

# *DIGITAL ENERGY METER CUM COST INDICATOR*

P-712

PROJECT REPORT

Submitted By

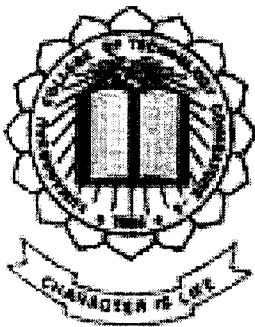
**C. Dayanand  
M. Kalaivanan  
B. Raja  
M. Sujan**

Guided By

**Mrs. K. Malarvizhi, B.E.,  
Lecturer, EEE Department.**

Sponsored by

**Tamil Nadu State Council for  
Science and Technology**



2001 - 2002

PROJECT REPORT  
SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE  
AWARD OF THE DEGREE OF  
BACHELOR OF ENGINEERING  
OF BHARATHIAR UNIVERSITY,  
COIMBATORE.

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS  
ENGINEERING**

*KUMARAGURU COLLEGE OF TECHNOLOGY*  
**COIMBATORE 641 006**

**KUMARAGURU COLLEGE OF TECHNOLOGY  
COIMBATORE-641 006.**

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS  
ENGINEERING**

**CERTIFICATE**

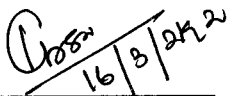
P. 712




This is to certify that the project entitled  
**"DIGITAL ENERGY METER CUM COST INDICATOR"**  
Has been submitted by

Mr./Ms. \_\_\_\_\_

in partial fulfillment of the requirements for the award of  
**BACHELOR OF ENGINEERING IN ELECTRICAL AND  
ELECTRONICS ENGINEERING**  
branch of the Bharathiar University, Coimbatore, during the academic  
year 2001-2002.

  
\_\_\_\_\_  
(Guide)

  
V. DURAISAMY, M.E., M.I.S.T.E  
Assistant Professor of Elec. Engg.  
(Head of Department)  
Kumaraguru College of Tech.  
COIMBATORE - 641 006.

Certified that the candidate with the Register  
no. \_\_\_\_\_ was examined in the Project work and Viva-voce  
Examination held on \_\_\_\_\_

\_\_\_\_\_  
(Internal Examiner)

\_\_\_\_\_  
(External Examiner)

**\* Chapter 5**

**# Interfacing And Control**

**@ Analog to Digital Converter**

**@ Decoders**

**@ Address Latches**

**@ LCD Module**

**@ 8255 PPI**

**\* Chapter 6**

**# Software**

**# Algorithm**

**\* Test Results**

**\* Conclusion**

**\* Appendix**

## **ACKNOWLEDGEMENT**

## **ACKNOWLEDGEMENT**

We would like to express our deep sense of gratitude towards our project guide Mrs.Malarvizhi K., B.E. for directing us towards a more application oriented project and for her unstinted support and interest in all our endeavors.

We would also like to thank our Project Co-coordinator, Mr.V Duraisamy, M.E., Assistant Professor, for having channelized our thoughts towards this project.

We would like to acknowledge the support shown by our beloved Principal, Dr .K. K. Padmanabhan, M.E.,Ph.D.

Further we would like to thank our former Head of the Department, Dr .K. A. Palaniswamy, M.Sc. (Engineering), Ph.D. for his motivation and all our staff members for having shown interest in our project.

Finally we would like to conclude with a special thanks to the Tamil Nadu State Council for Science and Technology, for having shown faith and appreciation in our project and subsequently sponsoring the same.

## **SYNOPSIS**

## **SYNOPSIS**

The micro controller based Digital Watt Hour Meter Cum Cost Indicator is designed to measure and display the number of units consumed and its corresponding cost. Now all houses, apartments, offices and industries are equipped with energy meter. The number of units consumed for a certain period (say for a billing period of 1 or 2 months) can be found by calculating the difference between readings at the start and end of the period in an energy meter. The DWHMCCI digitally displays the number of units consumed and the corresponding cost for it.

The 8951 micro controller here is chosen for reasons like its low cost, easy programming, lesser programming time, on-chip flash program memory etc. The purpose of using a micro controller here is to calculate the energy consumption and subsequently the cost (tariff).

The 8255 PPI is used to interface the micro controller with the LCD that displays the cost and the units of energy consumed.

It has been made easy for the readers to digest each data as clear pictures (wherever required) have been provided in all chapters.

# CONTENTS

**\* Acknowledgement**

**\* Synopsis**

**\* Chapter 1**

- # Introduction**
- # Electrical Energy**
- # Tariff Structures**
- # Energy Meter**
- # Need for DWHMCCI**

**\* Chapter 2**

- # Block Diagram of Micro Controller Based Energy Meter**
- # Meter**

**\* Chapter 3**

- # Hardware Review**
- # Voltage Sensing Circuit**
- # Current Sensing Circuit**
- # Power Factor Sensing Circuit**
- # Power Supply**

**\* Chapter 4**

- # Introduction to Micro Controller**
  - @ Need For 89C51**
  - @ Architecture**
  - @ Pin Description**
  - @ Memory Organization**
  - @ Timer And Counters**
  - @ Power Saving Modes of Operation**
  - @ Idle Mode**



# **INTRODUCTION**

# CHAPTER 1

## INTRODUCTION

Energy plays a significant role in development of nation. Today there is a trend toward tremendous growth and phenomenal modernization of electric system throughout the world. The amount of electrical energy supplied to a consumer is metered from the very day it become a stable commodity. Energy meters are essentially integrating instruments reading a continuous quantity and are quite different from other indicating instruments. Energy is infact power integrated over a period of time. Thus essentially they measure power over a period of time, integrating it to energy.

Induction type meters are the most common form of AC energy meters used in everyday domestic and industrial installation to measure electric energy supplied to customer in a given time. Since the practical unit in which energy meters measure energy is KWH, they are often referred to as watt- hour meters. Errors in conventional energy meters are mainly due to fractional forces at the rotor bearings and in the register mechanism. Ageing effect of brake magnet also contributes to errors in measurement. Such errors may result in loss either to the consumer or to the supplier. This situation calls for economically viable and accurate measurement of energy. An attempt has been made in this direction to design and develop a digital watt hour meter.

The TNEB tariff for most of the organizations like schools, colleges, hospitals, offices, residential houses, hotels and shops falls in one of the first three types of tariff.

### **1.3 ENERGY METER:**

Induction type energy meter is universally used for measurement of energy in domestic and industrial AC circuits. They possess lower friction and higher torque/weight ratio. They are inexpensive and accurate and retain their accuracy over a wide range of loads and temperature conditions.

#### **WORKING OF INDUCTION TYPE METERS:**

We have two fluxes produced by currents flowing in the windings of the instrument. These fluxes are alternating in nature and so they produce EMFs in metallic disk provided for the purpose as shown in figure. These EMFs in turn circulate currents in the metallic disc. Thus we have 2 fluxes and two eddy currents and therefore 2 torques are produced by first flux interacting with eddy current produced by the second flux and second flux interacting with eddy current produced by the first flux.

Therefore, Total torque is the sum of these two torques.

The current coil is connected in series with line and voltage coil is connected as shown in figure. Both coils are wound on a metal frame of special design, providing two magnetic circuits. A light aluminum disc suspended in the air gap of the current coil field, causes eddy currents to flow in the disc. The reaction of the eddy currents and the field of the voltage coil create a torque (motor action) on the disc, causing it to rotate. The developed torque is proportional to the field strength of the voltage coil and eddy currents in the disc, which is in turn a function of the field strength

## 1.1 ELECTRICAL ENERGY:

The capacity of an agent to do work is known as its energy. The most important forms of energy are mechanical, electrical and thermal energy. Here we concentrate on electrical energy. The unit of electrical energy is watt-sec or joule and is defined as follows:

1 watt-sec or 1 joule of energy is transferred between two points if a potential difference of 1 volt exists between them and 1 ampere current passes between them for 1 second.

*Electrical Energy in watt-second – voltage in volts \* current in amps \* time  
in secs.*

Joule or watt-sec is very small unit of electrical energy for practical purposes. In practice, for the measurement of electrical energy, bigger units like Watt-hour & Kilowatt-hour are used.

$$1 \text{ Watt-hour} = 1 \text{ Watt} \times 1 \text{ hour}$$

$$1 \text{ Kilowatt-hour} = 1 \text{ Kilowatt} \times 1 \text{ hour}$$

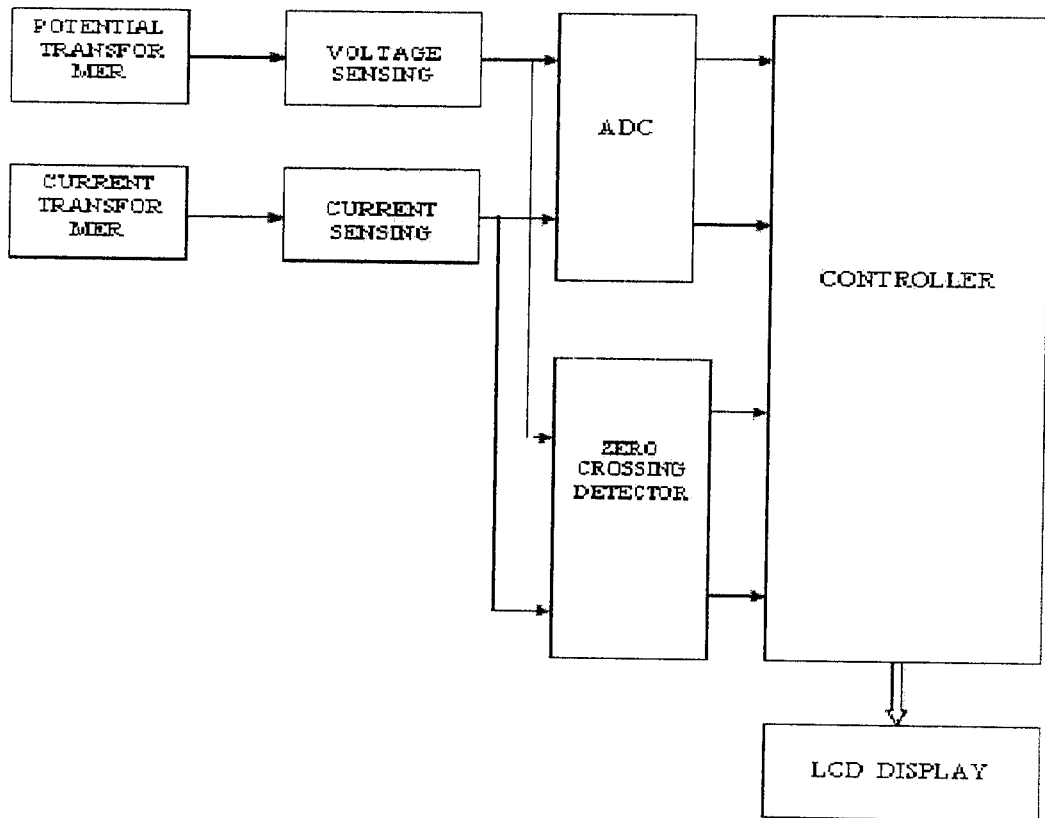
## 1.2 TARIFF:

The rate at which electric energy is supplied to a consumer is known as **tariff**. The various types of tariff are

- ❖ Simple tariff
- ❖ Flat rate tariff
- ❖ Block rate tariff
- ❖ Two part tariff
- ❖ Maximum demand tariff
- ❖ Power factor tariff
- ❖ Three part tariff

**BLOCKDIAGRAM**

# BLOCK DIAGRAM



## **CHAPTER 2**

### **BLOCK DIAGRAM OF MICRO-CONTROLLER BASED ENERGY METER**

A tapping from the secondary of P.T and C.T are taken and given to the precision rectifier and output is given to ADC. Then ADC output is given to the micro controller.

The tappings from the P.T and C.T are also given to the zero crossing detectors to finding instant values.

The time difference between the voltage and current wave is found by micro controller. The power factor value is proportional to the time difference is picked up from the lookup table and is displayed.

The various parameters like voltage, current, power factor and cost per KWH consumed are multiplied in micro controller and result is displayed and stored.

## CHAPTER 3

### HARDWARE REVIEW

#### 3.1 VOLTAGE SENSING CIRCUIT:

The voltage sensing circuit consists of 230V/6V potential transformer; op-amp-741 (2 nos.), diode-4148 (2 nos.), dual op-amp-4558, capacitor 220 $\mu$ F/16V, pot of 10K and various biasing and feed back resistors.

The primary of the 230V/6V potential transformer is connected to load voltage via M and V terminals. The respective secondary voltage signal is given as input to the precision rectifier, which is the absolute value circuit, which is obtained by summing the output of half wave and its input with the proper phase and amplitude relations.

Here A1 is an inverting rectifier. The output from A1 is added to the original input signal A2 with the signal amplitude and phase relations shown. Negative alterations of  $E_{in}$ , result in no output at E1 due to the rectification,  $E_{in}$  feeds A2 through a 20K $\Omega$  resistor and E1 feeds A2 through a 10K $\Omega$  resistor. The net effect of this scaling is that for equal amplitudes of  $E_{in}$  and E1, E1 will 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 alteration of  $E_{in}$ . This causes a current of precisely half the amplitude, which W1 alone would



of the current coil. The number of rotations of the disc is therefore proportional to the energy consumed by the load over a certain time period and is measured in KWH. The shaft that supports the aluminum disc is connected by a gear arrangement to the clock mechanism on the front of the meter, providing a decimally calibrated readout of the number of KWH.

The floating shaft watt-hour meter uses a unique design to suspend the disc. The rotating shaft has a small magnet at either end. The upper magnet of the shaft is attracted to a magnet in the upper bearing, and the lower magnet of the shaft is attracted to magnet in the lower bearing. The movement thus floats without touching both bearing surfaces and the only contact with the gear is that of the connecting the shaft with the gear train.

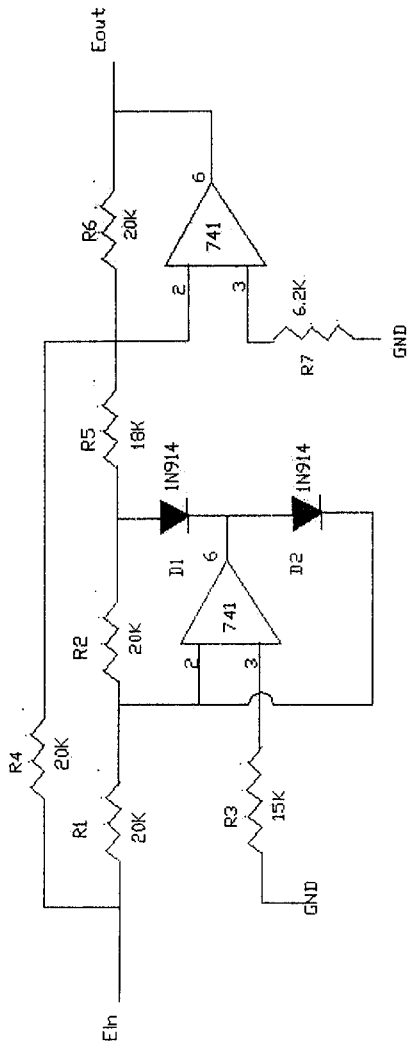
#### **1.4 NEED FOR DWHMCCI:**

From the analog meter the energy consumed for a certain period can be found by calculating the difference between the readings at the start and the end of the period from the energy meter. The DWHMCCI can display the energy consumed for a period along with the cost for that many units. Therefore the need for the manual calculations has been eliminated.

In apartments and other complexes there are several consumers using separate sub-meters. Here the DWHMCCI can be used with these sub-meters to calculate the energy consumption and cost.

For small industries, in order to operate the industry economically, luxury equipment like AC, Refrigerator, Fans etc., can be switched off when the rate goes beyond a certain value.

The electromechanical energy meters, which we use, takes into account variable load and constant load. The variable load is not taken in to consideration in case of the digital energy meter.



VOLTAGE SENSING CIRCUIT

generate due to the subtraction of  $E_{in}$ . It is the equivalent of having  $E_1$  feed through a  $20K\Omega$  input resistor and having  $E_{in}$  non-existent during this half cycle, and it results in a positive going output at A2. During negative alterations of  $E_{in}$ ,  $E_1$  is absent and  $E_{in}$  produces the alternate positive output swing that, in summation, produces the desired full wave rectified response. As before, operation with opposite output polarity is possible by reversing D1 and D2.

### 3.2 CURRENT SENSING CIRCUIT:

The primary of the 15/5A current transformer is connected to the load current via M and L terminals. The respective secondary current signal is given as input to I to V converter, which is a series connected resistor of  $47\Omega$ , 2W. Then the output of the I to V converter is given as input to the precision rectifier, which is an absolute value circuit, which is obtained by summing the output of half wave and its input with the proper phase and amplitude relations.

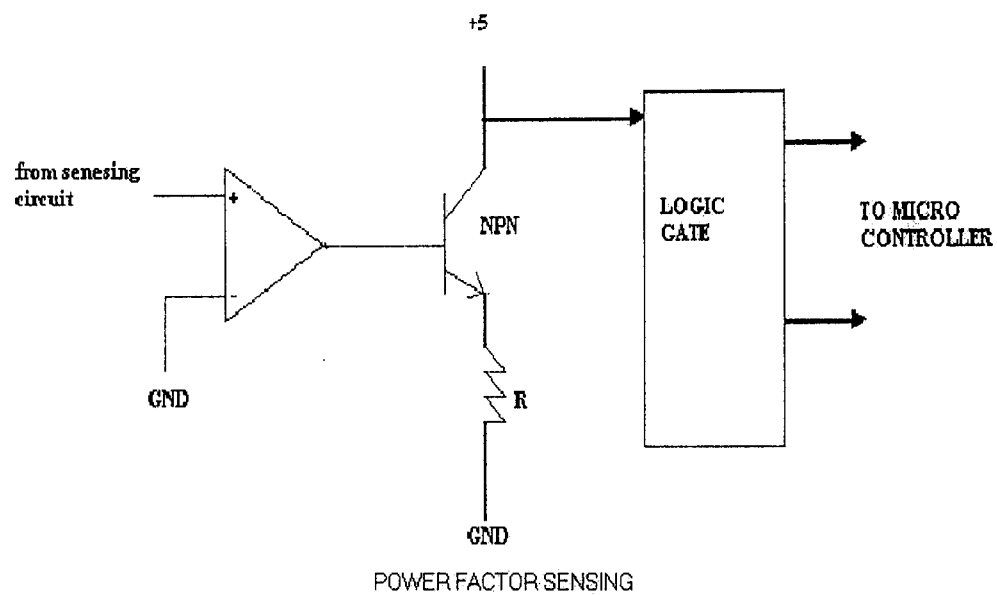
Here A1 is an inverting rectifier. The output from A1 is added to the original input signal A2, with the signal amplitude and phase relations shown. Negative alterations of  $E_{in}$  result in no output at E1 due to the rectification.  $E_{in}$  feeds A2 through a  $20K\Omega$  resistor, and E1 feeds A2 through a  $10K\Omega$  resistor. The net effect of this scaling is that, for equal amplitudes of  $E_m$  and E1, E1 will provide twice as much current into the summing point. This tact is used to advantage here, as the negative alteration of E1 produces twice the input current as that caused by the positive alteration of  $E_m$ . This causes a current of precisely half the amplitude, which W1 alone would generate due to the subtraction of  $E_{in}$ . It is the equivalent of having E1 fed through a  $20K\Omega$  input resistor and having  $E_{in}$  non-existent during this half cycle, and it results in a positive going output at A2. During negative alternations of  $E_{in}$ , E1 is absent and  $E_{in}$  produces the alternate positive output swing that in summation produces the

desired full wave rectified response. As before, operation with opposite output polarity is possible by reversing D1 and D2.

### **3.3POWER FACTOR SENSING CIRCUIT:**

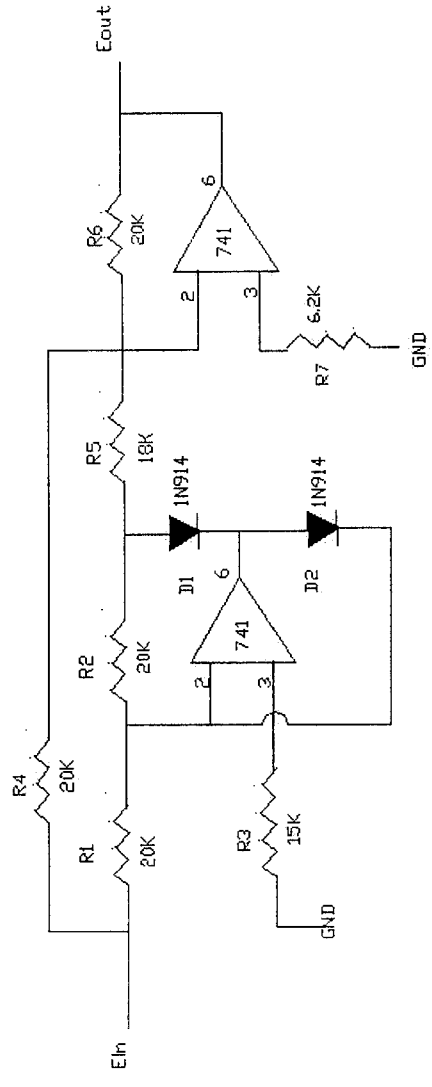
The respective voltage and current signals are taken from P.T and C.T and fed as input to the respective zero crossing detectors which convert the sine wave signal into square wave signal. These square wave signals have the amplitude of 12V, but the microcontroller requires only 5V square signals, hence these 12V signals are switched as 5V signals using transistor of BC547.

These 5V square wave signals are fed as input to the logic circuit to have the exact phase difference between voltage and current in terms of square pulse. This square pulse is fed to port 1.0 of microcontroller.



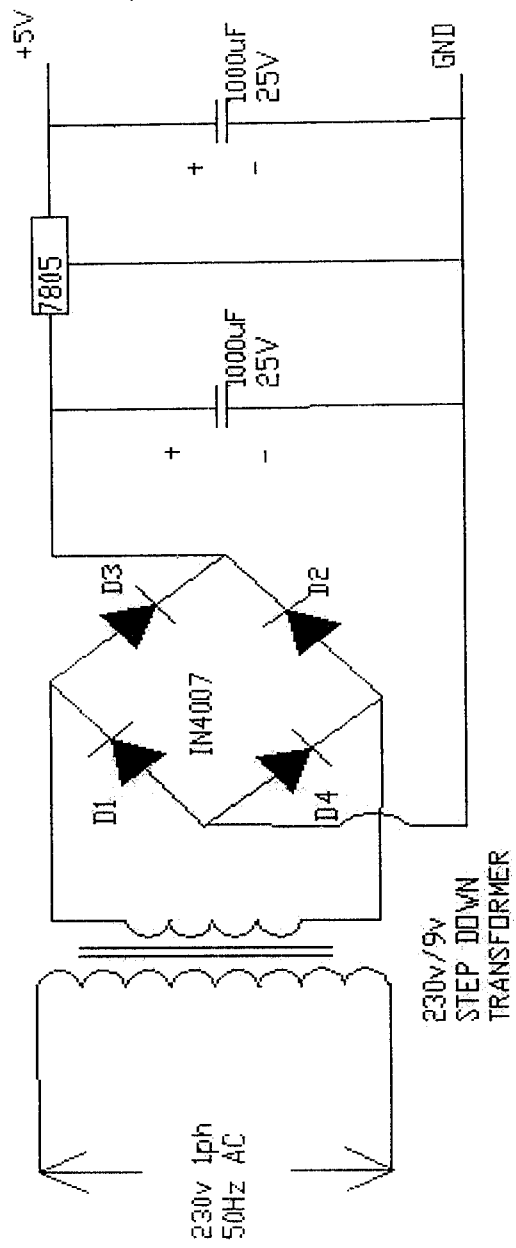
### **3.4 POWER SUPPLY:**

A block diagram containing the parts of a typical power supply and the voltage at various points in the unit is shown in fig . The ac voltage, typically 120V rms, is connected to a transformer, which steps that ac voltage down to the level for the desired dc output. 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 variation. A regulator circuit can use this dc input to provide a dc voltage that not only has much less ripple voltage but also remains the same dc value even if the input dc voltage varies somewhat, or the load connected to the output dc voltage changes. This voltage regulation is usually obtained using one of a number of popular voltage regulator IC units.



CURRENT SENSING CIRCUIT





P. 712



POWER SUPPLY CIRCUIT

**INTRODUCTION TO MICRO**  
**CONTROLLER**

## CHAPTER 4

### INTRODUCTION TO MICROCONTROLLER

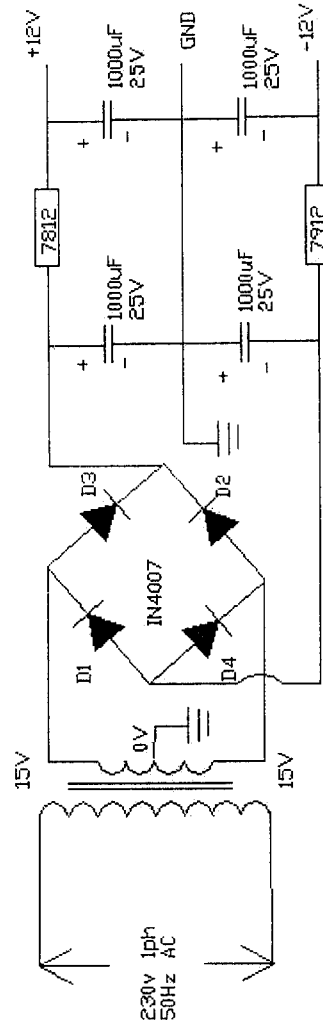
#### 4.1 INTRODUCTION TO MICROCONTROLLER:

Microcomputer is a microprocessor combined with RAM, ROM, I/O facilities all on a single chip. Such micro controller families are MCS-48, MCS-51. Because this family of devices was particularly oriented towards control applications it was dubbed a “micro controller”. 8951 comes under MCS-51 family.

#### 4.2 NEED FOR 8951:

8951 is an 8051 compatible micro controller with on board flash PROM. The flash code memory can be erased electrically in milliseconds with no need for an UV eraser, and such flash micro controller devices can be reliably erased and reprogrammed for over 1000 times.

This technology represents enormous time saving, but its advantages are not just confined to the development aspect. If numerous code revisions are the norm before all the bugs are ironed out, flash offers the route overall programming flexibility. The latest code revision can be programmed into a device in seconds whilst it is actually progressing down the production line. With the addition of the appropriate hardware it is even possible to program these devices actually in circuit without removing the device from the product. If the final requires frequent code updates, then once again flash is the answer.



230V/15-0-15  
STEP DOWN  
TRANSFORMER

POWER SUPPLY CIRCUIT

### **4.3 ARCHITECTURE:**

To utilize micro controller effectively, a designer must have a fundamental understanding of available components which begins with the micro controller itself with its CPU register structure, its instruction set, its addressing modes, its on-chip resources, key switches for setting up on an instrument or devices, displays for showing this setup information to the users, output transducers for sensing inputs and actuators for control.

The designer must thoroughly understand the algorithmic process required by each aspect of the design and be able to translate them into the language of microcontroller and implementation of its control via a sequence of microcontroller instructions to realize an effective organization of hardware and software so as to meet the specification for an instrument or device.

The MCS-51 family was developed in view of the increasing industry standard. It is a high performance microcontroller family in the world with outstanding performance records in various applications. The 8951 is one among the MCS-51 family and are a stand alone, high performance control oriented CPU with 4KB flash type chip program memory. It can accept 64KB in program memory and data memory, with the memory being combined. This version reduces development problems to a minimum and provides maximum flexibility. The block diagram of 8951 is shown in figure.

The major features of 8951 are:

- ❖ 8-bit CPU
- ❖ On-chip oscillator
- ❖ 128 bytes of RAM.
- ❖ 21 special function register.
- ❖ 32 I/O lines
- ❖ 64K address space for external program memory
- ❖ 64K address for external data memory
- ❖ Two 16 bit timer/counters
- ❖ A five source interrupt structure with two priority levels.
- ❖ A full duplex serial port.
- ❖ Bit address ability for Boolean processing

#### **4.4 PIN DESCRIPTION:**

The 8951-pin configuration is as shown in figure. The various functions are explained below:

$V_{ss}$  = Circuit ground potential.

$V_{cc}$  = Supply voltage during programming, verification and normal operation.

##### ***PORT 0:***

Port 0 is an 8-bit open bi-directional I/O port. It is also the multiplexed lower order address and data bus during access to external memory (during which accesses it activates internal pull-ups). It also outputs instruction bytes during program verification. (External pull-ups are required during program verification). Port 0 can sink eight LS TTL inputs.

##### ***PORT 1:***

**ALE/PROG:**

Address latch enables output for latching the low byte of the address during accesses to external memory. ALE is activated for this purpose at a constant rate of  $1/6^{\text{th}}$  oscillator frequency even when external memory is not being accessed. Consequently it can be used for external clocking or timing purposes. (However one ALE pulse is skipped during each access to external data memory).

**PSEN:**

Program strobe enable output is the read strobe external program memory. PSEN is activated twice each machine cycle during fetches from external memory. (However when executing out of external program memory two activation of PSEN is skipped during each access to external data memory). PSEN is not activated during fetches from internal program memory.

**EA/V<sub>PP</sub>:**

When EA is held high the CPU executes out of internal program memory (Unless the program counter exceeds OFFFH). When EA is held low the CPU executes out external program memory. In 8031 EA must be externally wired low.

**XTAL 1:**

Input to the inverting amplifier that forms the oscillator should be grounded when an external oscillator is used.

**XTAL 2:**

Output of the inverting amplifier that forms the oscillator, and output to the internal clock generator receives the external oscillator signal when an external oscillator is used.

Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. It receives the lower order address byte during program verification in 8951. Port 1 can sink/source three LS TTL inputs. It can derive MOS inputs without external pull-ups.

**PORT 2:**

Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. It emits the higher order address byte during accesses to external memory. It also receives the higher order address bits and control signals during program verification in the 8951. Port 2 can sink/source three LS TTL inputs. It can drive MOS inputs without external pull-ups.

**PORT 3:**

Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. It also serves the various functions of the MCS-51 family as listed below:

Port Pin	Alternate Function
P 3.0	RXD(Serial input port)
P 3.1	TXD(Serial output port)
P 3.2	INT0(External interrupt)
P 3.3	INT1(External interrupt)
P 3.4	To(Timer 0 external input)
P 3.5	T1(Timer 1 external input)
P 3.6	WR(External data memory write strobe)
P 3.7	RD(External data memory read strobe)

**RST/V<sub>PD</sub>:**

A high level on this pin for two machine cycles while the oscillator is running resets the device. An internal pull down permits Power On reset using only a capacitor connected to V<sub>cc</sub>.



#### **4.4 MEMORY ORGANIZATION:**

All flash micro controller have separate address spaces for program and data memory as shown in fig. The logical separation of program and data memory allows the data memory to be accessed by 8 bit addresses, which can be more quickly stored and manipulated by an 8-bit CPU. Nevertheless, 16 bit data memory addresses can also be generated through the DPTR register.

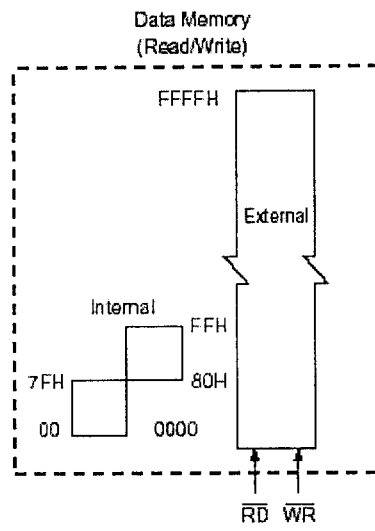
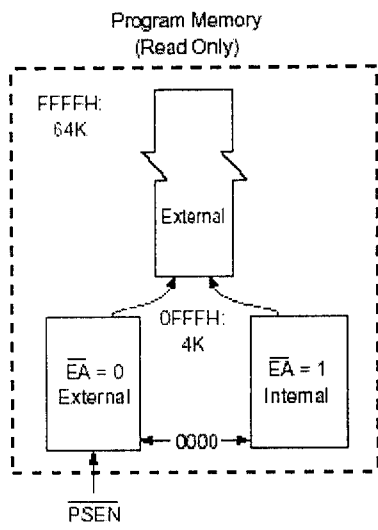
Program memory can also be read. There can be up to 64 KB of directly addressable program memory. The read strobe for external program memory is the program strobe enable. (PSEN)

Data memory occupies a separate address space from program memory. The CPU generates read and write signals, RD and WR during external data memory accesses.

External program memory and external data memory can be combined by applying the RD and PSEN signals to the inputs of AND gate and using the output of the gate as the read strobe to the external program/data memory.

The internal memory space is divided in to three blocks, which are referred to as the lower 128, the upper 128 and SFR space. Internal data memory addresses are always one byte wide, which implies an address of only 256 bytes. However the addressing modes for internal RAM can accommodate 384 bytes. Direct address higher than 7Fh access one memory space and indirect addresses higher than 7FH access a different memory space.

Fig shows how the lower 128 bytes of RAM are mapped. The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as RO through R7. Two bits in the



program status word select which register bank is in use. This architecture allows more efficient use of code space, since register instructions are shorter than instructions that uses direct addressing.

The next 16 bytes above the register banks form a block of bit addressable memory space. The micro controller instruction set includes a wide selection of single bit instructions can directly address the 128 bits in this are. These bit addresses are from 00H through 7FH.

Either direct or indirect addressing can access all of the bytes in the lower 128. Indirect addressing can only access the upper 128. The upper 128 bytes of RAM are only in the devices with 256 bytes of RAM. The list of bit addressable registers in the SFR space is shown in fig.

The lowest addresses of program can be either in the on chip flash or in external memory. To make this selection strap the external access pin either  $V_{cc}$  or GND.

#### **4.5 TIMERS/COUNTERS:**

The 8951 provided two 16 bit registers, timer 0 and timer 1, that can be used as timers or event counter. For each timer/counter register there is a control bit in special function register TMOD that selects the timer/counter function to be timer or counter.

In the timer function the register is incremented for every machine cycle. Thus one can think of it as counting machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the counter function the register is incremented in response to a 1 to 0 transition at its corresponding external input pin, T0 or T1. Since it takes 2 machine cycles (24 oscillator periods) to recognize a 1 to 0

transition, the maximum count rate is 1/24 of the oscillator frequency. There are no restrictions on the duty cycle of the external input signal, but to ensure that a given level is sampled at least once before it changes, it should hold for at least one full machine cycle.

In addition to the timer or counter selection, each timer/counter has four operating modes from which to select. Operating modes 0, 1, 2 are the same for both timer/counter. Mode 3 is different. The four operating modes are described below.

#### **MODE 0:**

Putting either timer in mode 0 makes it to look like 8048 timers, which is an 8-bit counter with a divide by 32-pre scalars.

In this mode the timer register is configured as a 13-bit register. As the count rolls over from all 1's to all 0's, it set the timer interrupt flag TF1. The counter input is enabled to the timer when TR1=1 and either GATE=0 to INT1=1. (Setting GATE=1 allows the timer to be controlled by external input INT1, to facilitate pulse with measurements) TR1 is control bits in special function register TCON. GATE is control bits in special function register TMOD.

The 13-bit register consists of all 8 bits of TH1 and the lower 5 bits of TL1. The upper 3 bits of TL1 is indeterminate and should be ignored. Setting the run flag does not clear the registers.

Mode 0 operation is the same for timer 0 as for timer 1. There are two different GATE bits, one for timer1 and one for timer0.

#### **MODE 1:**

Mode 1 is the same as mode 0; expect that the timer register be being run with all 16 bits.

#### **MODE 2:**

Mode 2 configures the timer register as an 8-bit counter with automatic reload. Overflow from TL1 not only sets TF1 but also reloads TL1 with the contents of TH1, which is preset by software to any desired one-byte value. The reload leaves TH1 unchanged.

**MODE 3:**

If you put the timer in to mode 3, it holds its count. The effect is the same as setting TR1=0. If timer 0 is put in mode, TL0 and TH0 become two separate counters. TL0 is using all of the timer 0 control bits: C/T, GATE, TR0, INT0 and TF0. TH0 is locked into a timer function and has taken over the use of TR1 and TF1 from timer 1.

Mode 3 is provided specially for applications in which two independent timer/counters plus the serial port are required. Timer 1 can be set up as the baud rate generator (mode 2) and operate timer 0 in mode 3.

**4.6 TIMER CONTROL AND STATUS REGISTERS:**

Two special function registers TMOD and TCON are used to define the operating modes and control the functions of the timers/counters. When an instruction changes the contents of TMOD or TCON, the change is latched into the SFR and takes effect at S1 P1 of the next instruction first cycle. The registers are shown below.

**Timer Mode Control Register (TMOD):**

GATE	C/T	M1	M0	GATE	C/T	M1	M0
------	-----	----	----	------	-----	----	----

Where M1 and M0 specify the mode, as follows

M1	M0	MODE	DESCRIPTION
0	0	0	13 bit counter
0	1	1	16 bit counter
1	0	2	8 bit counter with auto reload
1	1	3	Split timer 0 into two 8 bit counter or

			stop timer 1
--	--	--	--------------

### Timer Control Register (TCON):

TF1	TCON.7	Timer 1 overflow flag
TR1	TCON.6	Timer 1 run control bit
TF0	TCON.5	Timer 0 overflow flag
TR0	TCON.4	Timer 0 control bit
IE1	TCON.3	External interrupt 1 edge flag
IT1	TCON.2	Interrupt 1 type flag
IE0	TCON.1	External interrupt 0 edge flag
IT0	TCON.0	Interrupt 0 type flag

### INTERRUPTS:

Interrupts are used to interrupt the normal function of the micro controller at times of emergency like power failure. Interrupts are also used for various other functions like keyboard interface, process control, etc. Interrupts are of two types namely software and hardware interrupts. Anyhow interrupts make the whole system user friendly.

### INTERRUPT SOURCES:

The 8951 provide 5 interrupt sources: two external interrupts, two timers interrupts, and a serial port interrupt.

The external interrupts INT0 and INT1 can each be either level activated or transition activated, depending on bits IT0 and IT1 in register TCON. The flags that actually generate these interrupts are the IE0 and IE1 bits in TCON. When the service routine is vectored to, hardware clears the flag that generated an external interrupt only if the interrupt was transition activated. If the interrupt was level activated, then the external requesting source controls the request flag.

The timer 0 and timer 1 interrupts are generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine normally must determine whether RI and TI generated the interrupt, and the bit must be cleared in software.

All of the bits that generate interrupts can be set or cleared software, with the same result as though they had been set or cleared by hardware. That is, interrupts can be generated and pending interrupts can be canceled in software.

Each of the interrupt sources can be individually enable or disable by setting or clearing a bit in special function register IE at address 0A8H. As well as individual enable bits for each interrupt sources, there is a global enable/disable bit that is cleared to disable all interrupts or set to turn on interrupts.

### **IE: INTERRUPT ENABLE REGISTER:**

If bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	IE.7	Disables all interrupts. If EA=0 no interrupt will be acknowledged. If EA=1 each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
--	IE.6	Not implemented
--	IE.5	Not implemented
ES	IE.4	Enable or disable serial port interrupts
ET1	IE.3	Enable or disable timer 1 overflows interrupt
EX1	IE.2	Enable or disable external interrupt 1
ET0	IE.1	Enable or disable timer 0 overflow interrupt
EX0	IE.0	Enable or disable external interrupt 0

## **PRIORITY LEVEL STRUCTURE:**

Each interrupt source can also be individually programmed to one of the two priority levels by setting or clearing a bit in special function register IP at address 0B8H. IP is cleared after a system reset to place all interrupts at the lower priority level by default. A low priority interrupt can be interrupted by a high priority interrupt but not by another low priority interrupt. A high priority interrupt cannot be interrupted by any other source.

## **IP: INTERRUPT PRIORITY REGISTER:**

If the bit is 0, the corresponding interrupt has lower priority and if the bit is 1 the corresponding interrupt has highest priority.

--	IP.7	Not implemented
--	IP.6	Not implemented
--	IP.5	Not implemented
PS	IP.4	Defines the serial port interrupt priority level
PT1	IP.3	Defines the timer 1 interrupt priority level
PX1	IP.2	Defines the External interrupt 1 priority level
PT0	IP.1	Defines the timer 0 interrupt priority level
PX0	IP.0	Defines the external interrupt 0 priority level

If two requests of different priority levels are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence, as follows.

SL.NO	SOURCE	PRIORITY WITHIN LEVEL
1	IE0	(Highest)
2	TF0	



3	IE1	
4	TF1	
5	RI+TI	(Lowest)

### RESET:

The reset input is RST pin, which has the input to an Schmitt-trigger. A reset is accomplished by holding the RST pin high for at least two machine cycles (24 oscillator periods), while the oscillator is running. The CPU responds by generating an internal reset. The external reset signal is asynchronous to the internal clock. The internal reset algorithm, writes 0s to all the SFR's except the port latches, the stack pointer and SBUF. The port latches are initialized to FFH, the stack pointer to 07H, and SBUF is indeterminate. The internal RAM is not affected by reset. On power up the RAM content is indeterminate.

### 4.7 POWER ON RESET:

For CMOS devices, the external resistor can be removed because the RST pin has an internal pull down. The capacitor value can then be reduced to 1 $\mu$ F.

When the power is turned on, the circuit holds the RST pin high for an amount of time that depends on the capacitor value and the rate at which it charges. To ensure a valid reset, the RST pin must be held high long enough to allow the oscillator to start up plus two machine cycles.

On power up  $V_{cc}$  should rise within approximately 10 ms. The oscillator start up time depends on the oscillator frequency. With the given circuit, reducing  $V_{cc}$  quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited and will not harm the device.

**Note:** The port pins will be in a random state until the oscillator has started and the internal reset algorithm has written to them.

Powering up the device without a valid reset could cause the CPU to start executing instructions from an indeterminate location. This is because the SFRs. Specifically the program counter may not get properly initialized.

#### 4.8 POWER SAVING MODES OF OPERATION:

The ATMEL micro controllers have two power reducing modes. Idle and power down. The input through which backup power is supplied during these operations is  $V_{cc}$ . The idle mode (IDL=1) the oscillator continues to run and the interrupt. Serial port and timer blocks continue to be clocked, but the clock signal is gated off to the CPU. In power down (PD=1) the oscillator is frozen. Setting bits in special function register PCON activates the idle and power down modes. The address of this register is 87H.

#### 4.9 SPECIAL FUNCTION REGISTERS AT THEIR VALUE AT RESET

SYMBOL	NAME OF SPECIAL FUNCTION REGISTER	ADDRESS	VALUE AT RESET
*ACC	Accumulator	E0H	00000000B
*B	B Register	F0H	00000000B
*PSW	Program status word	D0H	00000000B
SP	Stack Pointer	81H	07H
DPTR	Data Pointer		
DPH	High byte	82H	00H
DPL	Low byte	83H	00H
*P0	Port 0	80H	11111111B
*P1	Port 1	90H	11111111B

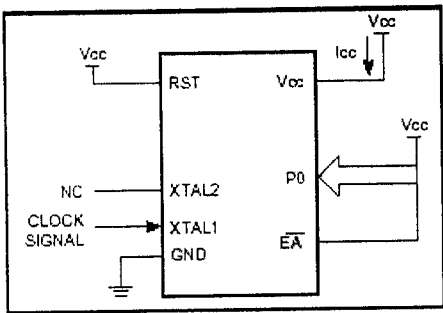
*P2	Port 2	A0H	11111111B
*P3	Port 3	D0H	11111111B
*IP	Interrupt priority control	B8H	xxx00000
*IE	Interrupt enable control	A8H	B
TMOD	Timer/Counter mode control	89H	0xx00000
*TCON	Timer/Counter control	88H	B
TH0	Timer/Counter 0 High byte	8AH	00H
TL0	Timer/Counter 0 Low byte	8DH	00000000B
TH1	Timer/Counter 1 High byte	8BH	00H
TL1	Timer/Counter 1 Low byte	98H	00H
*SCON	Serial control	99H	00H
SBUF	Serial Buffer	87H	00000000B
PCON	Power control		xxH

### PCON: POWER CONTROL REGISTER

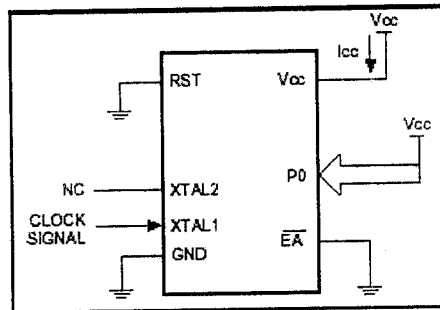
SMOD	PCON.7	Doubles the baud rate for serial transmission
--	PCON.6	Not implemented
--	PCON.5	Not implemented
--	PCON.4	Not implemented
GF1	PCON.3	General purpose flag1
GF0	PCON.2	General purpose flag0
PD	PCON.1	Power down mode set bit
IDL	PCON.0	Idle mode set bit

#### 4.10 IDLE MODE:

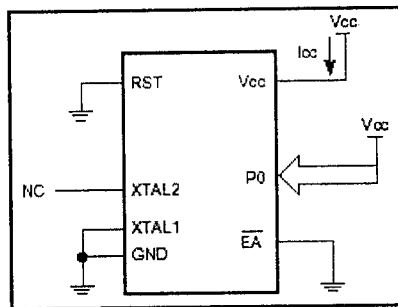
An instruction that sets PCON.0 is the last instruction executed before the idle mode begins. In the idle mode, the internal clock signal is gated off to the CPU but not to the interrupt, timer, and serial port functions. The CPU status preserved in its entirety; the stack pointer, program counter, program status word, accumulator and all other registers maintain their data during idle. The ports pins hold the logical states they had at the time idle was activated. ALE and PSEN hold at logic high levels.



Active Mode



Idle Mode



Power-down Mode

oscillator stops. With the clock frozen all functions are stopped, but the on-chip RAM and special function registers are held. The port pins output the values held by their respective SFRs, ALE and PSEN output lows.

In the power down mode operation  $V_{cc}$  can be reduced to as low as 2V. However,  $V_{cc}$  must not be reduced before the power down mode is invoked;  $V_{cc}$  must not be restored to its normal operating level before the power down mode is terminated. The reset should not be activated before  $V_{cc}$  is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize (Normally less than 10msec).

# CHAPTER 5

## INTERFACING AND CONTROL

### 5.1 ANALOG TO DIGITAL CONVERTER:

The ADC 0809 data acquisition component is a monolithic CMOS device with an 8-bit analog to digital converter, 8 channels 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 256 R voltage divider with analog switch tree and a successive approximation register. The 8 channels Multiplexer can directly access any of 8 single ended analog signals.

### 5.2 FEATURES:

- ❖ Resolution - 8 bit.
- ❖ Total unadjusted error -  $\pm 1/2$  LSB &  $\pm 1$  MSB.
- ❖ No missing codes.
- ❖ Conversion time - 100 $\mu$ s.
- ❖ Single Supply - 5 V<sub>dc</sub>.
- ❖ Operates arithmetically or with 5 V<sub>dc</sub> or analog span adjusted voltage reference.
- ❖ 8-channel multiplexer with latched control logic.
- ❖ Easy interface to all micro controllers or operates stand-alone.
- ❖ Output meet T<sup>2</sup> voltage level specifications.

# **INTERFACING AND CONTROL**

- ❖ 10V to 5V analog input voltage range with single 5V supply.
- ❖ No zero or full scale adjusts required.
- ❖ Standard hermetic or molded 28 pin DIP package.
- ❖ Temperature range -40°C to +85°C or -55°C to +125°C.
- ❖ Lower power consumption - 15 MW.
- ❖ Latched TRI-STATE output.

### **5.3 FUNCTION DESCRIPTION:**

The A/D converters 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 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 converts the DC input signal into an AC signal. This signal is then fed through a 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



There are two ways to terminate the idle. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the idle mode. The interrupt will be serviced, and following RETI the next instruction to be executed will be the one following the instruction that put the device into idle.

The flag bits GF0 and GF1 can be used to indicate whether an interrupt occurred during normal operation or during an idle. For example, an instruction that activates idle can also set one or both flag bits. When idle is terminated by an interrupt, the interrupt service routine can examine the flag bits.

The other way of terminating the idle mode is with hardware reset. Since the clock oscillator is still running, the hardware reset must be held active for only two machine cycles (24 oscillator periods) to complete the reset. The signal at the RST pin clears the IDL bit directly and synchronously. At this time, the CPU resumes program execution from where it left off that is at the instruction following the one that invoked the idle mode. Two or three machine cycles of program execution may take place before the internal reset algorithm takes control. On chip hardware inhibits access to the internal RAM during this time, but access to the port pins is not inhibited. To eliminate the possibility of unexpected outputs at the port pins, the instruction following the one that invokes idle should not write to a port pin or to external RAM.

#### **4.11 POWER DOWN MODE:**

An instruction that sets PCON.1 is the last instruction executed before power down mode begins. In the power down mode, the on chip

## **5.5 LCD DISPLAY**

### **5.5.1 INTRODUCTION:**

Liquid crystal displays (LCDs) have materials, which combine the properties of both liquids and crystals. Rather than having a melting point, they have a temperature range within which the molecules are almost as mobile as they would be in a liquid, but are grouped together in an ordered form similar to a crystal.

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

One polariser is pasted outside the two glass panels. These polarisers would rotate the light rays passing through them to a definite angle, in a particular direction.

When the LCD is in the off state, light rays are rotated by the two polarisers and the liquid crystal, such that the light rays come out of the LCD without any orientation, and hence the LCD appears transparent.

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

The LCD's are lightweight with only a few millimeters thickness. Since the LCD's consume less power, they are compatible with low power electronic circuits, and can be powered for long durations.

The LCD's don't generate light and so light is needed to read the display. By using backlighting, reading is possible in the dark. The LCD's have long life and a wide operating temperature range.

Changing the display size or the layout size is relatively simple which makes the LCD's more customers friendly.

The LCDs used exclusively in watches, calculators and measuring instruments are the simple seven-segment displays, having a limited amount of numeric data. The recent advances in technology have resulted in better legibility, more information displaying capability and a wider temperature range. These have resulted in the LCDs being extensively used in telecommunications and entertainment electronics. The LCDs have even started replacing the cathode ray tubes (CRTs) used for the display of text and graphics, and also in small TV applications.

### **5.5.2 POWERSUPPLY:**

The power supply should be of +5V, with maximum allowable transients of 10mV. To achieve a better / suitable contrast for the display, the voltage (VL) at pin 3 should be adjusted properly.

A module should not be inserted or removed from a live circuit. The ground terminal of the power supply must be isolated properly so that no voltage is induced in it. The module should be isolated from the other circuits, so that stray voltages are not induced, which could cause a flickering display.

### **5.5.3 HARDWARE:**

Develop a uniquely decoded 'E' strobe pulse, active high, to accompany each module transaction. Address or control lines can be assigned to drive the RS and R/W inputs.

Utilize the Host's extended timing mode, if available, when transacting with the module. Use instructions, which prolong the Read and Write or other appropriate data strobes, so as to realize the interface timing requirements.

If a parallel port is used to drive the RS, R/W and 'E' control lines, setting the 'E' bit simultaneously with RS and R/W would violate the module's set up time. A separate instruction should be used to achieve proper interfacing timing requirements.

#### **5.5.4 MOUNTING:**

Cover the display surface with a transparent protective plate, to protect the polarizer.

Don't touch the display surface with bare hands or any hard materials. This will stain the display area and degrade the insulation between terminals.

Do not use organic solvents to clean the display panel as these may adversely affect tape or with absorbant cotton and petroleum benzene.

The processing or even a slight deformation of the claws of the metal frame will have effect on the connection of the output signal and cause an abnormal display.

Do not damage or modify the pattern wiring, or drill attachment holes in the PCB. When assembling the module into another equipment, the space between the module and the fitting plate should have enough height, to avoid causing stress to the module surface.

Make sure that there is enough space behind the module, to dissipate the heat generated by the ICs while functioning for longer durations.

When an electrically powered screwdriver is used to install the module, ground it properly.

done simultaneous with the data write operation, the data may not be visible on the display.

If a character not found in the font table is displayed, or a character is missing, the CG ROM is faulty and the controller IC has to be changed

If particular pixels of the characters are missing, or not getting activated properly, there could be an assembling problem in the module.

In case any other problems are encountered you could send the module to our factory for testing and evaluation.

#### **5.5.8 CRYSTALONICS DISPLAY**

##### **INTRODUCTION:**

Crystalonics dot-matrix (alphanumeric) liquid crystal displays are available in TN, STN types, with or without backlight. The use of C-MOS LCD controller and driver ICs result in low power consumption. These modules can be interfaced with a 4-bit or 8-bit microprocessor /Micro controller.

The built-in controller IC has the following features:

- Correspond to high speed MPU interface (2MHz)
- 80 x 8 bit display RAM (80 Characters max)
- 9,920-bit character generator ROM for a total of 240 character fonts. 208 character fonts (5 x 8 dots) 32 character fonts (5 x 10 dots)

The first eight characters of a single line display, operated in the two-line display mode, as in CDM 16116.

The first line of characters of a two-line display as in CDM 16216 and 40216. The first and third line of characters of a four-line display operated in the two-line display mode, as in CDM 20416.

If the above mentioned does not occur, the module should be initialized by software.

Make sure that the control signals 'E', R/W and RS are according to the interface timing requirements.

#### **5.5.7 IMPROPER CHARACTER DISPLAY:**

When the characters to be displayed are missing between, the data read/write is too fast. A slower interfacing frequency would rectify the problem.

When uncertainty is there in the start of the first characters other than the specified ones are rewritten, check the initialization and the software routine.

In a multi-line display, if the display of characters in the subsequent lines doesn't take place properly, check the DD RAM addresses set for the corresponding display lines.

When it is unable to display data, even though it is present in the DD RAM, either the display on/off flag is in the off state or the display shift function is not set properly. When the display shift is

A/D converter extremely insensitive to temperature, long term drift and input offset errors.

## **5.4 DECODERS:**

### **5.4.1 FEATURES:**

- ❖ Demultiplexing capability.
- ❖ Multiple input enables easy expansion.
- ❖ Ideal for memory chip select decoding.
- ❖ Direct replacement for Intel 3205.

The 138-decoder accepts three binary weighted inputs ( $A_0$ ,  $A_1$ , and  $A_2$ ) and when enabled, provides eight mutually exclusive, active LOW outputs (0-7). The device features three enable inputs: two active LOW ( $E_1$ ,  $E_2$ ) and one active HIGH ( $E_3$ ). Every output will be HIGH unless  $E_1$  and  $E_2$  are low  $E_3$  is HIGH. This multiple enable function allows easy parallel expansion of the devices to a 1 of 32 decoder with just four 138s and one inverter.

### **5.4.2 ADDRESS LATCH FEATURES:**

- ❖ 8 bit transparent latch.
- ❖ 8-bit positive, edge triggered register.
- ❖ 3 state output buffers.
- ❖ Common 3-state output enables.
- ❖ Independent register and 3 state buffer operation.

The 373 is an octal transparent latch coupled to eight 3 state output buffers. The two sections of the device are controlled independently by latch enable and output enables control gates.



- 64 x 8 bit character generator RAM 8 character generator RAM 8 character fonts (5 x 8 dots) 4 characters fonts (5 x 10 dots)
- Programmable duty cycles
  - 1/8 – for one line of 5 x 8 dots with cursor
  - 1/11 – for one line of 5 x 10 dots with cursor
  - 1/16 – for one line of 5 x 8 dots with cursor
- Wide range of instruction functions display clear, cursor home, display on/off, cursor on/off, display character blink, cursor shift, display shift.
- Automatic reset circuit, that initializes the controller / driver ICs after power on.

### **5.5.9 FUNCTIONAL DESCRIPTION OF THE CONTROLLER IC:**

#### **REGISTERS:**

The controller IC has two 8 bit registers, an instruction register (IR) and a data register (DR). The IR stores the instruction codes and address information for display data RAM (DD RAM) and character generator RAM (CG RAM). The IR can be written, but not read by the MPU.

The DR temporally stores data to be written to /read from the DD RAM or CG RAM. The data written to DR by the MPU is

automatically written to the DD RAM or CG RAM as an internal operation.

When an address code is written to IR, the data is automatically transferred from the DD RAM or CG RAM to the DR. data transfer between the MPU is then completed when the MPU reads the DR. likewise, for the next MPU read of the DR, data in DD RAM or CG RAM at the address is sent to the DR automatically. Similarly, for the MPU write of the DR, the next DD RAM or CG RAM address is selected for the write operation.

The register selection table is as shown below:

RS	R/W	Operation
0	0	IR write as an internal operation
0	1	Read busy flag (DB7) and Address counter (DB0 to DB6)
1	0	DR write as an internal operation (DR to DD RAM or CG RAM)
1	1	DR read as an internal operation

### **BUSY FLAG:**

When the busy flag is 1, the controller is in the internal operation mode, and the next instruction will not be accepted.

When  $RS = 0$  and  $R/W = 1$ , the busy flag is output to DB7.

The next instruction must be written after ensuring that the busy flag is 0.

### **ADDRESS COUNTER:**

The address counter allocates the address for the DD RAM and CG RAM read/write operation when the instruction code for DD RAM address or CG RAM address setting, is input to IR, the address code is transferred from IR to the address counter. After writing/reading the display data to/from the DD RAM or CG RAM, the address counter increments/decrements by one the address, as an internal operation. The data of the address counter is output to DB0 to DB6 while  $R/W = 1$  and  $RS = 0$ .

### **DISPLAY DATA RAM (DD RAM)**

The characters to be displayed are written into the display data RAM (DD RAM), in the form of 8 bit character codes present in the character font table. The extended capacity of the DD RAM is 80 x 8 bits i.e. 80 characters.

register of the controller. All module timings are referenced to specific edges of the 'E' signal. The 'E' signal is applied only when a specific module transaction is desired.

The read and write strobes of the host, which provides the 'E' signals, should not be linked to the module's R/W line. An address bit which sets up earlier in the host's machine cycle can be used as R/W.

When the host processor is so fast that the strobes are too narrow to serve as the 'E' pulse

- a. Prolong these pulses by using the hosts 'Ready' input
- b. Prolong the host by adding wait states
- c. Decrease the Hosts Crystal frequency.

In spite of doing the above mentioned, if the problem continues, latch both the data and control information and then activate the 'E' signal

When the controller is performing an internal operation he busy flag (BF) will set and will not accept any instruction. The user should check the busy flag or should provide a delay of approximately 2ms after each instruction.

The module presents no difficulties while interfacing slower MPUs.

The liquid crystal display module can be interfaced, either to 4-bit or 8-bit MPUs.

For 4-bit data interface, the bus lines DB4 to DB7 are used for data transfer, while DB0 to DB3 lines are disabled. The data transfer is complete when the 4-bit data has been transferred twice. The busy flag must be checked after the 4-bit data has been transferred twice. Two more 4-bit operations then transfer the busy flag and address counter data.

For 8-bit data interface, all eight-bus lines (DB0 to DB7) are used.

## **5.6 THE 8255 PROGRAMMABLE PERIPHERAL INTERFACE**

The 8255 is a widely used, programmable, parallel I/O device. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O. It is flexible, versatile and economical (when multiple I/O ports are required), but somewhat complex. It is an important general-purpose I/O device that can be used with almost any microprocessor.

The 8255 has 24 I/O pins that can be grouped into two 8-bit parallel ports: A and B, with the remaining 8 bits as port C. The 8 bits of port C can be used as individual bits or be grouped in two 4 bit ports: C<sub>upper</sub> (Cu) and C<sub>lower</sub> (Cl). The functions of these ports are defined by writing a control register.

There are two modes of operation: the Bit Set/Reset (BSR) mode and the I/O mode. The BSR mode is used to set or reset the bits in port C. The

## **CHARACTER GENERATOR ROM (CG ROM)**

The character generator ROM generates 5 x 8 dot 5 x 10 dot character patterns from 8 bit character codes. It generates 208, 5 x 8 dot character patterns and 32, 5 x 10 dot character patterns.

## **CHARACTER GENERATOR RAM (CG RAM)**

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

### **5.5.10 INTERFACING THE MICROPROCESSOR / CONTROLLER:**

The module, interfaced to the system, can be treated as RAM input/output, expanded or parallel I/O. Since there is no conventional chip select signal, developing a strobe signal for the enable signal (E) and applying appropriate signals to the register select (RS) and read/write (R/W) signals are important.

The module is selected by gating a decoded module – address with the host – processor's read/write strobe. The resultant signal, applied to the LCDs enable (E) input, clocks in the data.

The 'E' signal must be a positive going digital strobe, which is active while data and control information are stable and true. The falling edge of the enable signal enables the data / instruction

I/O mode is further divided into three modes: Mode0 Mode1 and Mode2. In Mode0, all ports function as simple I/O ports. Mode1 is a handshake mode whereby ports A and/or B use bits from port C as handshake signals. In the handshaking mode, two types of I/O data transfer can be implemented: status check and interrupt. In Mode2, port A can be set up for bi-directional data transfer using handshake signals from port C, and port B can be set up either in Mode0 or Mode1.

### 5.6.1 CONTROL LOGIC:

$\overline{\text{RD}}$  (Read):

This control signal enables the read operation. When the signal is low the MPU reads data from a selected I.O port of the 8255.

$\overline{\text{WR}}$  (Write):

This control signal enables the write operation. When the signal goes low the MPU writes into a selected I/O port or the control register.

RESET (Reset):

This is an active high signal; it clears the control register and sets all ports in the input mode.

$\overline{\text{CS}}$ ,  $A_0$  and  $A_1$ :

These are device select signals.  $\overline{\text{CS}}$  is connected to a decoded address, and  $A_0$  and  $A_1$  are generally connected to MPU address lines  $A_0$  and  $A_1$ , respectively.

The  $\overline{\text{CS}}$  signal is the master Chip Select, and  $A_0$  and  $A_1$ , specify one of the I/O ports or the control register as given below:

$\overline{CS}$	$A_0$	$A_1$	Selected
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	X	X	8255 not selected

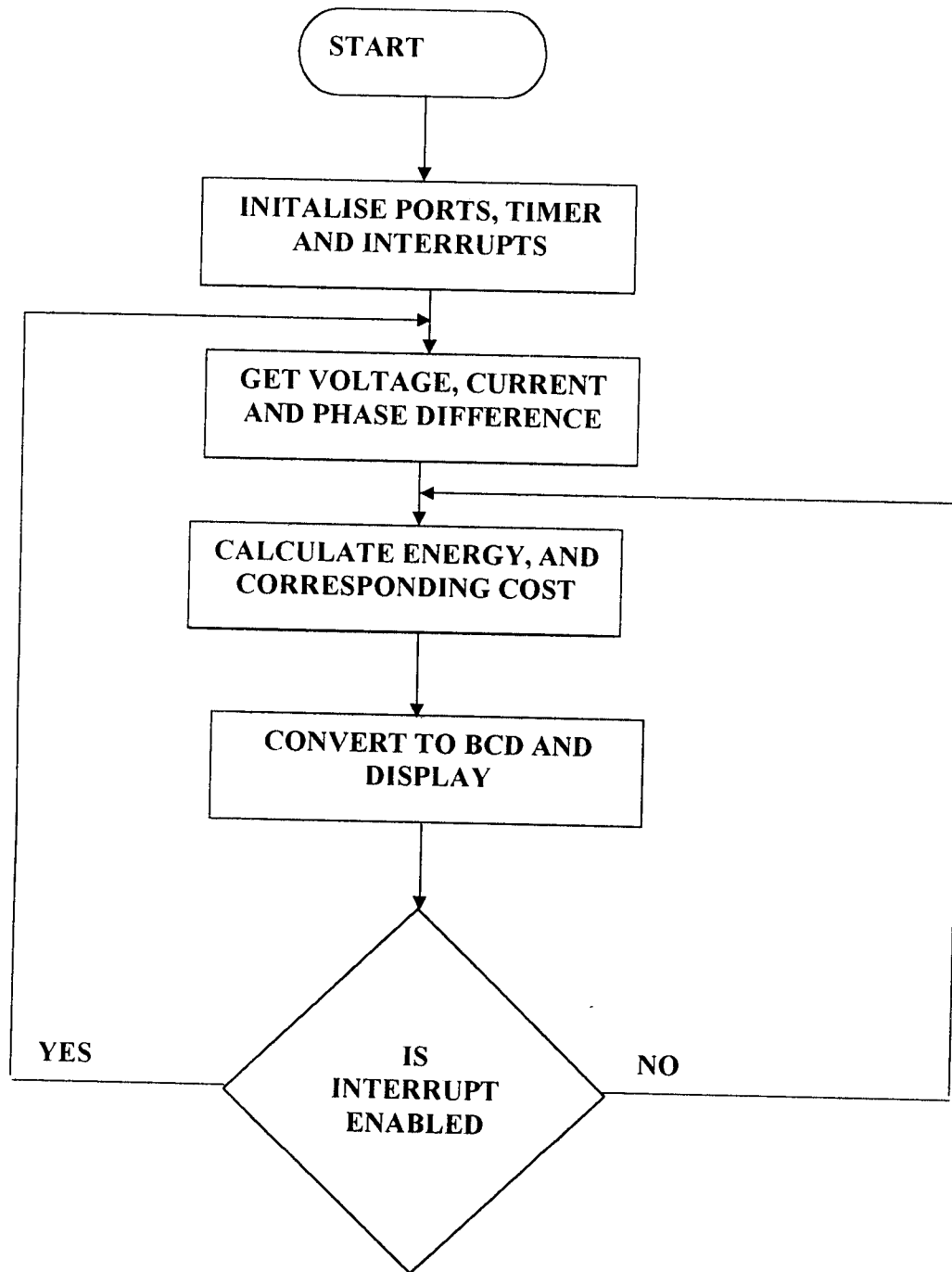


**SOFTWARE**

## **CHAPTER 6**

### **ALGORITHM**

- Initialise ports, timers.
- Get the voltage, current signals.
- Calculate phase difference.
- Calculate the power and Energy.
- Calculate the tariff.
- Display the Energy and Tariff.



```

                                org     0000h
                                ljmp    main

                                org     000bh
                                ljmp    timer0

                                org     001bh
                                ljmp    timer1

main:                            org     100h
                                mov     30h,#00h
                                mov     31h,#00h
                                mov     62h,#00h
                                mov     61h,#00h
                                mov     a,#088h
                                mov     r0,#01bh
                                movx   @r0,a
                                mov     a,#0ffh
                                mov     r0,#19h
                                movx   @r0,a
                                mov     a,#000h
                                mov     r0,#01ah
                                movx   @r0,a
                                lcall   delay
                                mov     r0,#018h
                                mov     a,#038h
                                movx   @r0,a
                                lcall   clock
                                lcall   del
                                mov     a,#008h
                                mov     r0,#018h
                                movx   @r0,a
                                lcall   clock
                                lcall   del
                                mov     a,#001h
                                mov     r0,#018h
                                movx   @r0,a
                                lcall   clock
                                lcall   del
                                mov     a,#006h
                                mov     r0,#018h
                                movx   @r0,a
                                lcall   clock
                                lcall   del
                                mov     r0,#018h
                                mov     a,#00ch
                                movx   @r0,a
                                lcall   clock
                                lcall   del

```

```

        mov     a,#045h
        movx   @r0,a
        lcall  clock
        lcall  del
in_dis:  mov     a,#080h           ; display blank
        mov     r0,#18h
        movx   @r0,a
        lcall  clock
        mov     r6,#"E"
        lcall  dr_w
        mov     r6,#"N"
        lcall  dr_w
        mov     r6,#"E"
        lcall  dr_w
        mov     r6,#"R"
        lcall  dr_w
        mov     r6,#"G"
        lcall  dr_w
        mov     r6,#"Y"
        lcall  dr_w
        mov     r6,#" "
        lcall  dr_w
        mov     r6,#"M"
        lcall  dr_w
        mov     r6,#"E"
        lcall  dr_w
        mov     r6,#"T"
        lcall  dr_w
        mov     r6,#"E"
        lcall  dr_w
        mov     r6,#"R"
        lcall  dr_w
        mov     r6,#" "
        lcall  dr_w
        mov     r6,#" "
        lcall  dr_w
        mov     r6,#" "
        lcall  dr_w
        mov     r6,#" "
        lcall  dr_w
        mov     a,#0c0h           ; display blank
        mov     r0,#18h
        movx   @r0,a
        lcall  clock
        mov     r6,#" "
        lcall  dr_w
        mov     r6,#" "
        lcall  dr_w
        mov     r6,#" "
        lcall  dr_w
        mov     r6,#" "
        lcall  dr_w

```

```

        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        mov     r6, #" "
        lcall  dr_w
        lcall  adc_in0
        lcall  adc_in1
        lcall  mul
        lcall  hex_dec
        lcall  hex_dec1
        lcall  hex_dec2
        lcall  change
        lcall  d_hex
        lcall  hex_dec2
        mov     3bh, #00h
        mov     3ch, #00h
        mov     3dh, #00h
        mov     3eh, #00h
        mov     8ch, #0bh        ; th0(timer\counter0
high byte)
        mov     8ah, #0dbh        ; t10(  ,,  ,,
low byte)
        setb   0afh        ; enable the interrui
pt source
        setb   0a9h        ; enable timer0 over
flow interrupt
        mov     89h, #11h        ; tmod(timer\counter
0 mode control reg)
        mov     8dh, #0bh
        mov     8bh, #0dbh
        setb   0afh
        setb   0abh        ; enable the interrui
pt source

```

```

                setb    8ch                ; to turn timer\coun
ter0 on
loop:           sjmp    loop

timer0:        push    psw
                push    acc
                inc     3bh
                mov     a,3bh
                subb   a,#0b0h
                jc     jreti
                inc     3ch
                mov     a,3ch
                subb   a,#0ffh
                jc     jreti
                lcall   d_hex
                lcall   compare
                mov     3bh,#00h
                mov     3ch,#00h
jreti:         lcall   t_load
                pop     psw
                pop     acc
                reti

t_load:        mov     8dh,#0bh
                mov     8bh,#0dbh
                ret

timer1:        push    psw
                push    acc
                inc     3dh
                mov     a,3dh
                subb   a,#0b0h
                jc     jretz
                inc     3eh
                mov     a,3eh
                subb   a,#0ffh
                jc     jretz
                lcall   d_hex
                lcall   compare1
                mov     3dh,#00h
                mov     3eh,#00h
jretz:         lcall   t_loadd
                pop     psw
                pop     acc
                reti

t_loadd:       mov     8dh,#0bh
                mov     8bh,#0dbh
                ret

```

```

adc_in0:    mov     dptr,#08h
            movx   a,@dptr
            lcall  del
            mov     dptr,#10h
            movx   a,@dptr
            mov     40h,a
            ret

adc_in1:    mov     dptr,#09h
            movx   a,@dptr
            lcall  del
            mov     dptr,#10h
            movx   a,@dptr
            mov     41h,a
            ret

hex_dec:    mov     a,40h
            mov     b,#64h
            div    ab
            mov     55h,a
            mov     a,b
            mov     b,#0ah
            div    ab
            mov     54h,a
            mov     53h,b
            ret

hex_dec1:   mov     a,41h
            mov     b,#64h
            div    ab
            mov     52h,a
            mov     a,b
            mov     b,#0ah
            div    ab
            mov     51h,a
            mov     50h,b
            ret

hex_dec2:   mov     r7,60h
            mov     r6,61h
            mov     a,r6           ;r6 msb
            mov     20h,#00h
            mov     r2,#00h
            mov     r3,#00h
            mov     r4,#00h

loop_m2:    cjne   a,#03h,loop_m1
            mov     a,r7
            subb   a,#0e8h
            jc     loop_h

```



```

                                inc     20h           ;1000's count
                                mov     r7,a
                                mov     a,r6
                                subb   a,#03h
                                mov     r6,a
                                ljmp   loop_h
loop_m1:                         jc     loop_h
                                inc     20h           ;1000's count
                                mov     a,#0e8h
                                mov     r5,a
                                mov     a,r7
                                clr     c
                                subb   a,r5
                                mov     r7,a
                                mov     a,#03h
                                addc   a,#00h
                                mov     r5,a
                                mov     a,r6
                                subb   a,r5
                                mov     r6,a
                                ljmp   loop_m2
loop_h:                          mov     a,r6
                                cjne   a,#00h,loop_y
                                ljmp   loop_x
loop_y:                          clr     c
                                mov     a,r7
                                inc     r2
                                subb   a,#64h
                                mov     r7,a
                                jnc    loop_y
                                mov     a,r6
                                dec     a
                                mov     r6,a
                                ljmp   loop_h
loop_x:                          mov     a,r7
                                cjne   a,#64h,loop_m
                                inc     r2
                                mov     a,r7
                                subb   a,#64h
                                ljmp   loop_d1
loop_m:                          jc     loop_111
                                inc     r2           ;100's count
                                subb   a,#64h
                                mov     r7,a
                                ljmp   loop_x
loop_111:                       cjne   a,#0ah,loop_d10
                                subb   a,#0ah
                                mov     r7,a
                                inc     r3           ;10's count
                                mov     a,#00h
                                ljmp   loop_d1
loop_d10:                       jc     loop_d1

```

```

        inc     r3
        mov     a,r7
        subb   a,#0ah
        mov     r7,a
        ljmp   loop_111
loop_d1:  mov     r4,a           ;ls
        ret

change:   mov     70h,20h
        mov     71h,r2
        mov     72h,r3

        ret

mul:      mov     a,40h
        mov     b,41h
        mul    ab
        mov     60h,a
        mov     61h,b
        ret

d_hex:    mov     a,#00h
        mov     r3,a
        mov     a,70h
        mov     r4,a
        mov     a,71h
        mov     r5,a
        mov     a,72h
        mov     r6,a

        mov     a,r3
        mov     b,a
        mov     a,#0e8h
        mul    ab
        mov     25h,a
        mov     a,b
        mov     26h,a           ;msb of thouse

nd 0e 2541

        mov     a,r3
        mov     b,a
        mov     a,#03h
        mul    ab
        mov     27h,a           ;lsb of thouse

nd 03 2542

        mov     b,a
        mov     a,26h

```

```

42          add      a,b          ;add 2541+25
           mov      26h,a          ;adde
d lsb in 2541
           mov      a,r4
           mov      b,a
           mov      a,#64h
           mul      ab
           moy      28h,a
           mov      a,b
           mov      29h,a          ;msb in 2544
           mov      a,r5
           mov      b,a
           mov      a,#0Ah
           mul      ab
           mov      2ah,a
           mov      a,r6
           mov      2bh,a
           mov      a,2bh
           mov      b,a
           mov      a,2ah
           add      a,b
           mov      2ch,a
           mov      b,a
           clr      c
           mov      a,28h
           add      a,b
           mov      2dh,a
           mov      a,29h
           addc     a,#00h          ;ADD(CARRY+1
00SMSB)
           mov      2ch,a          ;2545 MSB FIN_
HUN
           mov      a,2dh
           mov      b,a
           clr      c
           mov      a,25h
           add      a,b          ;ADD LSB(FI
N_HUN+THOU_E_LSB)
           mov      2fh,a          ;FINAL HEX SET
POINT LSB * 2fh
           mov      a,2ch
           mov      b,a
           mov      a,26h
           addc     a,b          ;MSB (FIN_HU
N+FIN_THOU) *3fh
           mov      3fh,a
           mov      61h,3fh
           mov      60h,2fh
           ret

```

compare:

```

mov      a,2fh
cjne    a,#00h,nexxx
mov      30h,#00h
mov      31h,#00h
lcall   hex_decc
lcall   displaya
ret
nexxx:  cjne    a,#2ah,extt
mov      a,#06h
add     a,30h
mov     30h,a
lcall   hex_decc
lcall   display
mov     a,30h
cjne    a,#0fch,jjtii
mov     30h,#02h
inc     31h
lcall   hex_decc
lcall   display
mov     a,31h
cjne    a,#27h,jjtii
lcall   dis_unit
mov     6fh,#25h
zzzz:  lcall   del
djnz    6fh,zzzz
mov     31h,#00h
mov     30h,#00h
clr     8ch
setb    8eh
jjtii: ret
extt:  jnc     nexx
mov     a,#06h
add     a,30h
mov     30h,a
lcall   hex_decc
lcall   display
mov     a,30h
cjne    a,#0fch,jrttt
mov     30h,#02h
inc     31h
lcall   hex_decc
lcall   display
mov     a,31h
cjne    a,#27h,jrttt
lcall   dis_unit
mov     6fh,#25h
zzzz1: lcall   del
djnz    6fh,zzzz1
mov     31h,#00h
mov     30h,#00h
clr     8ch

```

```

                                setb    8eh

jrttt:    ret

nexx:     cjne    a,#41h,next
          mov     a,#0ah
          add     a,30h
          mov     30h,a
          lcall   hex_decc
          lcall   display
          mov     a,30h
          cjne   a,#0fah,jjeti
          mov     30h,#04h
          inc     31h
          lcall   hex_decc
          lcall   display
          mov     a,31h
          cjne   a,#27h,jjeti
          lcall   dis_unit
          lcall   del
          lcall   del
          lcall   del
          mov     6fh,#25h
zzz2:     lcall   del
          djnz   6fh,zzz2
          mov     31h,#00h
          mov     30h,#00h
          clr     8ch
          setb   8eh

jjeti:    ret

next:     jnc     next2
          mov     a,#0ah
          add     a,30h
          mov     30h,a
          lcall   hex_decc
          lcall   display
          mov     a,30h
          cjne   a,#0fah,jrett
          mov     30h,#04h
          inc     31h
          lcall   hex_decc
          lcall   display
          mov     a,31h
          cjne   a,#27h,jrett
          lcall   dis_unit
          mov     6fh,#25h
zzz3:     lcall   del
          djnz   6fh,zzz3
          mov     31h,#00h
          mov     30h,#00h

```

```

        clr      8ch
        setb    8eh

jrett:   ret

next2:   cjne    a, #69h, nextt1
         mov     a, #10h
         add    a, 30h
         mov    30h, a
         lcall  hex_decc
         lcall  display
         mov    a, 30h
         cjne  a, #0f0h, rttt
         mov    30h, #00h
         inc   31h
         lcall  hex_decc
         lcall  display
         mov    a, 31h
         cjne  a, #27h, rttt
         lcall  dis_unit
         mov    6fh, #25h
zzz4:    lcall  del
         djnz   6fh, zzz4
         mov    31h, #00h
         mov    30h, #00h
         clr   8ch
         setb  8eh

rttt:    ret

nextt1:  jnc     next3
         mov    a, #10h
         add    a, 30h
         mov    30h, a
         lcall  hex_decc
         lcall  display
         mov    a, 30h
         cjne  a, #0f0h, jjrett
         mov    30h, #00h
         inc   31h
         lcall  hex_decc
         lcall  display
         mov    a, 31h
         cjne  a, #27h, jjrett
         lcall  dis_unit
         mov    6fh, #25h
zzz5:    lcall  del
         djnz   6fh, zzz5
         mov    31h, #00h
         mov    30h, #00h

```

```

                clr      8ch
                setb    8eh

jjrett:        ret

next3:         cjne    a,#0ffh,nextt2
                mov     a,#20h
                add     a,30h
                mov     30h,a
                lcall   hex_decc
                lcall   display
                mov     a,30h
                cjne    a,#0e0h,jjjti
                mov     30h,#00h
                inc     31h
                lcall   hex_decc
                lcall   display
                mov     a,31h
                cjne    a,#27h,jjjti
                lcall   dis_unit
                mov     6fh,#25h
zzz6:         lcall   del
                djnz    6fh,zzz6
                mov     31h,#00h
                mov     30h,#00h
                clr     8ch
                setb    8eh

jjjti:        ret

nextt2:       jnc     nextt3
                mov     a,#20h
                add     a,30h
                mov     30h,a
                lcall   hex_decc
                lcall   display
                mov     a,30h
                cjne    a,#0e0h,jjjtt
                mov     30h,#00h
                inc     31h
                lcall   hex_decc
                lcall   display
                mov     a,31h
                cjne    a,#27h,jjjtt
                lcall   dis_unit
                mov     6fh,#25h
zzz7:         lcall   del
                djnz    6fh,zzz7
                mov     31h,#00h
                mov     30h,#00h
                clr     8ch
                setb    8eh

```





```

delay:
mov    7ah,#010h
jmp:   mov    7bh,#010h
loopd: djnz   7bh,loopd
       djnz   7ah,imp
       ret

del:   mov    7ch,#0fTh
loopd1: mov   7dh,#0fTh
loopd2: djnz   7dh,loopd2
       djnz   7ch,loopd1
       ret

dr_w:  mov    r0,#18h
       mov    a,r6
       movx  @r0,a
       lcall clock1
       ret

clock: mov    r0,#1ah
       mov    a,#04h
       movx  @r0,a
       lcall delay
       mov    a,#00h
       movx  @r0,a
       lcall delay
       mov    a,#01h
       movx  @r0,a
       lcall delay
       mov    a,#05h
       movx  @r0,a
       lcall delay
       mov    r0,#1ah
       ret

clock1: mov    r0,#1ah
       mov    a,#05h
       movx  @r0,a
       lcall delay
       mov    a,#01h
       movx  @r0,a
       lcall delay
       mov    r0,#18h
       movx  @r0,a
       lcall delay
       ret

display: mov    a,#080h
       mov    r0,#18h
       movx  @r0,a
       lcall clock
       mov    r6,#"U"
       lcall display
       mov    r6,#"N"
       lcall display
       mov    r6,#"I"
       lcall display
       mov    r6,#"L"
       lcall display
       ret
: display blank

```

```
ret
jjeztz:
jnc ext:
mov a,#0ah
add a,30h
mov 30h,a
lcall hex_decc
lcall display1
mov a,30h
cjne a,#0fah,jrettz
mov 30h,#04h
inc 31h
lcall hex_decc
lcall display1
mov a,31h
cjne a,#27h,jrettz
lcall dis_unit2
mov 6fh,#25h
lcall del
djnz 6fh,kzz3
mov 31h,#00h
```

```
retttz:
cjne a,#41h,ext
mov a,#0ah
add a,30h
mov 30h,a
lcall hex_decc
lcall display1
mov a,30h
cjne a,#0fah,jjeztz
mov 30h,#04h
inc 31h
lcall hex_decc
lcall display1
mov a,31h
cjne a,#27h,jjeztz
lcall dis_unit2
lcall del
lcall del
lcall del
lcall del
mov 6fh,#25h
lcall del
djnz 6fh,zzk2
mov 31h,#00h
mov 30h,#00h
```

```
mov 31h,#00h
mov 30h,#00h
```

```

jjjtt:      ret
nextt3:     ret

compare1:
    mov     a,2fh
    cjne   a,#00h,exxx
    mov     30h,#00h
    mov     31h,#00h
    lcall  hex_decc
    lcall  display1
    ret
exxx:      cjne   a,#2ah,xtt
    mov     a,#06h
    add    a,30h
    mov     30h,a
    lcall  hex_decc
    lcall  display1
    mov     a,30h
    cjne   a,#0fch,jjtiiz
    mov     30h,#02h
    inc    31h
    lcall  hex_decc
    lcall  display1
    mov     a,31h
    cjne   a,#27h,jjtiiz
    lcall  dis_unit2
    mov     6fh,#25h
fzzz:     lcall  del
    djnz   6fh,fzzz
    mov     31h,#00h
    mov     30h,#00h

jjtiiz:    ret

xtt:      jnc     exx
    mov     a,#06h
    add    a,30h
    mov     30h,a
    lcall  hex_decc
    lcall  display1
    mov     a,30h
    cjne   a,#0fch,jrtttz
    mov     30h,#02h
    inc    31h
    lcall  hex_decc
    lcall  display1
    mov     a,31h
    cjne   a,#27h,jrtttz
    lcall  dis_unit2
    mov     6fh,#25h
zzzk1:   lcall  del
    djnz   6fh,zzzk1

```

```

                                mov     30h,#00h

jrettz:      ret

ext2:       cjne     a,#69h,extt1
            mov     a,#10h
            add     a,30h
            mov     30h,a
            lcall   hex_decc
            lcall   display1
            mov     a,30h
            cjne   a,#0f0h,rtttz
            mov     30h,#00h
            inc     31h
            lcall   hex_decc
            lcall   display1
            mov     a,31h
            cjne   a,#27h,rtttz
            lcall   dis_unit2
            mov     6fh,#25h
kzz4:       lcall   del
            djnz   6fh,kzz4
            mov     31h,#00h
            mov     30h,#00h

rtttz:      ret

extt1:      jnc     ext3
            mov     a,#10h
            add     a,30h
            mov     30h,a
            lcall   hex_decc
            lcall   display1
            mov     a,30h
            cjne   a,#0f0h,jjrettz
            mov     30h,#00h
            inc     31h
            lcall   hex_decc
            lcall   display1
            mov     a,31h
            cjne   a,#27h,jjrettz
            lcall   dis_unit2
            mov     6fh,#25h
kzz5:       lcall   del
            djnz   6fh,kzz5
            mov     31h,#00h
            mov     30h,#00h

```

```

lcall    dr_w
mov      r6, #"S"
lcall    dr_w
mov      r6, #":"
lcall    dr_w
mov      r6, #"0"
lcall    dr_w
mov      r6, #"0"
lcall    dr_w
mov      r6, #"0"
lcall    dr_w
mov      r6, #"("
lcall    dr_w
mov      r6, #"."
lcall    dr_w
mov      a, 20h
add     a, #30h
mov     r6, a
lcall    dr_w
mov     a, r2
add     a, #30h
mov     r6, a
lcall    dr_w
mov     a, r3
add     a, #30h
mov     r6, a
lcall    dr_w
mov     a, 21h
add     a, #30h
mov     r6, a
lcall    dr_w
mov     r6, #")"
lcall    dr_w
mov     a, #0c0h
mov     r0, #18h
movx    @r0, a
lcall    clock
mov     r6, #"C"
lcall    dr_w
mov     r6, #"O"
lcall    dr_w
mov     r6, #"S"
lcall    dr_w
mov     r6, #"T"
lcall    dr_w
mov     r6, #" "
lcall    dr_w
mov     r6, #":"
lcall    dr_w
mov     r6, #" "
lcall    dr_w
mov     r6, #" "

```

```

; display blank

```

```

        lcall    dr_w
        mov     r6, #" "
        lcall    dr_w
        mov     r6, #" "
        lcall    dr_w
        mov     r6, #"0"
        lcall    dr_w
        mov     r6, #"5"
        lcall    dr_w
        mov     r6, #"0"
        lcall    dr_w
        mov     r6, #"."
        lcall    dr_w
        mov     r6, #"0"
        lcall    dr_w
        mov     r6, #"0"
        lcall    dr_w
        ret
displaya: mov     a, #080h                ; display blank
        mov     r0, #18h
        movx   @r0, a
        lcall    clock
        mov     r6, #"U"
        lcall    dr_w
        mov     r6, #"N"
        lcall    dr_w
        mov     r6, #"I"
        lcall    dr_w
        mov     r6, #"T"
        lcall    dr_w
        mov     r6, #"S"
        lcall    dr_w
        mov     r6, #":"
        lcall    dr_w
        mov     r6, #"0"
        lcall    dr_w
        mov     r6, #"0"
        lcall    dr_w
        mov     r6, #"0"
        lcall    dr_w
        mov     r6, #"("
        lcall    dr_w
        mov     r6, #"."
        lcall    dr_w
        mov     a, 20h
        add    a, #30h
        mov     r6, a
        lcall    dr_w
        mov     a, r2
        add    a, #30h
        mov     r6, a
        lcall    dr_w

```

```

mov     r6, #"U"
lcall  dr_w
mov     r6, #"N"
lcall  dr_w
mov     r6, #"I"
lcall  dr_w
mov     r6, #"T"
lcall  dr_w
mov     r6, #"S"
lcall  dr_w
mov     r6, #":"
lcall  dr_w
mov     r6, #"0"
lcall  dr_w
mov     r6, #"0"
lcall  dr_w
mov     r6, #"1"
lcall  dr_w
mov     r6, #"("
lcall  dr_w
mov     r6, #"."
lcall  dr_w
mov     a, 20h
add     a, #30h
mov     r6, a
lcall  dr_w
mov     a, r2
add     a, #30h
mov     r6, a
lcall  dr_w
mov     a, r3
add     a, #30h
mov     r6, a
lcall  dr_w
mov     a, 21h
add     a, #30h
mov     r6, a
lcall  dr_w
mov     r6, #")"
lcall  dr_w
mov     a, #0c0h           ; display blank
mov     r0, #18h
movx   @r0, a
lcall  clock
mov     r6, #"C"
lcall  dr_w
mov     r6, #"O"
lcall  dr_w
mov     r6, #"S"
lcall  dr_w
mov     r6, #"T"
lcall  dr_w

```

```

mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#"1"
lcall  dr_w
mov     a,#0c0h           ; display blank
mov     r0,#18h
movx   @r0,a
lcall  clock
mov     r6,#"C"
lcall  dr_w
mov     r6,#"O"
lcall  dr_w
mov     r6,#"S"
lcall  dr_w
mov     r6,#"T"
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#":"
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#"5"
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     r6,#"."
lcall  dr_w
mov     r6,#"6"
lcall  dr_w
mov     r6,#"5"
lcall  dr_w
ret

```

```

dis_unit2:  mov     a,#080h           ; display blank
            mov     r0,#18h

```



```

mov     r6,#" "
lcall  dr_w
mov     r6,#": "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     r6,#"5"
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     r6,#"."
lcall  dr_w
mov     r6,#"6"
lcall  dr_w
mov     r6,#"5"
lcall  dr_w
ret

```

```

dis_unit:  mov     a,#080h           ; display blank
            mov     r0,#18h
            movx    @r0,a
            lcall  clock
            mov     r6,#"U"
            lcall  dr_w
            mov     r6,#"N"
            lcall  dr_w
            mov     r6,#"I"
            lcall  dr_w
            mov     r6,#"T"
            lcall  dr_w
            mov     r6,#"S"
            lcall  dr_w
            mov     r6,#": "
            lcall  dr_w
            mov     r6,#" "
            lcall  dr_w
            mov     r6,#" "
            lcall  dr_w
            mov     r6,#" "
            lcall  dr_w
            mov     r6,#" "
            lcall  dr_w

```

```

mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#"5"
lcall  dr_w
mov     r6,#"1"
lcall  dr_w
mov     r6,#"."
lcall  dr_w
mov     r6,#"3"
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
ret

```

```

hex_decc:      mov     r7,30h
               mov     r6,31h
               mov     a,r6           ;r6 msb
               mov     20h,#00h
               mov     r2,#00h
               mov     r3,#00h
               mov     r4,#00h
loop_m2:      cjne   a,#03h,loop_m1
               mov     a,r7
               subb   a,#0e8h
               jc     loop_h
               inc    20h           ;1000's count
               mov    r7,a
               mov    a,r6
               subb   a,#03h
               mov    r6,a
               ljmp  loop_h
loop_m1:      jc     loop_h
               inc    20h           ;1000's count
               mov    a,#0e8h
               mov    r5,a
               mov    a,r7
               clr    c
               subb   a,r5
               mov    r7,a
               mov    a,#03h
               addc   a,#00h
               mov    r5,a
               mov    a,r6

```

```

                                subb    a,r5
                                mov     r6,a
lloop_h:                        ljmp    lloop_m2
                                mov     a,r6
                                cjne   a,#00h,lloop_y
lloop_y:                        ljmp    lloop_x
                                clr     c
                                mov     a,r7
                                inc     r2
                                subb   a,#64h
                                mov     r7,a
                                jnc    lloop_y
                                mov     a,r6
                                dec     a
                                mov     r6,a
lloop_x:                        ljmp    lloop_h
                                mov     a,r7
                                cjne   a,#64h,lloop_m
                                inc     r2
                                mov     a,r7
                                subb   a,#64h
lloop_m:                        ljmp    lloop_d1
                                jc     lloop_111
                                inc     r2                ;100's count
                                subb   a,#64h
                                mov     r7,a
lloop_111:                      ljmp    lloop_x
                                cjne   a,#0ah,lloop_d10
                                subb   a,#0ah
                                mov     r7,a
                                inc     r3                ;10's count
                                mov     a,#00h
lloop_d10:                      ljmp    lloop_d1
                                jc     lloop_d1
                                inc     r3
                                mov     a,r7
                                subb   a,#0ah
                                mov     r7,a
lloop_d1:                        ljmp    lloop_111
                                mov     21h,a            ;1s
                                ret

```

dis\_set:

```

mov     a,#080h                ; display blank
mov     r0,#18h
movx   @r0,a
lcall  clock
mov     r6,#"v"
lcall  dr_w
mov     r6,#"o"

```

```

movx    @r0,a
lcall   clock
mov     r6,#"U"
lcall   dr_w
mov     r6,#"N"
lcall   dr_w
mov     r6,#"I"
lcall   dr_w
mov     r6,#"T"
lcall   dr_w
mov     r6,#"S"
lcall   dr_w
mov     r6,#":"
lcall   dr_w
mov     r6,#" "
lcall   dr_w
mov     r6,#" "
lcall   dr_w
mov     r6,#" "
lcall   dr_w
mov     r6,#" "
lcall   dr_w
mov     r6,#" "
lcall   dr_w
mov     r6,#" "
lcall   dr_w
mov     r6,#" "
lcall   dr_w
mov     r6,#" "
lcall   dr_w
mov     r6,#"2"
lcall   dr_w
mov     a,#0c0h           ; display blank
mov     r0,#18h
movx    @r0,a
lcall   clock
mov     r6,#"C"
lcall   dr_w
mov     r6,#"O"
lcall   dr_w
mov     r6,#"S"
lcall   dr_w
mov     r6,#"T"
lcall   dr_w
mov     r6,#" "
lcall   dr_w
mov     r6,#":"
lcall   dr_w
mov     r6,#" "
lcall   dr_w

```

```

mov     a, r3
add     a, #30h
mov     r6, a
lcall  dr_w
mov     a, 21h
add     a, #30h
mov     r6, a
lcall  dr_w
mov     r6, #")"
lcall  dr_w
mov     a, #0c0h           ; display blank
mov     r0, #18h
movx    @r0, a
lcall  clock
mov     r6, #"C"
lcall  dr_w
mov     r6, #"0"
lcall  dr_w
mov     r6, #"S"
lcall  dr_w
mov     r6, #"T"
lcall  dr_w
mov     r6, #" "
lcall  dr_w
mov     r6, #":"
lcall  dr_w
mov     r6, #" "
lcall  dr_w
mov     r6, #" "
lcall  dr_w
mov     r6, #" "
lcall  dr_w
mov     r6, #"0"
lcall  dr_w
mov     r6, #"5"
lcall  dr_w
mov     r6, #"0"
lcall  dr_w
mov     r6, #"."
lcall  dr_w
mov     r6, #"0"
lcall  dr_w
mov     r6, #"0"
lcall  dr_w
ret

```

```

display1:  mov     a, #080h           ; display blank
           mov     r0, #18h
           movx    @r0, a
           lcall  clock

```

```

lcall    dr_w
mov      r6, #"l"
lcall    dr_w
mov      r6, #"t"
lcall    dr_w
mov      r6, #" "
lcall    dr_w
mov      a, 52h
add      a, #30h
mov      r6, a
lcall    dr_w
mov      a, 51h
add      a, #30h
mov      r6, a
lcall    dr_w
mov      a, 50h
add      a, #30h
mov      r6, a
lcall    dr_w
mov      r6, #"c"
lcall    dr_w
mov      r6, #"u"
lcall    dr_w
mov      r6, #"r"
lcall    dr_w
mov      r6, #" "
lcall    dr_w
mov      a, 55h
add      a, #30h
mov      r6, a
lcall    dr_w
mov      a, 54h
add      a, #30h
mov      r6, a
lcall    dr_w
mov      a, 53h
add      a, #30h
mov      r6, a
lcall    dr_w
mov      r6, #" "
lcall    dr_w
mov      a, #0c0h           ; display blank
mov      r0, #18h
movx     @r0, a
lcall    clock
mov      r6, #" "
lcall    dr_w
mov      r6, #"p"
lcall    dr_w
mov      r6, #"o"
lcall    dr_w
mov      r6, #"w"

```

```

lcall    dr_w
mov      r6, #"e"
lcall    dr_w
mov      r6, #"r"
lcall    dr_w
mov      r6, #" "
lcall    dr_w
mov      r6, #" "
lcall    dr_w
mov      a, 20h
add      a, #30h
mov      r6, a
lcall    dr_w
mov      a, r2
add      a, #30h
mov      r6, a
lcall    dr_w
mov      a, r3
add      a, #30h
mov      r6, a
lcall    dr_w
mov      a, r4
add      a, #30h
mov      r6, a
lcall    dr_w
mov      r6, #" "
lcall    dr_w
mov      r6, #" "
lcall    dr_w
mov      r6, #" "
lcall    dr_w
mov      r6, #" "
lcall    dr_w
ret

```

dis\_set1:

```

mov      a, #080h           ; display blank
mov      r0, #18h
movx    @r0, a
lcall    clock
mov      r6, #"E"
lcall    dr_w
mov      r6, #"N"
lcall    dr_w
mov      r6, #"E"
lcall    dr_w
mov      r6, #"R"
lcall    dr_w
mov      r6, #"G"
lcall    dr_w
mov      r6, #"Y"
lcall    dr_w

```

```

mov        r6, #0"
lcall     dr_w
ret

dis_set2:

mov        a, #080h           ; display blank
mov        r0, #18h
movx      @r0, a
lcall     clock
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        r6, # " "
lcall     dr_w
mov        a, #0c0h           ; display blank
mov        r0, #18h
movx      @r0, a
lcall     clock
mov        r6, # " "
lcall     dr_w
mov        r6, # "c"
lcall     dr_w
mov        r6, # "o"
lcall     dr_w
mov        r6, # "s"
lcall     dr_w
mov        r6, # "t"

```



```

    lcall    dr_w
    mov     r6,#" "
    lcall    dr_w
    mov     r6,#" "
    lcall    dr_w
    mov     a,20h
    add     a,#30h
    mov     r6,a
    lcall    dr_w
    mov     a,r2
    add     a,#30h
    mov     r6,a
    lcall    dr_w
    mov     r6,#"."
    lcall    dr_w
    mov     a,r3
    add     a,#30h
    mov     r6,a
    lcall    dr_w
    mov     a,r4
    add     a,#30h
    mov     r6,a
    lcall    dr_w
    mov     r6,#" "
    lcall    dr_w
    mov     r6,#" "
    lcall    dr_w
    mov     r6,#" "
    lcall    dr_w
    mov     r6,#" "
    lcall    dr_w
    ret

```

dis\_load:

```

    mov     a,#080h           ; display blank
    mov     r0,#18h
    movx    @r0,a
    lcall    clock
    mov     r6,#"0"
    lcall    dr_w
    mov     r6,#"v"
    lcall    dr_w
    mov     r6,#"e"
    lcall    dr_w
    mov     r6,#"r"
    lcall    dr_w
    mov     r6,#" "
    lcall    dr_w
    mov     r6,#" "
    lcall    dr_w
    mov     r6,#"L"
    lcall    dr_w

```

```

mov     r6,#":"
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     a,#0c0h           ; display blank
mov     r0,#18h
movx   @r0,a
lcall  clock
mov     r6,#"C"
lcall  dr_w
mov     r6,#"O"
lcall  dr_w
mov     r6,#"S"
lcall  dr_w
mov     r6,#"T"
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#":"
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#" "
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     r6,#"0"
lcall  dr_w
mov     r6,#"."
lcall  dr_w
mov     r6,#"0"
lcall  dr_w

```



## **TEST RESULTS**

## TEST RESULTS:

### ENERGY METER:

Microcontroller based energy-meter was implanted in the laboratory and tested for accuracy with a standard energy-meter, which was connected in cascade.

The results are shown in the following table:

Digital Energy (KWH)	Conventional Energy (KWH)	% Error
0.008	0.00875	-8.57
0.016	0.018	-11.11
0.021	0.023	-8.69
0.028	0.03	-6.66
0.034	0.036	-5.55

On comparing the results with the conventional meter, we infer that the microcontroller-based energy-meter has no appreciable error.

**CONCLUSION**

## CONCLUSION

We here by conclude that the **DWHMCCI** designed by will be useful for various applications as listed below:

1. It can be used to calculate energy utilization of a consumer over a certain period and its corresponding cost according to the tariff fixed in it.
2. It can be used to record the amount of energy consumed by industrial equipment over a short or long period.
3. It also shows us the actual cost of consumption bypassing the need for manual calculation.

## SCOPE FOR FUTURE DEVELOPMENT:

### 1. INTERFACING WITH PCs:

The 8951 micro controller based DWHMCCI can be interfaced with a digital computer using a RS-232. The information from the system goes to the PC through the RXT, TXD pins of the 8951 and the PC can store information like previous value and energy consumed per day or month or year.

### 2. OVER A TELEPHONE LINE:

Using a personal computer with a modem a user can duplex any program containing a new message, which is programmed into the 8951, embedded in the application over a commercial telephone line. When programming is complete, the application executes a new program, which displays a new message.

### 3. ATTACHING A PRINTER FOR BILLING:

The necessary I/O lines our present in a design to enable direct printing of the number of units, cost, the EB division, energy-meter number, etc. using a printer in a remote location like a billing center.

### 4. CUTTING THE LOADS AUTOMATICALLY:

Various luxury electrical appliances which consume high power can be cut off at pre-determined rates with the help of relays interfaced the micro controller.

## **BIBLIOGRAPHY**

1. ATMEL MICROCONTROLLER HANDBOOK
2. LAMPEX LCD MANUAL
3. A COURSE IN ELECTRICAL, ELECTRONICS,  
MEASUREMENTS AND INSTRUMENTATION.

By

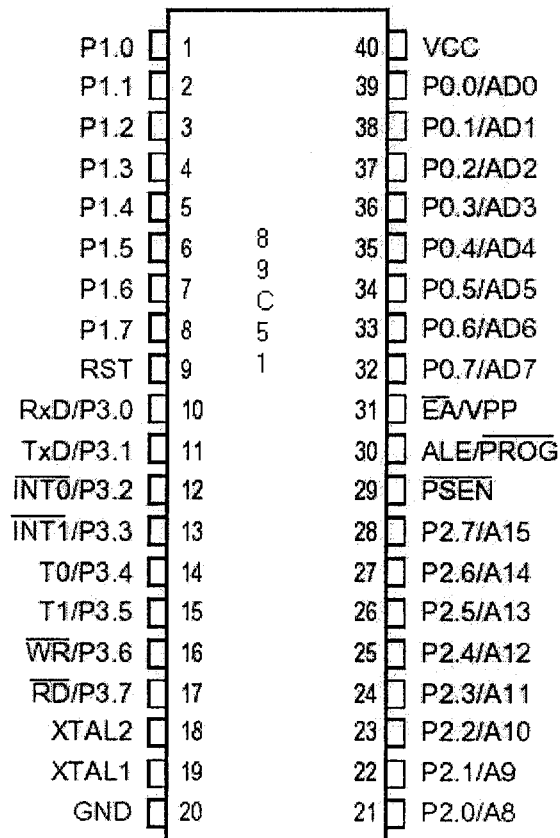
A.K.SHAWNY

4. MOTOROLA OPTO ELECTRONICS MANUAL.
5. PCB CIRCUIT DESIGN AND TECHNOLOGY.

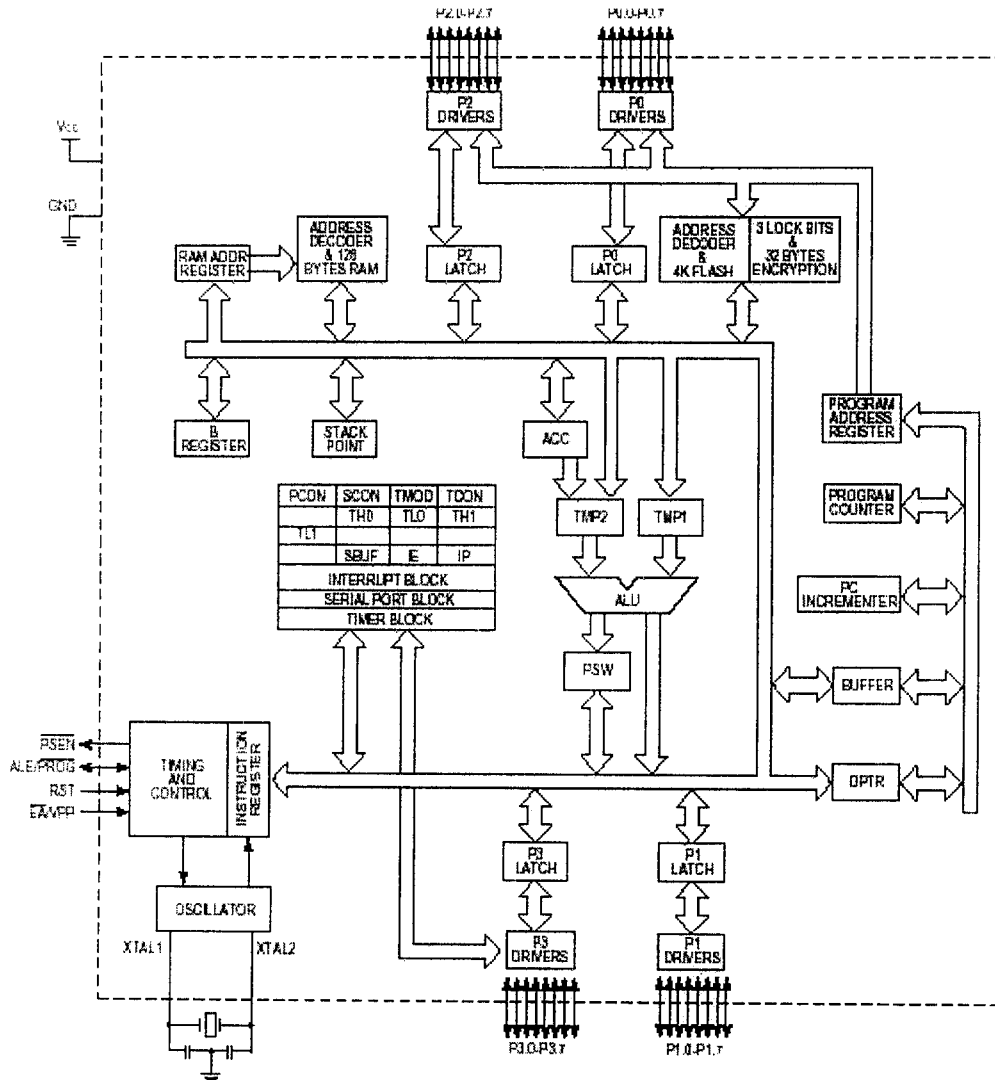


## **APPENDIX**

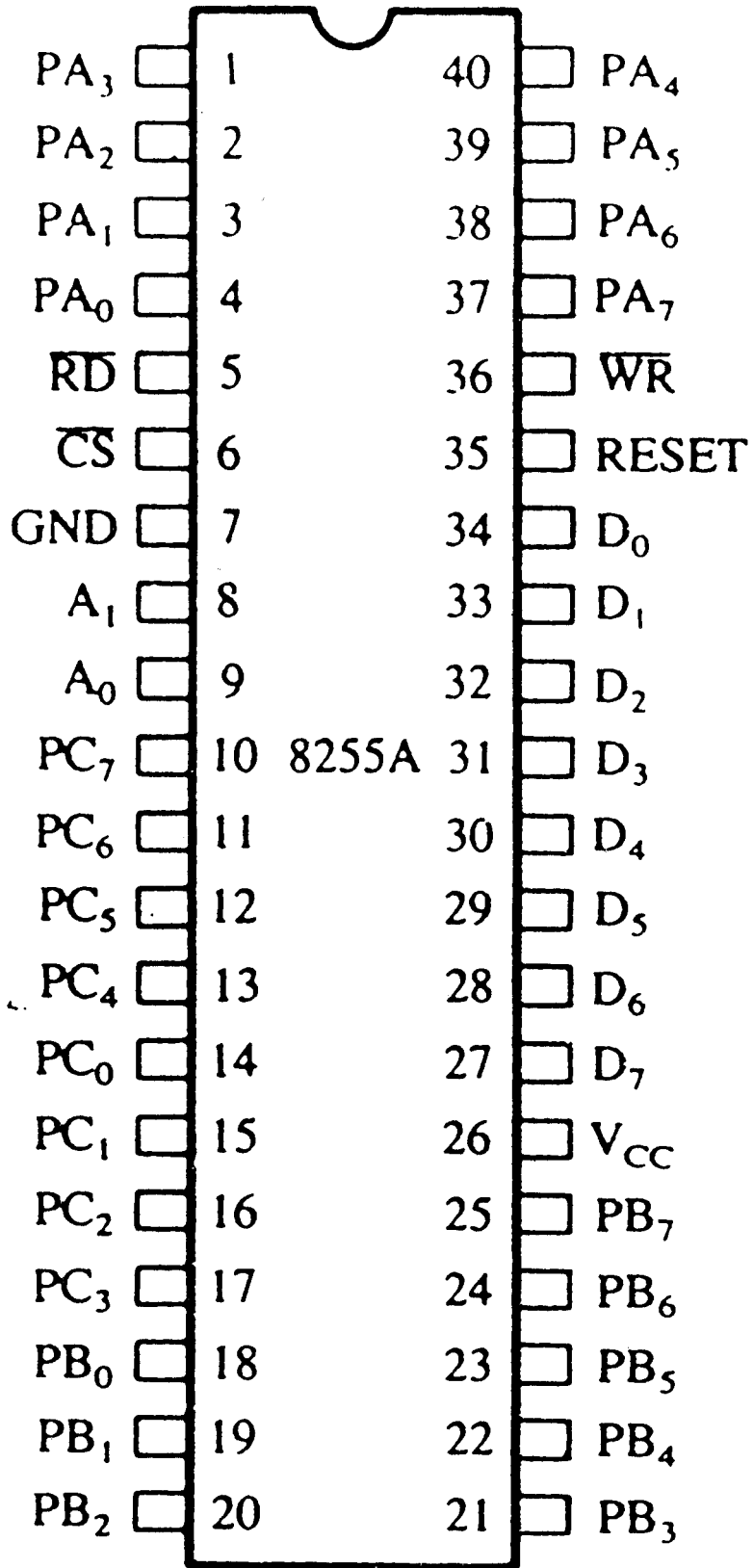
## PIN CONFIGURATION

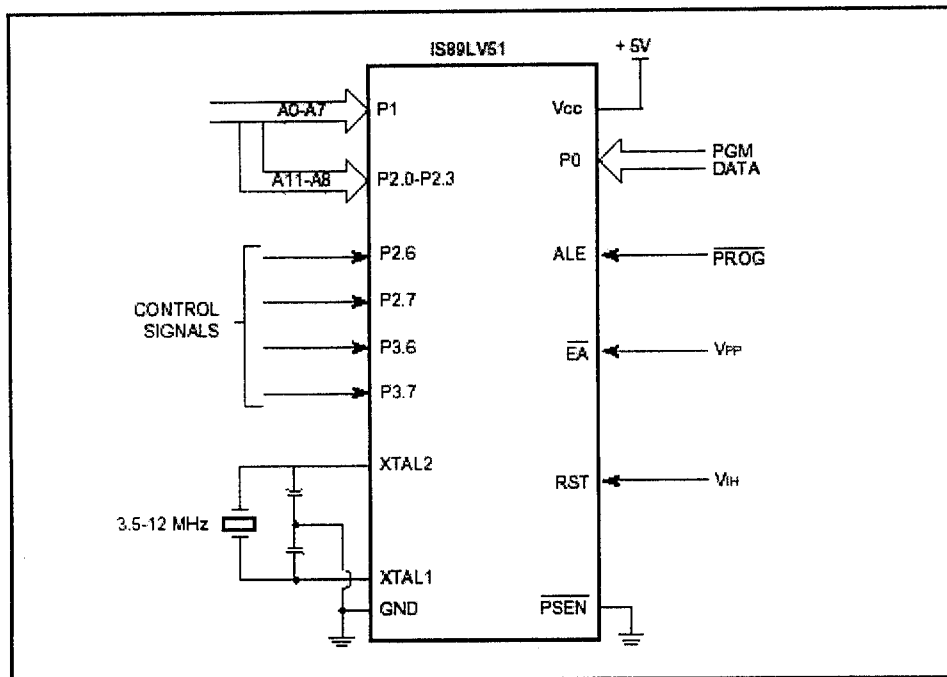


# Architecture of 8951

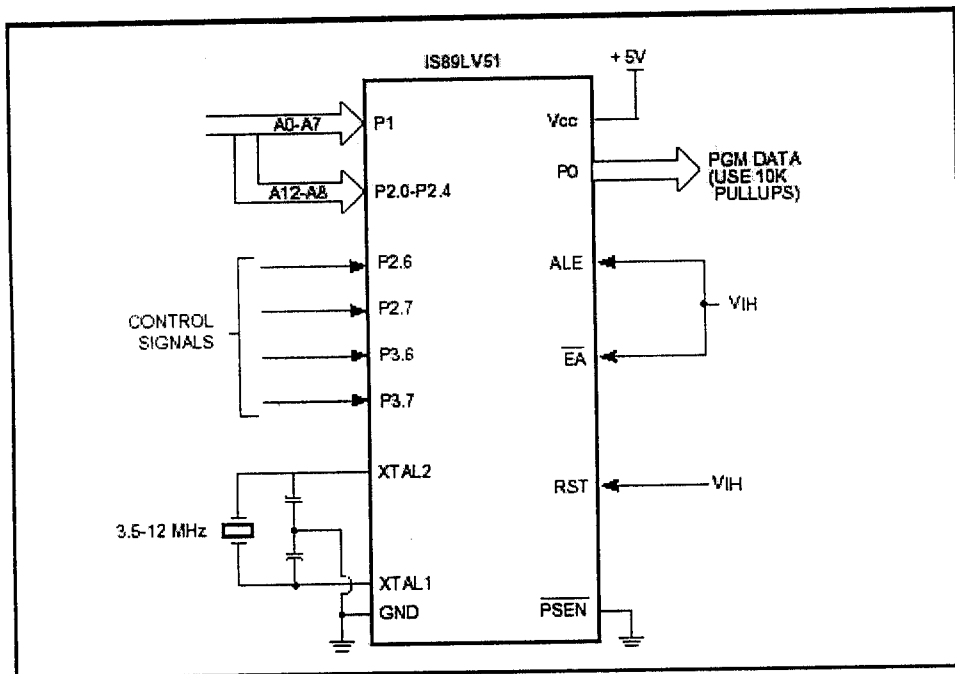


# Pin Configuration

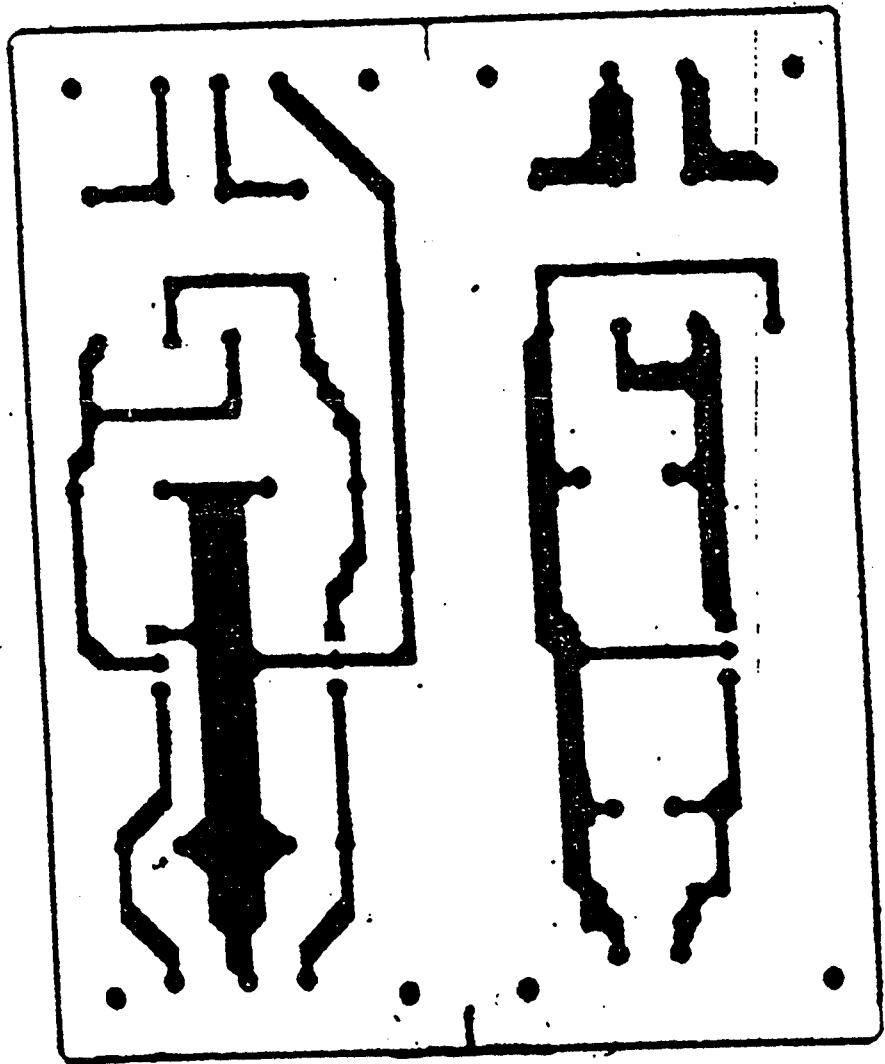


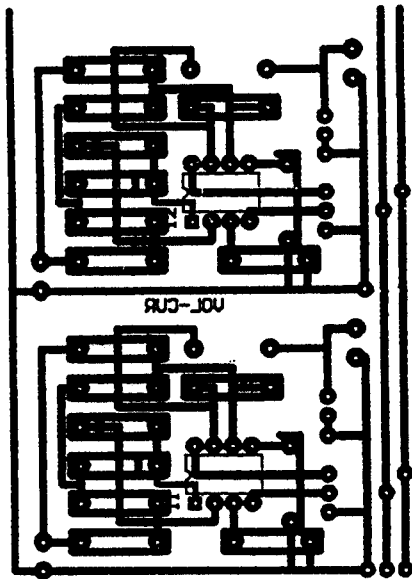


Programming the Flash Memory

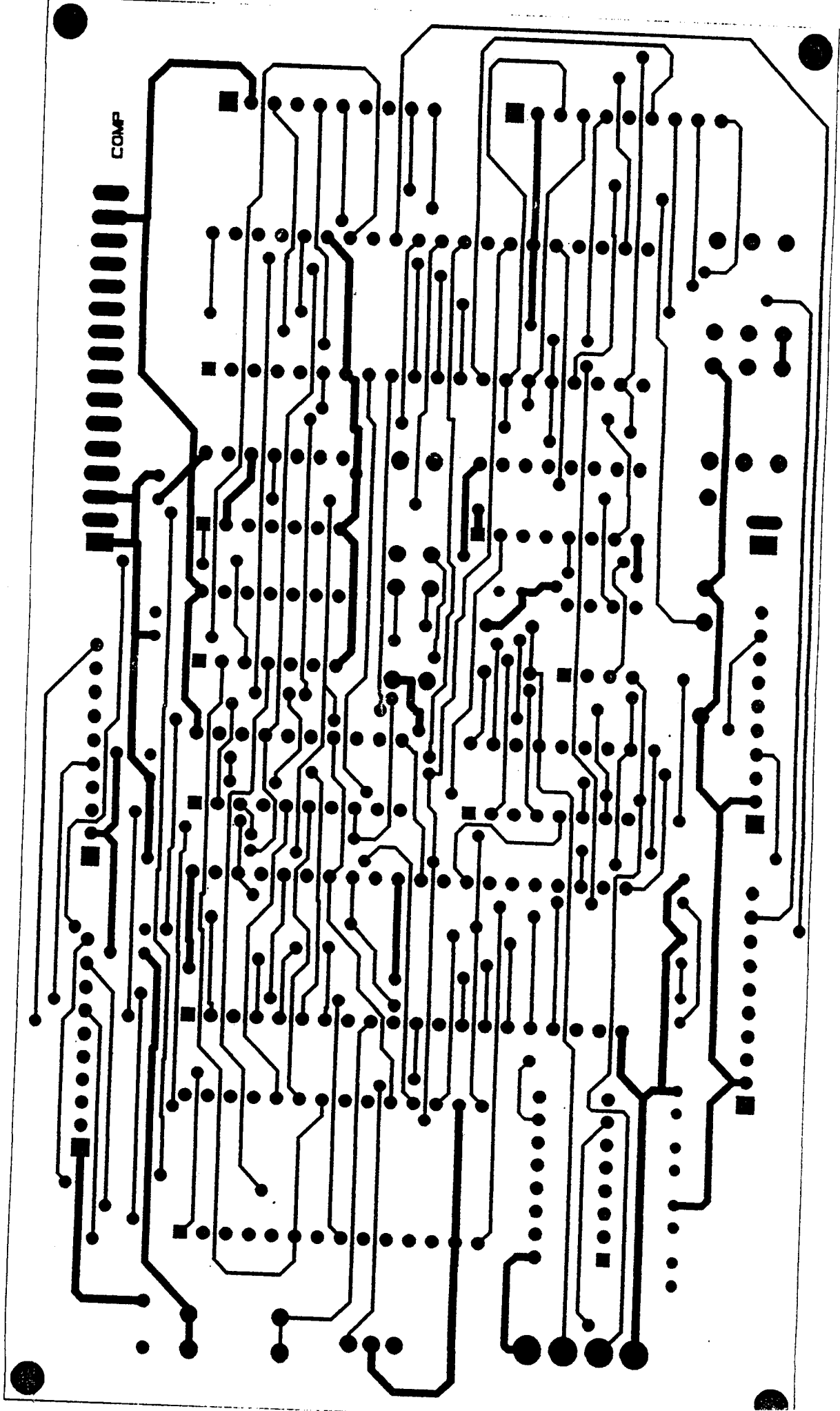


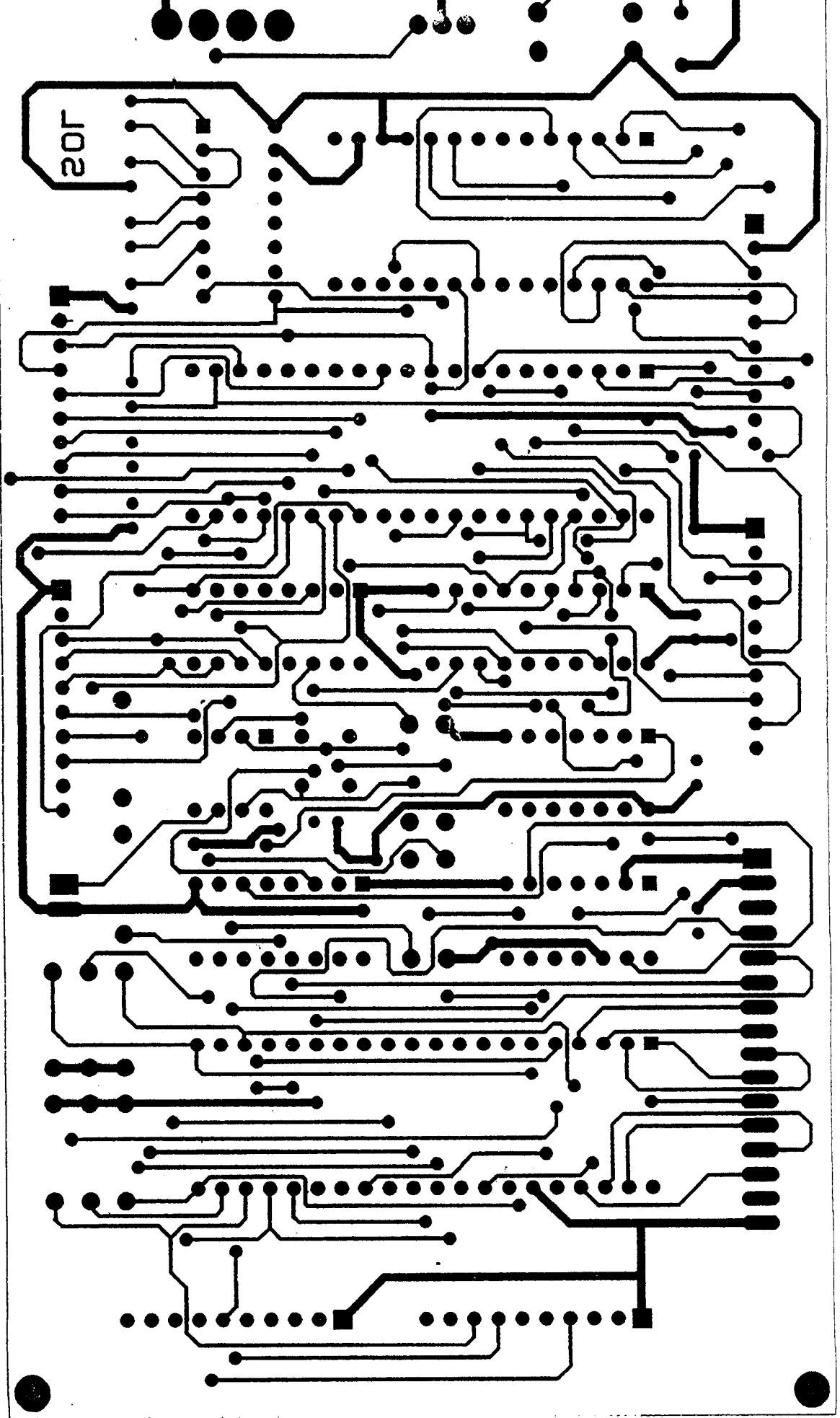
Verifying the Flash Memory











## ICL7106, ICL7107, ICL7107S

### Absolute Maximum Ratings

Supply Voltage	
ICL7106, V+ to V-	15V
ICL7107, V+ to GND	6V
ICL7107, V- to GND	9V
Analog Input Voltage (Either Input) (Note 1)	V+ to V-
Reference Input Voltage (Either Input)	V+ to V-
Clock Input	
ICL7106	TEST to V+
ICL7107	GND to V+

### Thermal Information

Thermal Resistance (Typical, Note 2)	$\theta_{JA}$ (°C/W)
PDIP Package	50
MQFP Package	75
Maximum Junction Temperature	150°C
Maximum Storage Temperature Range	-65°C to 150°C
Maximum Lead Temperature (Soldering 10s)	300°C
(MQFP - Lead Tips Only)	

### Operating Conditions

Temperature Range	0°C to 70°C
-------------------	-------------

*CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.*

#### NOTES:

- Input voltages may exceed the supply voltages provided the input current is limited to  $\pm 100\mu\text{A}$ .
- $\theta_{JA}$  is measured with the component mounted on a low effective thermal conductivity test board in free air. See Tech Brief TB379 for details.

### Electrical Specifications (Note 3)

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
<b>SYSTEM PERFORMANCE</b>					
Zero Input Reading	$V_{IN} = 0.0\text{V}$ , Full Scale = 200mV	-000.0	$\pm 000.0$	+000.0	Digital Reading
Stability (Last Digit) (ICL7106S, ICL7107S Only)	Fixed Input Voltage (Note 6)	-000.0	$\pm 000.0$	+000.0	Digital Reading
Ratiometric Reading	$V_{IN} = V_{REF}$ , $V_{REF} = 100\text{mV}$	999	999/1000	1000	Digital Reading
Roll-over Error	$-V_{IN} = +V_{IN} = 200\text{mV}$ Difference in Reading for Equal Positive and Negative Inputs Near Full Scale	-	$\pm 0.2$	$\pm 1$	Counts
Linearity	Full Scale = 200mV or Full Scale = 2V Maximum Deviation from Best Straight Line Fit (Note 5)	-	$\pm 0.2$	$\pm 1$	Counts
Common Mode Rejection Ratio	$V_{CM} = 1\text{V}$ , $V_{IN} = 0\text{V}$ , Full Scale = 200mV (Note 5)	-	50	-	$\mu\text{V/V}$
Noise	$V_{IN} = 0\text{V}$ , Full Scale = 200mV (Peak-To-Peak Value Not Exceeded 95% of Time)	-	15	-	$\mu\text{V}$
Leakage Current Input	$V_{IN} = 0$ (Note 5)	-	1	10	pA
Zero Reading Drift	$V_{IN} = 0$ , 0°C To 70°C (Note 5)	-	0.2	1	$\mu\text{V}/^\circ\text{C}$
Scale Factor Temperature Coefficient	$V_{IN} = 199\text{mV}$ , 0°C To 70°C, (Ext. Ref. 0ppm/°C) (Note 5)	-	1	5	ppm/°C
End Power Supply Character V+ Supply Current	$V_{IN} = 0$ (Does Not Include LED Current for ICL7107)	-	1.0	1.8	mA
End Power Supply Character V- Supply Current	ICL7107 Only	-	0.6	1.8	mA
COMMON Pin Analog Common Voltage	25k $\Omega$ Between Common and Positive Supply (With Respect to + Supply)	2.4	3.0	3.2	V
Temperature Coefficient of Analog Common	25k $\Omega$ Between Common and Positive Supply (With Respect to + Supply)	-	80	-	ppm/°C
<b>DISPLAY DRIVER ICL7106 ONLY</b>					
Peak-To-Peak Segment Drive Voltage	$V_{+} = \text{to } V_{-} = 9\text{V}$ (Note 4)	4	5.5	6	V
Peak-To-Peak Backplane Drive Voltage					

## Design Information Summary Sheet

### • OSCILLATOR FREQUENCY

$$f_{OSC} = 0.45/RC$$
$$C_{OSC} > 50pF; R_{OSC} > 50k\Omega$$
$$f_{OSC} (Typ) = 48kHz$$

### • OSCILLATOR PERIOD

$$t_{OSC} = RC/0.45$$

### • INTEGRATION CLOCK FREQUENCY

$$f_{CLOCK} = f_{OSC}/4$$

### • INTEGRATION PERIOD

$$t_{INT} = 1000 \times (4/f_{OSC})$$

### • 60/50Hz REJECTION CRITERION

$$t_{INT}/t_{60Hz} \text{ or } t_{INT}/t_{50Hz} = \text{Integer}$$

### • OPTIMUM INTEGRATION CURRENT

$$I_{INT} = 4\mu A$$

### • FULL SCALE ANALOG INPUT VOLTAGE

$$V_{INFS} (Typ) = 200mV \text{ or } 2V$$

### • INTEGRATE RESISTOR

$$R_{INT} = \frac{V_{INFS}}{I_{INT}}$$

### • INTEGRATE CAPACITOR

$$C_{INT} = \frac{(I_{INT})(t_{INT})}{V_{INT}}$$

### • INTEGRATOR OUTPUT VOLTAGE SWING

$$V_{INT} = \frac{(I_{INT})(t_{INT})}{C_{INT}}$$

### • $V_{INT}$ MAXIMUM SWING:

$$(V^- + 0.5V) < V_{INT} < (V^+ - 0.5V); V_{INT} (Typ) = 2V$$

### • DISPLAY COUNT

$$COUNT = 1000 \times \frac{V_{IN}}{V_{REF}}$$

### • CONVERSION CYCLE

$$t_{CYC} = t_{CLOCK} \times 4000$$
$$t_{CYC} = t_{OSC} \times 16,000$$
$$\text{when } f_{OSC} = 48kHz; t_{CYC} = 333ms$$

### • COMMON MODE INPUT VOLTAGE

$$(V^- + 1V) < V_{IN} < (V^+ - 0.5V)$$

### • AUTO-ZERO CAPACITOR

$$0.01\mu F < C_{AZ} < 1\mu F$$

### • REFERENCE CAPACITOR

$$0.1\mu F < C_{REF} < 1\mu F$$

### • $V_{COM}$

Biased between  $V_i$  and  $V^-$ .

### • $V_{COM} \cong V^+ - 2.8V$

Regulation lost when  $V^+$  to  $V^- < \cong 6.8V$

If  $V_{COM}$  is externally pulled down to  $(V^+ + V^-)/2$ , the  $V_{COM}$  circuit will turn off.

### • ICL7106 POWER SUPPLY: SINGLE 9V

$$V^+ - V^- = 9V$$

Digital supply is generated internally

$$V_{GND} \cong V^+ - 4.5V$$

### • ICL7106 DISPLAY: LCD

Type: Direct drive with digital logic supply amplitude.

### • ICL7107 POWER SUPPLY: DUAL $\pm 5.0V$

$$V^+ = +5V \text{ to GND}$$

$$V^- = -5V \text{ to GND}$$

Digital Logic and LED driver supply  $V^+$  to GND

### • ICL7107 DISPLAY: LED

Type: Non-Multiplexed Common Anode

## Absolute Maximum Ratings\*

Operating Temperature .....	-55°C to +125°C
Storage Temperature .....	-65°C to +150°C
Voltage on Any Pin with Respect to Ground .....	-1.0V to +7.0V
Maximum Operating Voltage .....	6.6V
DC Output Current .....	15.0 mA

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

## DC Characteristics

$T_A = -40^\circ\text{C}$  to  $85^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 20\%$  (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
$V_{IL}$	Input Low-voltage	(Except EA)	-0.5	$0.2 V_{CC} - 0.1$	V
$V_{IL1}$	Input Low-voltage (EA)		-0.5	$0.2 V_{CC} - 0.3$	V
$V_{IH}$	Input High-voltage	(Except XTAL1, RST)	$0.2 V_{CC} + 0.9$	$V_{CC} + 0.5$	V
$V_{IH1}$	Input High-voltage	(XTAL1, RST)	$0.7 V_{CC}$	$V_{CC} + 0.5$	V
$V_{OL}$	Output Low-voltage <sup>(1)</sup> (Ports 1,2,3)	$I_{OL} = 1.6 \text{ mA}$		0.45	V
$V_{OL1}$	Output Low-voltage <sup>(1)</sup> (Port 0, ALE, PSEN)	$I_{OL} = 3.2 \text{ mA}$		0.45	V
$V_{OH}$	Output High-voltage (Ports 1,2,3, ALE, PSEN)	$I_{OH} = -60 \mu\text{A}$ , $V_{CC} = 5\text{V} \pm 10\%$	2.4		V
		$I_{OH} = -25 \mu\text{A}$	$0.75 V_{CC}$		V
		$I_{OH} = -10 \mu\text{A}$	$0.9 V_{CC}$		V
$V_{OH1}$	Output High-voltage (Port 0 in External Bus Mode)	$I_{OH} = -800 \mu\text{A}$ , $V_{CC} = 5\text{V} \pm 10\%$	2.4		V
		$I_{OH} = -300 \mu\text{A}$	$0.75 V_{CC}$		V
		$I_{OH} = -80 \mu\text{A}$	$0.9 V_{CC}$		V
$I_D$	Logical 0 Input Current (Ports 1,2,3)	$V_{IN} = 0.45\text{V}$		-50	$\mu\text{A}$
$I_{TL}$	Logical 1 to 0 Transition Current (Ports 1,2,3)	$V_{IN} = 2\text{V}$ , $V_{CC} = 5\text{V} \pm 10\%$		-650	$\mu\text{A}$
$I_{L1}$	Input Leakage Current (Port 0, EA)	$0.45 < V_{IN} < V_{CC}$		$\pm 10$	$\mu\text{A}$
RRST	Reset Pull-down Resistor		50	300	$\text{K}\Omega$
$C_{IO}$	Pin Capacitance	Test Freq. = 1 MHz, $T_A = 25^\circ\text{C}$		10	pF
$I_{CC}$	Power Supply Current	Active Mode, 12 MHz		20	mA
		Idle Mode, 12 MHz		5	mA
	Power-down Mode <sup>(2)</sup>	$V_{CC} = 6\text{V}$		100	$\mu\text{A}$
		$V_{CC} = 3\text{V}$		40	$\mu\text{A}$

- Notes: 1. Under steady state (non-transient) conditions,  $I_{OL}$  must be externally limited as follows:  
 Maximum  $I_{OL}$  per port pin: 10 mA  
 Maximum  $I_{OL}$  per 8-bit port: Port 0: 26 mA  
 Ports 1, 2, 3: 15 mA  
 Maximum total  $I_{OL}$  for all output pins: 71 mA  
 If  $I_{OL}$  exceeds the test condition,  $V_{OL}$  may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum  $V_{CC}$  for Power-down is 2V.

## AC Characteristics

Under operating conditions, load capacitance for Port 0, ALE/ $\overline{\text{PROG}}$ , and  $\overline{\text{PSEN}}$  = 100 pF; load capacitance for all other outputs = 80 pF.

### External Program and Data Memory Characteristics

Symbol	Parameter	12 MHz Oscillator		16 to 24 MHz Oscillator		Units
		Min	Max	Min	Max	
$1/f_{\text{CLCL}}$	Oscillator Frequency			0	24	MHz
$t_{\text{AHL}}$	ALE Pulse Width	127		$2t_{\text{CLCL}}-40$		ns
$t_{\text{AVL}}$	Address Valid to ALE Low	43		$t_{\text{CLCL}}-13$		ns
$t_{\text{CLAX}}$	Address Hold After ALE Low	48		$t_{\text{CLCL}}-20$		ns
$t_{\text{LIV}}$	ALE Low to Valid Instruction In		233		$4t_{\text{CLCL}}-65$	ns
$t_{\text{LPL}}$	ALE Low to $\overline{\text{PSEN}}$ Low	43		$t_{\text{CLCL}}-13$		ns
$t_{\text{PLPH}}$	$\overline{\text{PSEN}}$ Pulse Width	205		$3t_{\text{CLCL}}-20$		ns
$t_{\text{PLIV}}$	$\overline{\text{PSEN}}$ Low to Valid Instruction In		145		$3t_{\text{CLCL}}-45$	ns
$t_{\text{PIX}}$	Input Instruction Hold After $\overline{\text{PSEN}}$	0		0		ns
$t_{\text{FIXZ}}$	Input Instruction Float After $\overline{\text{PSEN}}$		59		$t_{\text{CLCL}}-10$	ns
$t_{\text{PXAV}}$	$\overline{\text{PSEN}}$ to Address Valid	75		$t_{\text{CLCL}}-8$		ns
$t_{\text{AVIV}}$	Address to Valid Instruction In		312		$5t_{\text{CLCL}}-55$	ns
$t_{\text{PLAZ}}$	$\overline{\text{PSEN}}$ Low to Address Float		10		10	ns
$t_{\text{RLPH}}$	$\overline{\text{RD}}$ Pulse Width	400		$6t_{\text{CLCL}}-100$		ns
$t_{\text{WLWH}}$	$\overline{\text{WR}}$ Pulse Width	400		$6t_{\text{CLCL}}-100$		ns
$t_{\text{RLDV}}$	$\overline{\text{RD}}$ Low to Valid Data In		252		$5t_{\text{CLCL}}-90$	ns
$t_{\text{RHDX}}$	Data Hold After $\overline{\text{RD}}$	0		0		ns
$t_{\text{RHNDZ}}$	Data Float After $\overline{\text{RD}}$		97		$2t_{\text{CLCL}}-28$	ns
$t_{\text{LLDV}}$	ALE Low to Valid Data In		517		$8t_{\text{CLCL}}-150$	ns
$t_{\text{AVDV}}$	Address to Valid Data In		585		$9t_{\text{CLCL}}-165$	ns
$t_{\text{LLWL}}$	ALE Low to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	200	300	$3t_{\text{CLCL}}-50$	$3t_{\text{CLCL}}+50$	ns
$t_{\text{AVWL}}$	Address to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	203		$4t_{\text{CLCL}}-75$		ns
$t_{\text{OVWX}}$	Data Valid to $\overline{\text{WR}}$ Transition	23		$t_{\text{CLCL}}-20$		ns
$t_{\text{OVWH}}$	Data Valid to $\overline{\text{WR}}$ High	433		$7t_{\text{CLCL}}-120$		ns
$t_{\text{WHDX}}$	Data Hold After $\overline{\text{WR}}$	33		$t_{\text{CLCL}}-20$		ns
$t_{\text{RLAZ}}$	$\overline{\text{RD}}$ Low to Address Float		0		0	ns
$t_{\text{WHLH}}$	$\overline{\text{RD}}$ or $\overline{\text{WR}}$ High to ALE High	43	123	$t_{\text{CLCL}}-20$	$t_{\text{CLCL}}+25$	ns

## Flash Programming and Verification Characteristics

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5.0 \pm 10\%$

Symbol	Parameter	Min	Max	Units
$V_{PP}^{(1)}$	Programming Enable Voltage	11.5	12.5	V
$I_{PP}^{(1)}$	Programming Enable Current		1.0	mA
$f_{ACLCL}$	Oscillator Frequency	3	24	MHz
$t_{AVGL}$	Address Setup to $\overline{\text{PROG}}$ Low	$48t_{CLCL}$		
$t_{GHAX}$	Address Hold After $\overline{\text{PROG}}$	$48t_{CLCL}$		
$t_{DVGL}$	Data Setup to $\overline{\text{PROG}}$ Low	$48t_{CLCL}$		
$t_{GHDX}$	Data Hold After $\overline{\text{PROG}}$	$48t_{CLCL}$		
$t_{EHS}$	P2.7 ( $\overline{\text{ENABLE}}$ ) High to $V_{PP}$	$48t_{CLCL}$		
$t_{SHGL}$	$V_{PP}$ Setup to $\overline{\text{PROG}}$ Low	10		$\mu\text{s}$
$t_{GHS}^{(1)}$	$V_{PP}$ Hold After $\overline{\text{PROG}}$	10		$\mu\text{s}$
$t_{GLGH}$	$\overline{\text{PROG}}$ Width	1	110	$\mu\text{s}$
$t_{AVDV}$	Address to Data Valid		$48t_{CLCL}$	
$t_{ELOV}$	$\overline{\text{ENABLE}}$ Low to Data Valid		$48t_{CLCL}$	
$t_{EHOZ}$	Data Float After $\overline{\text{ENABLE}}$	0	$48t_{CLCL}$	
$t_{GHBL}$	$\overline{\text{PROG}}$ High to $\overline{\text{BUSY}}$ Low		1.0	$\mu\text{s}$
$t_{WC}$	Byte Write Cycle Time		2.0	ms

Note: 1. Only used in 12-volt programming mode.