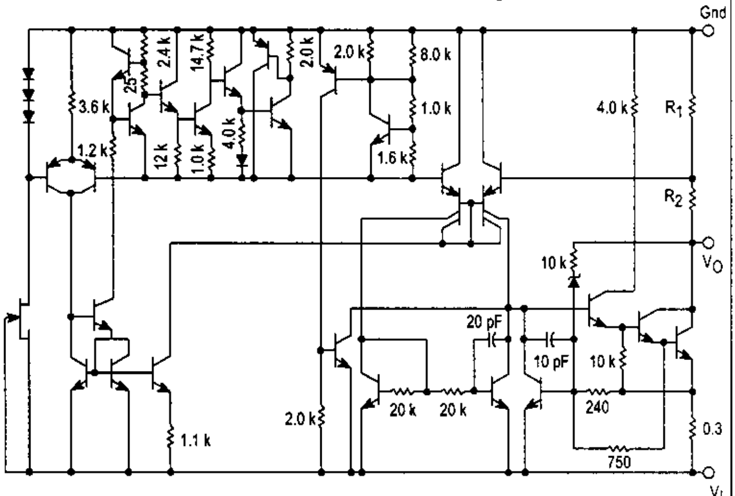
Three-Terminal Negative Voltage Regulators

The MC7900 series of fixed output negative voltage regulators are intended as complements to the popular MC7800 series devices. These negative regulators are available in the same seven–voltage options as the MC7800 devices. In addition, one extra voltage option commonly employed in MECL systems is also available in the negative MC7900 series.

Available in fixed output voltage options from −5.0 V to −24 V, these regulators employ current limiting, thermal shutdown, and safe–area compensation – making them remarkably rugged under most operating conditions. With adequate heatsinking they can deliver output currents in excess of 1.0 A.

- No External Components Required
- Internal Thermal Overload Protection
- Internal Short Circuit Current Limiting
- Output Transistor Safe–Area Compensation
- Available in 2% Voltage Tolerance (See Ordering Information)

Representative Schematic Diagram



This device contains 26 active transistors.

ORDERING INFORMATION

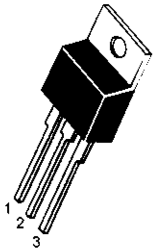
Device	Output Voltage Tolerance	Operating Temperature Range	Package
MC79XXACD2T	2%	T_J = 0° to +125°C	Surface Mount
MC79XXCD2T	4%		
MC79XXACT	2%		Insertion Mount
MC79XXCT	4%		
MC79XXBD2T	4%	T_J = −40° to +125°C	Surface Mount
MC79XXBT			Insertion Mount

XX indicates nominal voltage.

THREE–TERMINAL NEGATIVE FIXED VOLTAGE REGULATORS

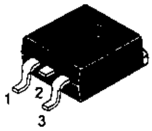
T SUFFIX
PLASTIC PACKAGE
CASE 221A

Heatsink surface connected to Pin 2.



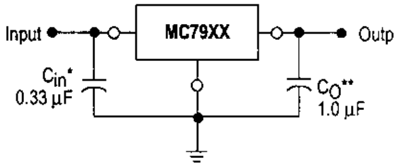
Pin 1. Ground
2. Input
3. Output

D2T SUFFIX
PLASTIC PACKAGE
CASE 936
(D^2PAK)



Heatsink surface (shown as terminal 4 in case outline drawing) is connected to Pin 2.

STANDARD APPLICATION



A common ground is required between the input and the output voltages. The input voltage must remain typically 2.0 V above more negative even during the high point of the input ripple voltage.

XX, These two digits of the type number indicate nominal voltage.

* C_{in} is required if regulator is located an appreciable distance from power supply filter.

** C_O improve stability and transient response.

DEVICE TYPE/NOMINAL OUTPUT VOLTAGE

MC7905	5.0 V	MC7912	12 V
MC7905.2	5.2 V	MC7915	15 V
MC7906	6.0 V	MC7918	28 V
MC7908	8.0 V	MC7924	24 V

National Semiconductor

October 1999

ADC0808/ADC0809
8-Bit µP Compatible A/D Converters with 8-Channel Multiplexer

General Description

The ADC0808, ADC0809 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 8-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 8-channel multiplexer can directly access any of 8-single-ended analog signals.

The device eliminates the need for external zero and full-scale adjustments. Easy interfacing to microprocessors is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATE® outputs.

The design of the ADC0808, ADC0809 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC0808, ADC0809 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications from process and machine control to consumer and automotive applications. For 16-channel multiplexer with common output (sample/hold port) see ADC0816 data sheet. (See AN-247 for more information.)

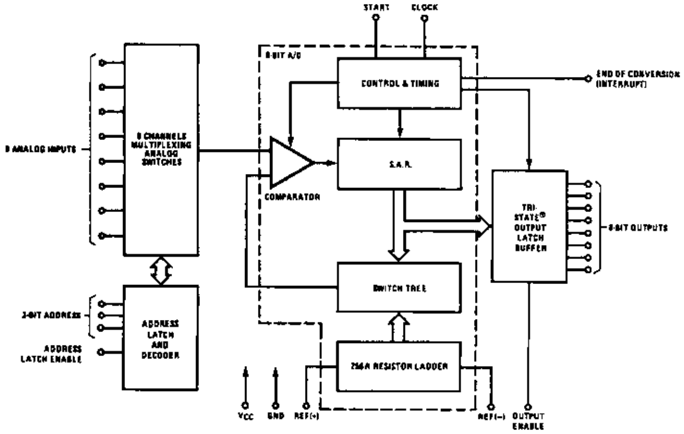
Features

- Easy interface to all microprocessors
- Operates ratiometrically or with 5 V_{DC} or analog span adjusted voltage reference
- No zero or full-scale adjust required
- 8-channel multiplexer with address logic
- 0V to 5V input range with single 5V power supply
- Outputs meet TTL voltage level specifications
- Standard hermetic or molded 28-pin DIP package
- 28-pin molded chip carrier package
- ADC0808 equivalent to MM74C949
- ADC0809 equivalent to MM74C949-1

Key Specifications

■ Resolution	8 Bits
■ Total Unadjusted Error	±½ LSB and ±1 LSB
■ Single Supply	5 V_{DC}
■ Low Power	15 mW
■ Conversion Time	100 µs

Block Diagram



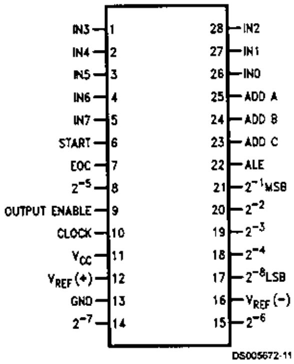
DS005672-1

See Ordering Information

TRI-STATE® is a registered trademark of National Semiconductor Corp.

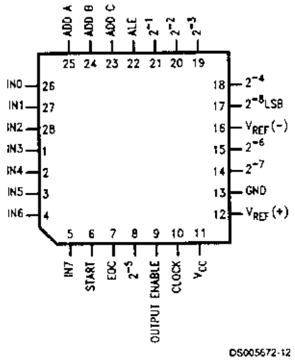
Connection Diagrams

Dual-In-Line Package



Order Number ADC0808CCN or ADC0809CCN
See NS Package J28A or N28A

Molded Chip Carrier Package



Order Number ADC0808CCV or ADC0809CCV
See NS Package V28A

Ordering Information

TEMPERATURE RANGE		−40°C to +85°C			−55°C to +125°C
Error	±½ LSB Unadjusted	ADC0808CCN	ADC0808CCV	ADC0808CCJ	ADC0808CJ
	±1 LSB Unadjusted	ADC0809CCN	ADC0809CCV		
Package Outline		N28A Molded DIP	V28A Molded Chip Carrier	J28A Ceramic DIP	J28A Ceramic DIP

Functional Description

Multiplexer. The device contains an 8-channel single-ended analog signal multiplexer. A particular input channel is selected by using the address decoder. *Table 1* shows the input states for the address lines to select any channel. The address is latched into the decoder on the low-to-high transition of the address latch enable signal.

TABLE 1.

SELECTED ANALOG CHANNEL	ADDRESS LINE		
	C	B	A
IN0	L	L	L
IN1	L	L	H
IN2	L	H	L
IN3	L	H	H
IN4	H	L	L
IN5	H	L	H
IN6	H	H	L
IN7	H	H	H

CONVERTER CHARACTERISTICS

The Converter

The heart of this single chip data acquisition system is its 8-bit analog-to-digital converter. The converter is designed to give fast, accurate, and repeatable conversions over a wide range of temperatures. The converter is partitioned into 3 major sections: the 256R ladder network, the successive approximation register, and the comparator. The converter's digital outputs are positive true.

The 256R ladder network approach (*Figure 1*) was chosen over the conventional R/2R ladder because of its inherent monotonicity, which guarantees no missing digital codes. Monotonicity is particularly important in closed loop feedback control systems. A non-monotonic relationship can cause oscillations that will be catastrophic for the system. Additionally, the 256R network does not cause load variations on the reference voltage.

The bottom resistor and the top resistor of the ladder network in *Figure 1* are not the same value as the remainder of the network. The difference in these resistors causes the output characteristic to be symmetrical with the zero and full-scale points of the transfer curve. The first output transition occurs when the analog signal has reached +½ LSB and succeeding output transitions occur every 1 LSB later up to full-scale.

The successive approximation register (SAR) performs 8 iterations to approximate the input voltage. For any SAR type converter, n-iterations are required for an n-bit converter. *Figure 2* shows a typical example of a 3-bit converter. In the ADC0808, ADC0809, the approximation technique is extended to 8 bits using the 256R network.

The A/D converter's successive approximation register (SAR) is reset on the positive edge of the start conversion (SC) pulse. The conversion is begun on the falling edge of the start conversion pulse. A conversion in process will be interrupted by receipt of a new start conversion pulse. Continuous conversion may be accomplished by tying the end-of-conversion (EOC) output to the SC input. If used in this mode, an external start conversion pulse should be applied after power up. End-of-conversion will go low between 0 and 8 clock pulses after the rising edge of start conversion.

The most important section of the A/D converter is the comparator. It is this section which is responsible for the ultimate accuracy of the entire converter. It is also the comparator drift which has the greatest influence on the repeatability of the device. A chopper-stabilized comparator provides the most effective method of satisfying all the converter requirements.

The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is then fed through a high gain AC amplifier and has the DC level restored. This technique limits the drift component of the amplifier since the drift is a DC component which is not passed by the AC amplifier. This makes the entire A/D converter extremely insensitive to temperature, long term drift and input offset errors.

Figure 4 shows a typical error curve for the ADC0808 as measured using the procedures outlined in AN-179.

Functional Description (Continued)

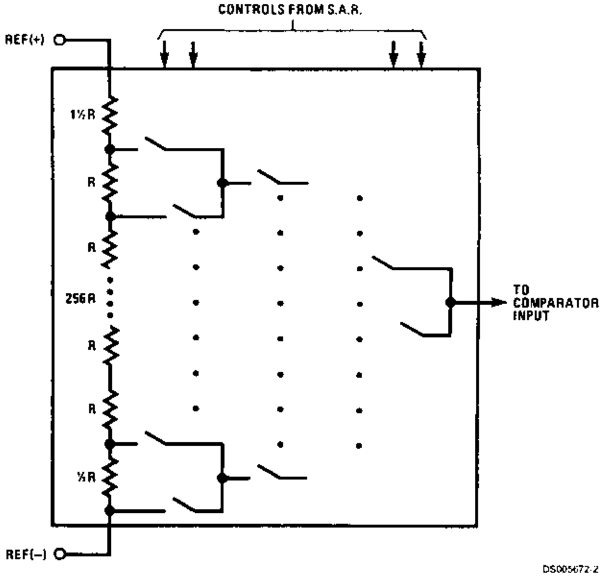


FIGURE 1. Resistor Ladder and Switch Tree

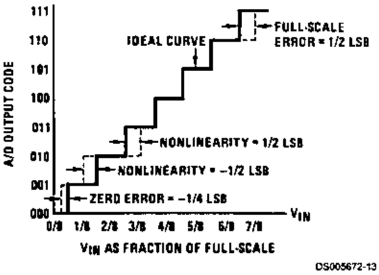


FIGURE 2. 3-Bit A/D Transfer Curve

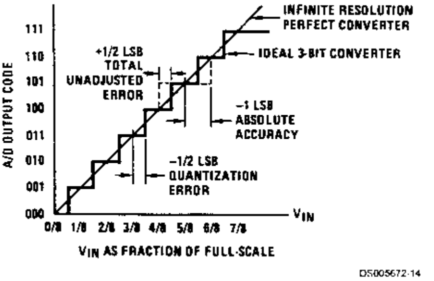


FIGURE 3. 3-Bit A/D Absolute Accuracy Curve

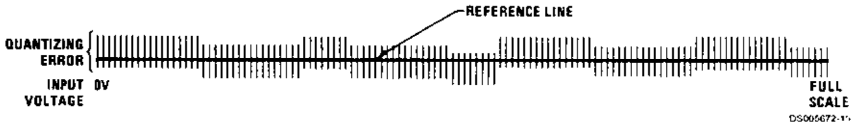
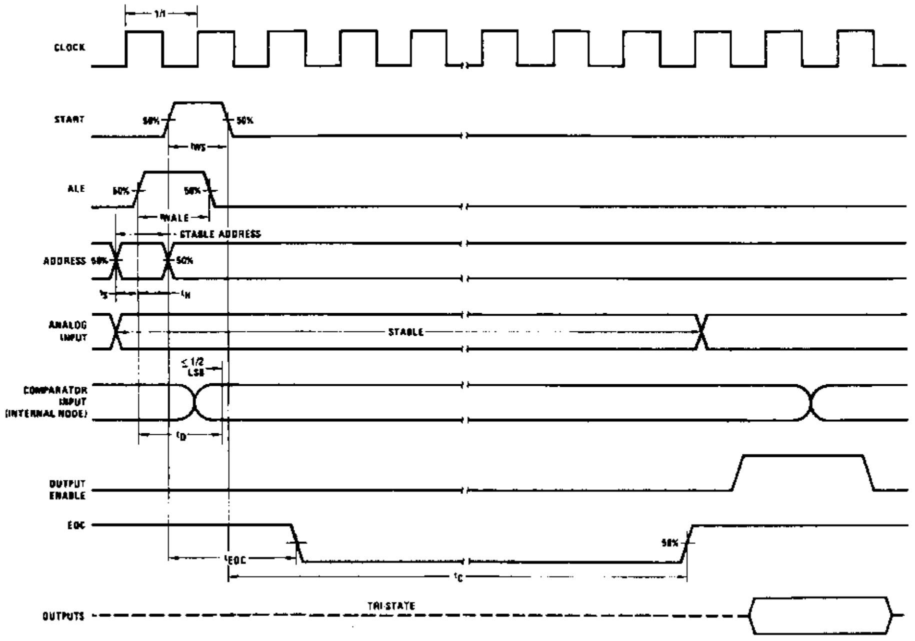


FIGURE 4. Typical Error Curve

Timing Diagram



DS005672-4

FIGURE 5.

National Semiconductor

May 1999

LM565/LM565C Phase Locked Loop

General Description

The LM565 and LM565C are general purpose phase locked loops containing a stable, highly linear voltage controlled oscillator for low distortion FM demodulation, and a double balanced phase detector with good carrier suppression. The VCO frequency is set with an external resistor and capacitor, and a tuning range of 10:1 can be obtained with the same capacitor. The characteristics of the closed loop system — bandwidth, response speed, capture and pull in range — may be adjusted over a wide range with an external resistor and capacitor. The loop may be broken between the VCO and the phase detector for insertion of a digital frequency divider to obtain frequency multiplication.

The LM565H is specified for operation over the −55˚C to +125˚C military temperature range. The LM565CN is specified for operation over the 0˚C to +70˚C temperature range.

Features

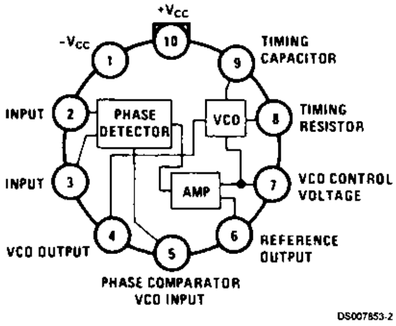
- 200 ppm/˚C frequency stability of the VCO
- Power supply range of ±5 to ±12 volts with 100 ppm/% typical
- 0.2% linearity of demodulated output
- Linear triangle wave with in phase zero crossings available
- TTL and DTL compatible phase detector input and square wave output
- Adjustable hold in range from ±1% to > ±60%

Applications

- Data and tape synchronization
- Modems
- FSK demodulation
- FM demodulation
- Frequency synthesizer
- Tone decoding
- Frequency multiplication and division
- SCA demodulators
- Telemetry receivers
- Signal regeneration
- Coherent demodulators

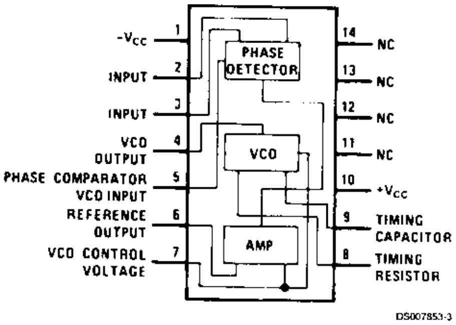
Connection Diagrams

Metal Can Package



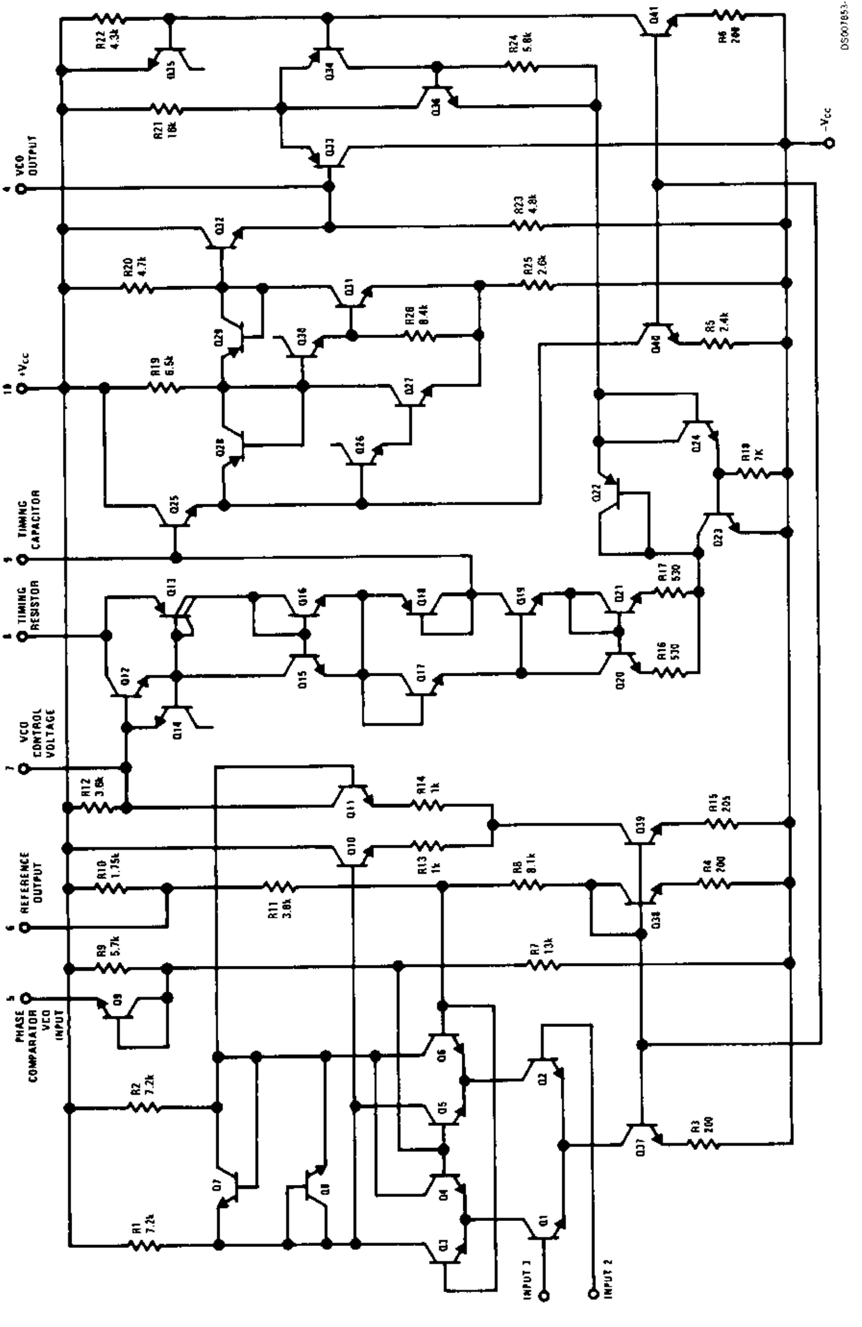
Order Number LM565H
See NS Package Number H10C

Dual-in-Line Package



Order Number LM565CN
See NS Package Number N14A

Schematic Diagram



National Semiconductor

November 1994

LM741 Operational Amplifier

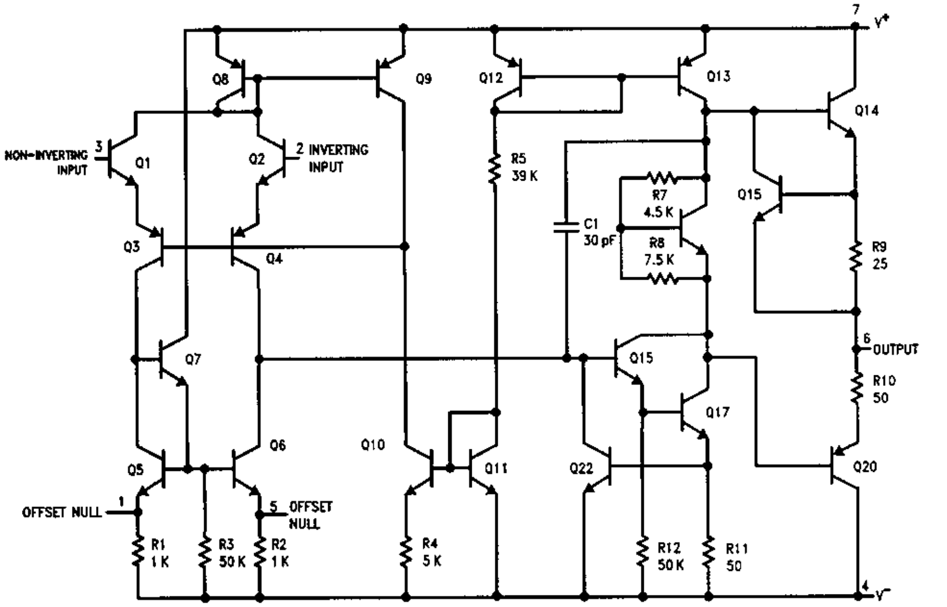
General Description

The LM741 series are general purpose operational amplifiers which feature improved performance over industry standards like the LM709. They are direct, plug-in replacements for the 709C, LM201, MC1439 and 748 in most applications.

The amplifiers offer many features which make their application nearly foolproof: overload protection on the input and output, no latch-up when the common mode range is exceeded, as well as freedom from oscillations.

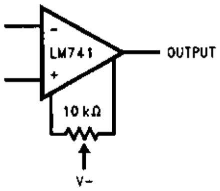
The LM741C/LM741E are identical to the LM741/LM741A except that the LM741C/LM741E have their performance guaranteed over a 0°C to +70°C temperature range, instead of −55°C to +125°C.

□□□□□ □□□ □ □□□□□



TL/H/9341–1

□□□□□□□ □□□□ □□□□□□



TL/H/9341–7

RRD-B30M115/Printed in U. S. A.

XR-2206

Monolithic Function Generator

June 1997-3

FEATURES

- Low-Sine Wave Distortion, 0.5%, Typical
- Excellent Temperature Stability, 20ppm/°C, Typ.
- Wide Sweep Range, 2000:1, Typical
- Low-Supply Sensitivity, 0.01%V, Typ.
- Linear Amplitude Modulation
- TTL Compatible FSK Controls
- Wide Supply Range, 10V to 26V
- Adjustable Duty Cycle, 1% TO 99%

APPLICATIONS

- Waveform Generation
- Sweep Generation
- AM/FM Generation
- V/F Conversion
- FSK Generation
- Phase-Locked Loops (VCO)

GENERAL DESCRIPTION

The XR-2206 is a monolithic function generator integrated circuit capable of producing high quality sine, square, triangle, ramp, and pulse waveforms of high-stability and accuracy. The output waveforms can be both amplitude and frequency modulated by an external voltage. Frequency of operation can be selected externally over a range of 0.01Hz to more than 1MHz.

The circuit is ideally suited for communications, instrumentation, and function generator applications requiring sinusoidal tone, AM, FM, or FSK generation. It has a typical drift specification of 20ppm/°C. The oscillator frequency can be linearly swept over a 2000:1 frequency range with an external control voltage, while maintaining low distortion.

ORDERING INFORMATION

Part No.	Package	Operating Temperature Range
XR-2206M	16 Lead 300 Mil CDIP	-55°C to +125°C
XR-2206P	16 Lead 300 Mil PDIP	–40°C to +85°C
XR-2206CP	16 Lead 300 Mil PDIP	0°C to +70°C
XR-2206D	16 Lead 300 Mil JEDEC SOIC	0°C to +70°C

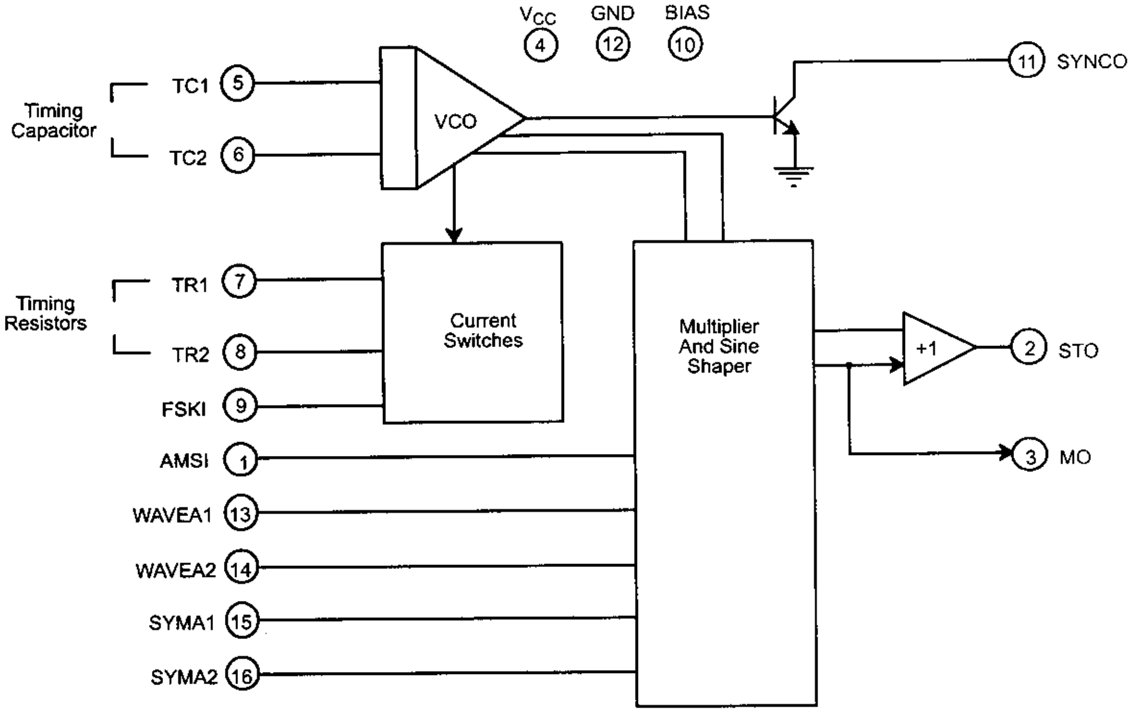
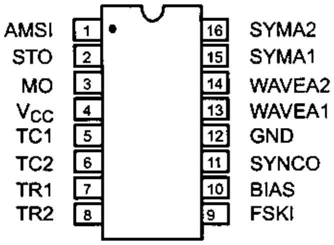
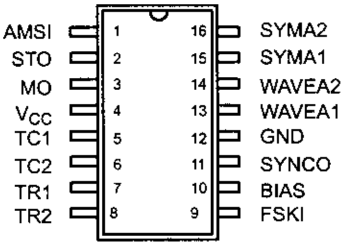


Figure 1. XR-2206 Block Diagram



16 Lead PDIP, CDIP (0.300")



16 Lead SOIC (Jedec, 0.300")

PIN DESCRIPTION

Pin #	Symbol	Type	Description
1	AMSI	I	**Amplitude Modulating Signal Input.**
2	STO	O	**Sine or Triangle Wave Output.**
3	MO	O	**Multiplier Output.**
4	V_{CC}		**Positive Power Supply.**
5	TC1	I	**Timing Capacitor Input.**
6	TC2	I	**Timing Capacitor Input.**
7	TR1	O	**Timing Resistor 1 Output.**
8	TR2	O	**Timing Resistor 2 Output.**
9	FSKI	I	**Frequency Shift Keying Input.**
10	BIAS	O	**Internal Voltage Reference.**
11	SYNCO	O	**Sync Output.** This output is a open collector and needs a pull up resistor to V_{CC}.
12	GND		**Ground pin.**
13	WAVEA1	I	**Wave Form Adjust Input 1.**
14	WAVEA2	I	**Wave Form Adjust Input 2.**
15	SYMA1	I	**Wave Symetry Adjust 1.**
16	SYMA2	I	**Wave Symetry Adjust 2.**