Design & Development of Intelligent Mobile Robot

Project Report

Submitted By

Hari Prashanth.R

Karthikeyan.K.B

Ramachandran.T.V

Thiruvenkadam.V.

Guided By

G. Mohan Kumar, M.E., MBA., PGDMrM,
MIE,MISTE,MSAE.

SENIOR LECTURER, DEPARTMENT OF MECHATRONICS ENGINEERING

In partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF ENGINEERING IN MECHATRONICS

of the Bharathiar University, Coimbatore.

Department of Mechatronics Engineering

Kumaraguru College of Technology

Coimbatore – 641 006.

Kumaraguru College of Technology

Department of Mechatronics Engineering

Coimbatore – 641 006

CERTIFICATE

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

'MOBILE INTELLIGENT ROBOT'

has been submitted by Mr / Ms . . . .

In partial fulfillment of the requirement for the award of the Degree of **BACHELOR OF ENGINEERING IN MECHATRONICS** Branch of the Bharathiar University, Coimbatore.

During the academic year 2002 – 2003

18/3/03

. . . .
Internal Guide

. . . .
Head of the Department

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

was examined by us in Project Viva – Voce Examination held on 21/5/03 . . . . .

. . . .
Internal Examiner

. . . .
External Examiner

Dedicated to our parents
who sacrificed their today for
our better tomorrow...

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

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

We express our gratitude and indebtedness to our respected guide **Dr.V.Gunaraj** M.E., PhD., Head of the department of Mechatronics engineering for his full fledged technical guidance, constant encouragement and suggestions in carrying out this project.

We also take this opportunity to thank our Assistant Professor **Mr.P.Rajendran** M.E., for sharing his in-depth knowledge on the subject and providing us with tips as and when required.

We also thank our Project Guide Mr. **G. Mohan Kumar,** M.E., MBA, PGDMrM, for his valuable guidance and encouragement at all stages of the project work.

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

CONTENTS

SYNOPSIS

1.0 SYNOPSIS

The 'Robot' is always a fascinating thing for all, so is for us. Being Mechatronics students we consider our duty to work for a Robot project. The Project gives an opportunity to use our knowledge that we have acquired for the past four years.

The Robot is really a challenging project to work with. The circuits are to be designed and appropriate Mechanical model is to be created. We also needed the help of computers to work for the physical design of the Project which helped us to get an optimal design.

The Objective behind a Robot Project is showing everyone that we are fully capable of developing a multidisciplinary product. As every product that is created needs an industrial application, we also have developed this Robot having a wide range of applications in the Industrial Environment

To sight a few examples how this Robot can be used in the industries are:

- Removing of unused and waste nuts, bolts and similar particles from the assembly lines.
- Separation of iron particles from the left-over casting sand.

The Robot is Intelligent enough to detect any kind of obstacle that is placed in front of it and avoid it. The Robot has 3 axis freedoms the movements two of them are in X and Y directions and third movement is in the Z direction. The Robot design is very flexible and can accommodate any kind of futuristic advancements.

2.0 Introduction

The robot has basically four configurations

1. Polar configuration
2. Cylindrical configuration
3. Cartesian configuration
4. Jointed arm configuration

Polar configuration has telescoping arm that can be raised or lowered about a horizontal pivot. The pivot is mounted on a rotating base.

The cylindrical configuration uses a vertical column and a slide that can be moved up or down the column. The robot arm is attached to the slide so that it can be moved radically with respect to the column.

The Cartesian coordinate uses three perpendicular slides to construct the x, y, and z axes.

The joined arm is similar to that of the human arm. It consists of two straight components, corresponding to the human forearm and upper arm, mounted on a vertical pedestal.

The robot that we have developed is a combination of the polar and Cartesian configuration.

2.1 Control System and Dynamic Performance:

In order to operate, a robot must have means of controlling its drive system to properly regulate its motion. There are basically four categories of robot control

1. Limited sequence robots
2. Play back robots with point to point control
3. Playback robots with continuous path control
4. Intelligent robots

The robot being built by us is an intelligent Robot which has a control of the path. The location of the part is being given as the input to the robot and the path to reach the object is being done by the robot itself.

The spatial resolution of the robot is the smallest increment of movement into which the robot can divide its volume. In our case the robot has a resolution of 1.884mm

2.2 End Effectors

For the industrial application, the capabilities of the basic robot must be augmented by means of additional devices. They include the tooling which attaches to the robot's wrist and the sensor systems which allow the robot to interact with the environment. The Robot is equipped with an electromagnetic End-effectors which when supplied with the necessary voltages is converted into a magnet which is used for picking up the nuts and bolts and other magnetic substances.

2.3 Robot Motion analysis and control

In order for a robot to perform useful work is it necessary for the arm to consist a number of joints. It is necessary to control the path which end of the arm follows, as opposed to merely controlling the final positions of the joints.

2.4 Positional Representation

The kinematics of the RR robot are more difficult to analyze that the LL robot. One way is to utilize the two joints angles $Ø_1$ and $Ø_2$. This is known as the representation in "Joint Space" and we may specify it as

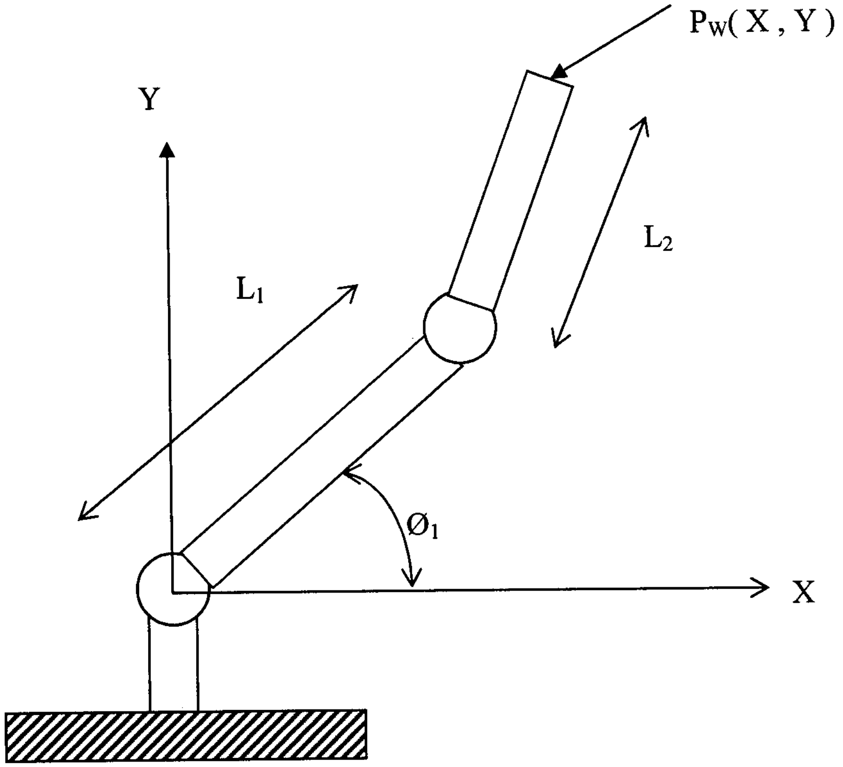
$$P_j = (Ø_1 \ , \ Ø_2)$$

Another way to define the arm position is in " world" space. This involves the use of a certain coordinate system that is external to the robot.

This is represented as

$$P = (X , Y)$$

Going from joint space to the world space is called forward transformation and going from world space to joint space is called the reverse transformation.



A two dimensional 2 degree-of- freedom manipulator(type RR)

2.4 Forward Transformation of a 2-Degree of Freedom Arm:

The position of the end in the world space, the coordinates x and y of the end of the arm (point P) in the world space

$$X = L_1 \cos \varnothing_1 + L_2 \cos (\varnothing_1 + \varnothing_2)$$

$$Y = L_1 \sin \varnothing_1 + L_2 \sin (\varnothing_1 + \varnothing_2)$$

The position of the end of a toll with respect to the base frame would be given by

$$\mathbf{T = CAG_n}$$

The arm's base is related to the camers's base by the transformation **C**. The arm is described by the transformation A and the tools by G

2.5 Robot Dynamics:

Accurate control of the manipulator requires precise control of each joint. The control of the joint depends on knowledge of the forces that will be acting on the joint. While these forces and masses are relatively easy to determine for a single joint, it becomes more difficult to determine them as the complexity of the manipulator increases.

The torque generated at the end of the robot can be separated as two components of forces F_x and F_y.

Where

$$F_x = \frac{T_1 \; L_2 \cos(\O_1 + \O_2) - T_2 \,(L_1 \cos \O_1 + L_2 \cos(\O_1 + \O_2))}{L_1 L_2 \sin \O_2}$$

$$F_y = \frac{T_1 \; L_2 \sin(\O_1 + \O_2) - T_2 \,(L_1 \sin \O_1 + L_2 \sin(\O_1 + \O_2))}{L_1 L_2 \sin \O_2}$$

2.6 Configuration of A Robot Controller

The elements needed in the controller include

- Joint servo controllers
- Joint Power Amplifiers
- Mathematical processor
- Executive processor
- Program memory
- Input devices

The number of joint in the manipulator and joint power amplifier would correspond to the number of joints in the manipulator. Motion commands are executed by the controller from two possible sources: operator input or program memory. Microcontrollers are typically utilized in several of the component of the Robot controller. Each of the control boards makes use of a common data bus and address bus. The microcontroller communicates with each other by sending messages into common areas in the system memory.

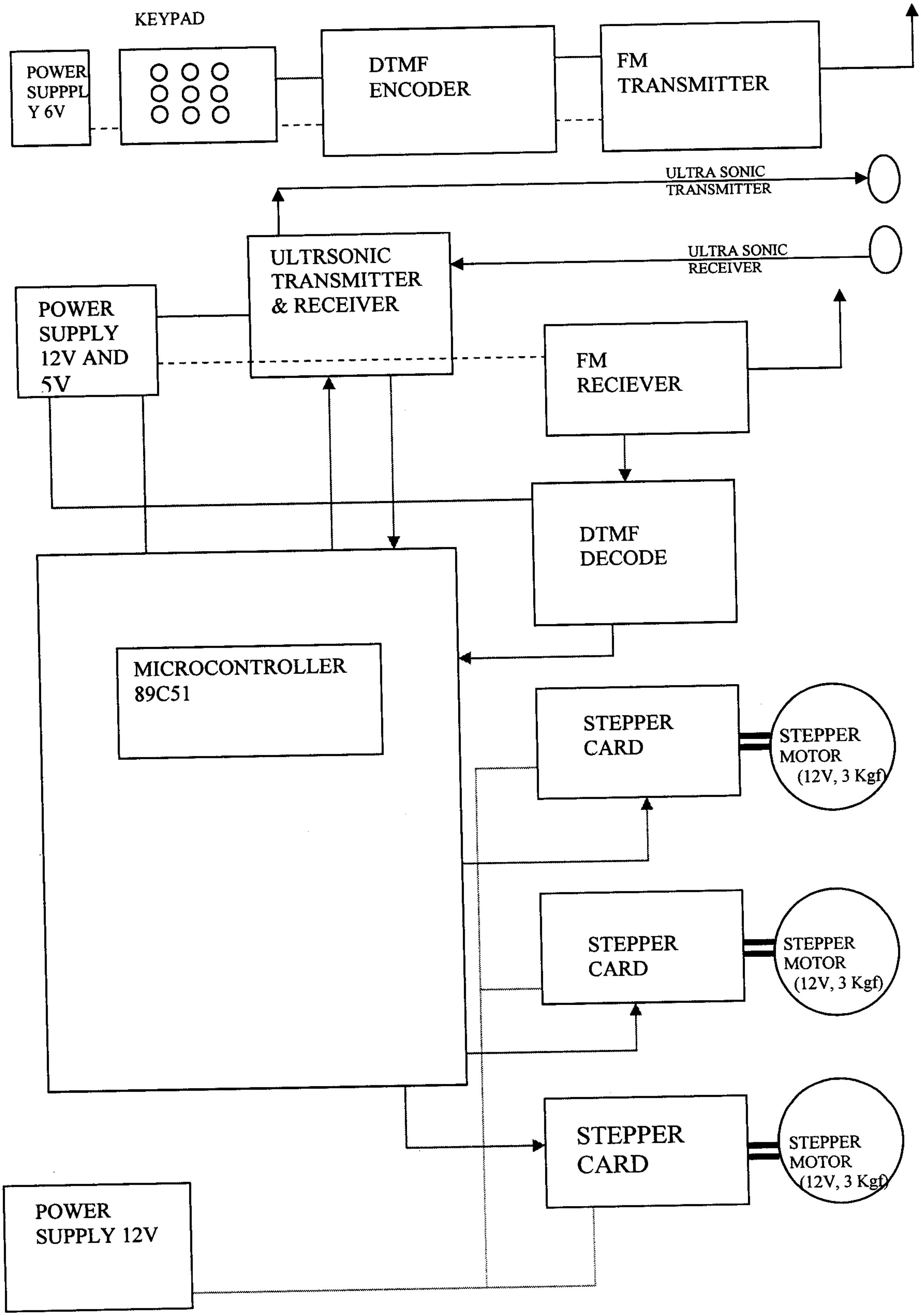
The Robot design can be divided into two parts

- Hardware design
- Mechanical design

3.0 HARDWARE DESIGN

The hardware unit consists of the following circuits

- Microcontroller
- Stepper motor drive card
- Power supply circuits
- Ultrasonic transmitter and receiver
- DTMF encoder and transmitter
- Relay circuit



Block diagram of the hardware of the Robot

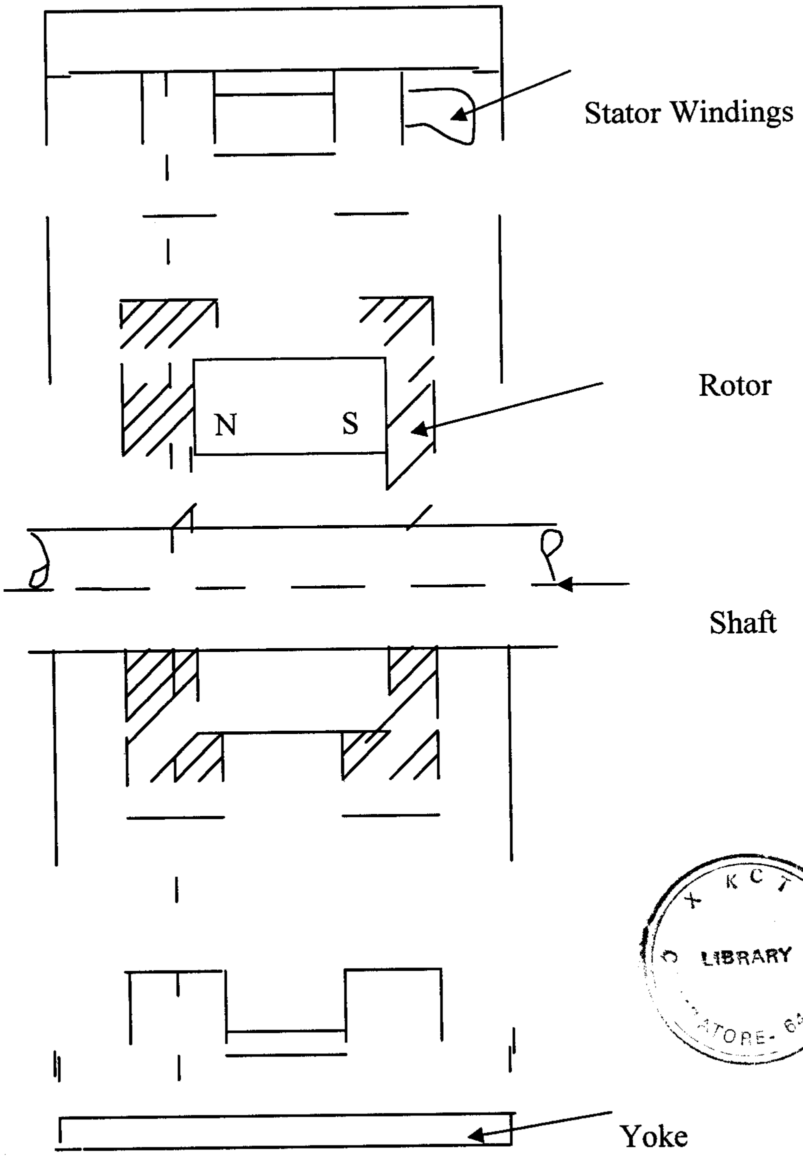
4.0 STEPPER MOTORS

Stepper motors were first introduced in 1957. It took more than a decade for these to come to India. The production of stepper motors in India picked up only in the last decade or so with the growth of computer peripherals, office equipment and CNC machine tool industries. This led to the creation of a sizeable user-base in India. However, most users are faced with difficulties in selecting stepper motors of the right size, as very little information is available on them.

A stepper motor is defined in British standard specification as "*a reversible brush less DC motor, the rotor of which rotates in discrete angular steps when its stator windings are energized in a programmed manner, Rotation occurs as a result of interaction between the rotor poles and the poles of the sequentially energized stator windings. The rotor carries no electrical windings but rather has salient soft or magnetized poles*".

Stepper motors are of three basic types: (a) permanent magnet (PM) – utilizes a rotor having permanently magnetized salient poles of hard magnetic material; (b) Variable reluctance (VR) – soft magnetic material; and (c) hybrid (HY) – utilizes a rotor comprising an axial permanent magnet magnet juxtaposed between two soft iron rotor discs having salient poles, as shown in figure-1

In a stepper motor, the task of energizing the stator windings in the prescribed sequence is performed by an electronic controller in response to pulses received from an exogenous source, such as a microcomputer or a programmed tape or disc, as shown in figure. The stepper motor is thus an incremental actuator, which receives input in the form of pulses, and whose output consequently is the actuator of choice in computer control systems.



(Axial view of the permanent magnet hybrid stepper motor)

4.1 Applications:

Stepper motors are employed in a variety of applications. These applications may be divided into the following classes:

Instrumentation – electronic analogue watches, clocks on railway platforms/offices/factories, and camera shutter operation, photo-printing machines etc.

Computer peripherals – line printers, X – Y plotters, floppy disc and hard disc drives, paper reader and punch in a Teletype, etc. Office electronics equipment – electronic typewriters, telex machines, teleprinters, computer typesetting machines, etc.

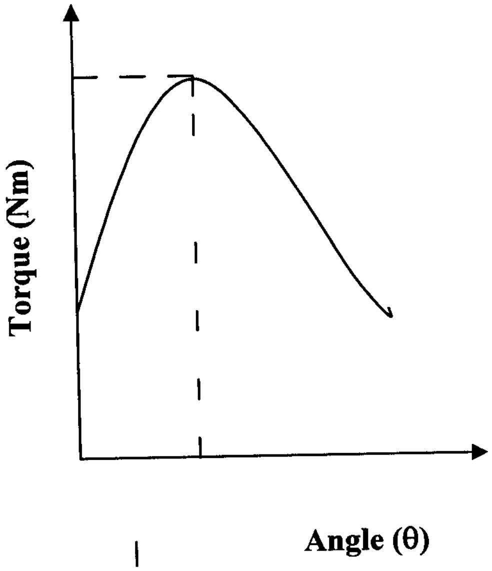
Machine tool controls – NC milling machines and lathes, CNC systems, machines, etc.

Heavy-duty applications – indexing tables, X-Y tables, X-ray table positioning, radiation treatment machine, CAT scanners, solar panel positioner in a satellite, etc.

4.2 Characteristics:

Characteristics that are of importance in the selection of stepper motors are:

- **Step Angle (θS):** It is the fixed angle through which the shaft of the unloaded stepper motor rotates when two adjacent stator phases are energized, singly in sequence. Step angle determines the resolution (least angular rotation that can be detected) and the positioning accuracy that can be achieved.
- **Holding torque (TH):** It is the maximum steady state torque at a specified current that can be applied to the shaft of an energized stepper motor, without causing slipping of the rotor. It is determined from the torque-angle curve of the stepper motor as shown.



- **Synchronism**: In stepper motor operation, synchronism means a strict one-to-one correspondence between the number of pluses applied to the motor controller and the number of steps through which the rotor actually moves. Synchronism is of vital importance in the design of incremental servo systems utilizing stepper motors.
- **Pull-in rate (FPI):** It is the maximum stepping rate up to which the motor can start or stop without losing synchronism, at a specified output torque.
- **Pullout rate (FpO)**: It is the maximum stepping rate up to which the stepper motor can slew without losing synchronism, at a specified output torque.
- **Pull-in Torque (TpI):** It is the maximum torque against which the motor can start or stop without losing synchronism, at a specified stepping rate.
- **Pullout torque (TpO):** It is the maximum torque at which the motor can slew without losing synchronism, at a fixed stepping rate. The torque vs. stepping rate curves defines these ratings

4.3 Incremental servo systems :

As mentioned earlier, synchronism between the number of input pulses applied to the motor controller and the number of discrete angular steps through which the motor rotates is of vital importance in stepper motor application. It is on account of this synchronism that we can keep a track of the angular displacement (θ) of the motor shaft by keeping a count of the number of input pulses.

There is no need for angular position sensor such as a potentiometer, synchro, resolver and encoder to determine θ. This implies that it is possible to design an open loop system utilizing a stepper motor for controlling angular position θ. Such a position control system, employing the stepper motor, is known as incremental servo; it is capable of yielding the same positioning accuracy as a closed loop system using DC or AC servomotor.

The main advantages of incremental servo over closed loop DC or AC servo are: (a) it is simple and inexpensive to design and develop; (b) there are no stability problems since it is an open loop system; and (c) even then, it is capable of yielding high positioning accuracy comparable to that of closed loop DC or AC servo systems. In this way, synchronism is the cardinal principle in the design of incremental servo systems. Naturally, stepper motor selection is also governed by the same principle of synchronism.

BASIC CONSIDERATIONS IN STEPPER MOTOR SELECTION:

In addition to synchronism, there are three further basic considerations for selecting stepper motors for a particular application. These are: (a) positioning accuracy (δ) – expressed in degrees or minutes of arc (rotary motion) – mm or microns (linear motion); (b) speed of operation (V) – specified in degrees or rad/sec, or rpm (rotary motion) –mm or m per sec.(linear motion); and (c) acceleration (α) – specified indirectly in terms of the time Δt required to attain operating speed V.

It is possible to determine the step angle θs of the motor from the positioning accuracy. Stepping rate Fs can similarly be determined from the operating speed V. Acceleration in rad/sec^2 is computed from the ratio V/ Δt, where V is the time required to attain it from rest.

The load determines the total inertia J_T reflected to the motor shaft and the drive used to couple it to the motor. Knowledge of J_T and α enables us to compute accelerating torque $J\alpha$. Adding friction and load torques yields the total torque T_M that must be developed by the motor.

Once these basic data are available, we may proceed to select a stepper motor of proper size in the following sequence.

i. Select at the outset a range of stepper motors having step angle θ_s, which meets positioning accuracy requirements.

ii. Refer to torque vs stepping rate curves of the above motors, and select a motor which is capable of developing a torque T_M (calculated above) at a pull-in rate of F_s steps/sec. detailed procedure for stepper motor selection is shown.

4.4 Selection of typical stepper motor drives:

Four typical drives are used in incremental servos to couple the motor to the load, namely, the direct drive, string and pulley drive, rack and pinion drive, and lead screw drive.

Direct drive: This is the simplest possible drive in which the stepper motor is directly connected to the load.

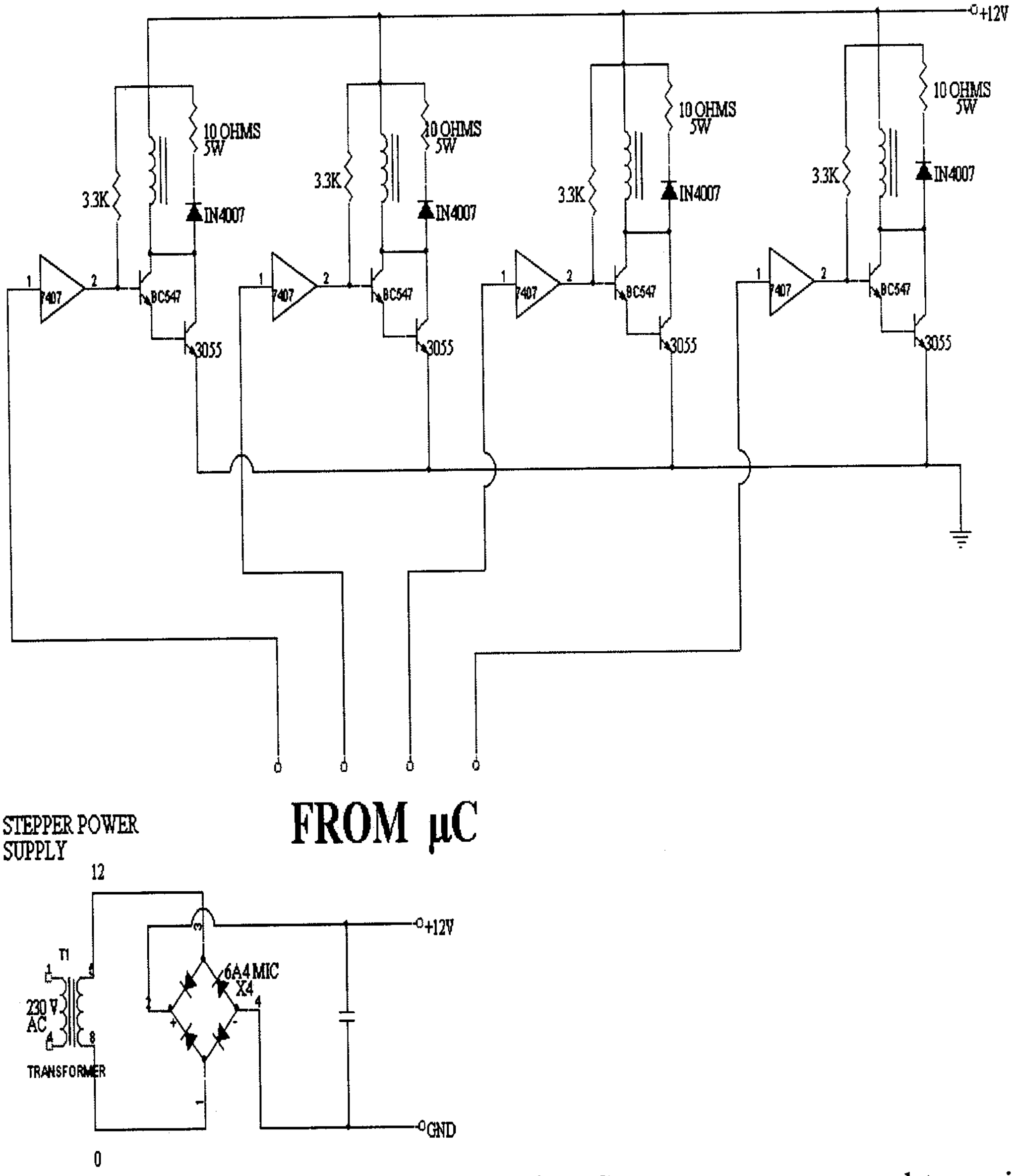
String and pulley drive: This drive is used when the stepper motor has to raise a load.

Rack and pinion drive: This drive is selected when the load is to be moved linearly in a horizontal plane.

Lead screw drive: This drive is preferred in NC and CNC systems for machine tools such as milling machines, turret lathes, jig boring machines, and planning machines.

Stepper Motor Driver Circuit:

A unique type of motor useful for moving things in small increments is the stepper motor. In dot matrix printer such as the Epson FX-80, one small stepper motor is used



move the print head to the next character position. Stepper motors are used to position the read/write head over the desired track of a hard or floppy disk, and to position the pen on X-Y plotters.

Instead of rotating smoothly around and around as most motors do, stepper motors rotate or step from one fixed position to the next. Common step sizes range from 0.9° to 30°. A stepper motor is stepped from one position to the next by changing the currents through the fields in the motor. The two common field connections are referred to as two-phase and four phase.

In our project we are using four phase stepper motor. The buffers are inverting, a high on an output pin turns on current to a winding. The switching sequence to step a motor such as this clockwise, as we face the motor shaft, or counter clockwise.

STEP	***SW4***	***SW3***	***SW2***	***SW1***
1	0	0	1	1
2	1	0	0	1
3	1	1	0	0
4	1	0	0	1
1	0	0	1	1

Clockwise direction↓

Counter clockwise direction ↑

Consider the four phases as SW1, SW2, SW3 & SW4.If SW1 and SW2 are turned on. Turning off SW2 and turning on SW4 will cause the motor to rotate one step of 1.8° clockwise. Changing to SW4 and SW3 on will cause the motor to rotate another 1.8° clockwise. Changing to SW3 and SW2 on will cause another step. After that, changing to SW2 and SW1 on again will cause another step clockwise. We can repeat the sequence until the motor has rotated as many steps clockwise as we want. To step the motor counter clockwise, work through the switch sequence in the reverse direction. In either case the motor will be held in its last position by the current through the coils. The switch sequence that can be used to rotate the motor half steps of 0.9° clockwise or counter clockwise.

STEP	SW4	SW3	SW2	SW1
1	0	0	1	1
2	0	0	0	1

4	1	0	0	0
5	1	1	0	0
6	0	1	0	0
7	0	1	1	0
8	0	0	1	0
1	0	0	1	1

To take the first step clockwise from SW2 and SW1 being on, the pattern of 1's and 0's is simply rotated one bit position around to the right. The 1 from SW1 is rotated around into bit 4. To take the next step the switch pattern is rotated one more bit position. To step counter clockwise the switch pattern is rotated left one bit position for each step desired. Suppose 00110011 is loaded into ACC and output this to the switches. Duplicating the switch pattern in the upper half of ACC will make stepping easy. To step the motor clockwise, just rotate this pattern right one bit position and output it to the switches. To step counter clockwise, rotate the switch pattern left one bit position and output it. After output one-step code just waits for a few milliseconds before output another step command, because the motor can only step so fast. Maximum stepping rates for different types of steppers vary from a few hundred steps per second to several thousand steps per second. To achieve high stepping rates the stepping rate is slowly increased to the maximum, then it is decreased as the desired number of steps is approached.

To add a stepper motor to the robot add the clamp diodes across each winding to save the transistors from inductive kick and the current limiting resistors (R1 & R2). The motor we used here has a nominal voltage rating of 5.5V. This means that we could have designed the circuit to operate with a voltage of about 6.5V on the emitters of the driver transistors. For low stepping rates, this would work fine. However for higher stepping rates and more torque while stepping, we use a higher supply voltage and current-limiting resistors R1 and R2. The designed the circuit to operate with voltage of about 6.5V on the emitters of the driver transistors (5.5V for the motor plus 1V for the drop across the transistor) for low stepping rates, this would work fine.

However, for higher stepping rates and more torque while stepping, we use a higher supply voltage and current -limiting resisters as shown. The point of this is that by adding series resistance, we decrease L/R time constant. This allows the current to change more rapidly in the windings.

For the motor we used, the current per winding is 0.88A. Since only one winding on each resistor is ever on at a time, 6.5V/0.88A gives a resistor value of 6.25Ω. To be conservative we used 8Ω, 10-W resistors. The optional transistor switch and diode connection to the +5V supply are used as follows.

When not stepping, the switch to +12V is off so the motor is held in position by the current from the +5V supply. Before we send a step command, we turn on the transistor to +12V to give the motor more current for stepping.

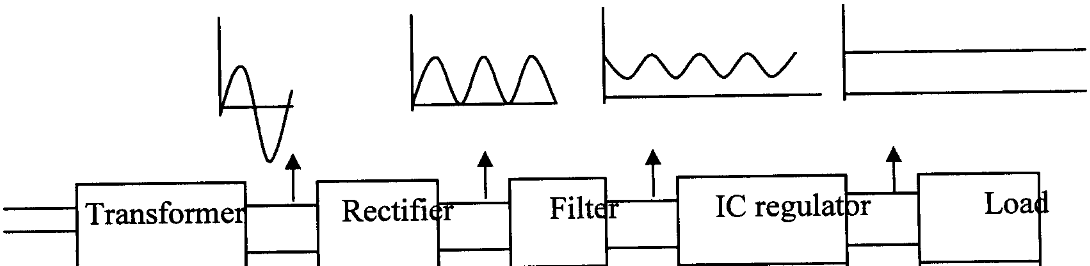
When stepping is done we turn off the switch to +12V, and drop back to the +5V supply. This cuts the power dissipation.

5.0 POWER SUPPLIES

5.1 Introduction:

The present chapter introduces the operation of power supply circuits built using filters, rectifiers, and then voltage regulators. Starting with an ac voltage, a steady dc voltage is obtained by rectifying the ac voltage, then filtering to a dc level, and finally, regulating to obtain a desired fixed dc voltage. The regulation is usually obtained from an IC voltage regulator unit, which takes a dc voltage and provides a somewhat lower dc voltage, which remains the same even if the input dc voltage varies, or the output load connected to the dc voltage changes.

A block diagram containing the parts of a typical power supply and the voltage at various points in the unit is shown in fig 19.1. The ac voltage, typically 120 V 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.



5.2 IC Voltage regulators:

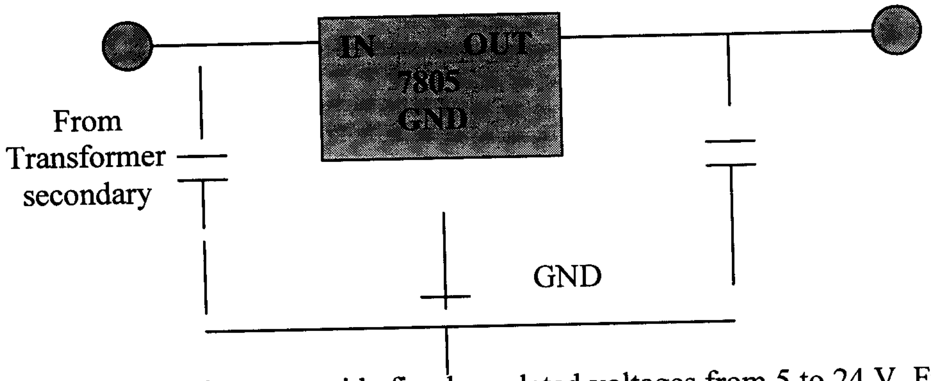
Voltage regulators comprise a class of widely used ICs. Regulator IC units contain the circuitry for reference source, comparator amplifier, control device, and overload protection all in a single IC. Although the internal construction of the IC is somewhat different from that described for discrete voltage regulator circuits, the external operation is much the same. IC units provide regulation of either a fixed positive voltage, a fixed negative voltage, or an adjustably set voltage.

A power supply can be built using a transformer connected to the ac supply line to step the ac voltage to a desired amplitude, then rectifying that ac voltage, filtering with a capacitor and RC filter, if desired, and finally regulating the dc voltage using an IC regulator. The regulators can be selected for operation with load currents from hundreds of milliamperes to tens of amperes, corresponding to power ratings from milliwatts to tens of watts.

THREE-TERMINAL VOLTAGE REGULATORS:

Fig shows the basic connection of a three-terminal voltage regulator IC to a load. The fixed voltage regulator has an unregulated dc input voltage, Vi, applied to one input terminal, a regulated output dc voltage, Vo, from a second terminal, with the third terminal connected to ground. For a selected regulator, IC device specifications list a voltage range over which the input voltage can vary to maintain a regulated output voltage over a range of load current. The specifications also list the amount of output voltage change resulting from a change in load current (load regulation) or in input voltage (line regulation).

FIXED POSITIVE VOLTAGE REGULATORS:



The series 78 regulators provide fixed regulated voltages from 5 to 24 V. Figure 19.26 shows how one such IC, a 7812, is connected to provide voltage regulation with output from this unit of +12V dc. An unregulated input voltage Vi is filtered by capacitor C1 and connected to the IC's IN terminal. The IC's OUT terminal provides a regulated + 12V which is filtered by capacitor C2 (mostly for any high-frequency noise). The third IC terminal is connected to ground (GND). While the input voltage may vary over some permissible voltage range, and the output load may vary over some acceptable range, the output voltage remains constant within specified voltage variation limits. These limitations are spelled out in the manufacturer's specification sheets. A table of positive voltage regulated ICs is provided in table 19.1. TABLE 19.1 Positive Voltage Regulators in 7800 series

6.0 DTMF

6.1 Introduction:

This is a full DTMF receiver that integrates both band split filter and decoder functions into a single 18-pin DIP or SOIC package. Manufactured using state of the art CMOS process technology, the M-8870 offers low power consumption and precise data handling. Its filter section uses switched capacitor technology for both the low and high group filters and for dial tone rejection. Its decoder uses digital counting techniques to detect and decode all 16 DTMF tone pairs into 4 bit code external component count is minimized by provision of a on chip differential input amplifier, clock generator, and latched tri-state interface bus. Minimal external components required include a low cost 3.579545 MHz color burst crystal, a timing resistor, and timing capacitor

IC Part	**Output Voltage (V)**	**Minimum Vi (V)**
7805	+5	7.3
7806	+6	8.3
7808	+8	10.5
7810	+10	12.5
7812	+12	14.6
7815	+15	17.7
7818	+18	21.0
7824	+24	27.1

The new M-8870-02 provides a “power down” option which, when enabled, drops consumption too less than .5 mw. The 02 versions can also inhibit the decoding of the fourth column digits.

DTMF is the generic name for pushbutton telephone signaling equivalent to the bell system’s touch-tone. Dual tone multi frequency (DTMF) signaling is quickly replacing dial-pulse signaling in telephone banking or electronic mail systems, in which the user can select options from a menu by sending signals from a telephone.

6.2 DTMF Standards:

The DTMF tone-signaling standard is also known as touch tone or MFPB (Multi-frequency push button). Bell labs for use by AT&T in the dial-pulse-signaling standard developed touch-tone. Each administration has defined its own DTMF specifications. They are all very similar to the CCITT standard, varying by small amounts in the guard bands

Two tones are used to generate a DTMF digit. One tone is chosen out of four row tones, and the other is chosen out of four column tones. Two of eight tones can be combined so as to generate sixteen different DTMF digits. Of these sixteen keys shown in the figure, twelve are the familiar keys of a touch-tone keypad and four are reserved for future uses.

A 90-minute audiocassette tape to test DTMF decoders is available from Mitel semiconductors. There also exists a standard describing requirement for systems, which test DTMF systems. This standard is available from the IEEE as ANSI/IEEE standard 752-1986.

6.3 DTMF Decoder:

The DTMF signals transmitted over the telephone lines can be received and decoded outputs can be suitably used along with certain additional circuitry to design a DTMF code detection unit.

The DTMF digits transmitted over the telephone lines would have a nominal width of 50ms followed by a pause of similar duration between consecutive digits would be transmitted in one second. The on- cradle and off-cradle status of handset can be detected, based on the line voltage state, before the start of ringing. The voltage drops to 10 to 12V DC on lifting of the handset from the cradle. The ringing status can be detected with the use of either a capacitively coupled rectifier bridge or an AC opto-coupler or even a DC opto coupler with an external diode shunted in anti-parallel across the internal diode of the opto-coupler together with a current limiting series resistor.

The heart of our project is PC ADD-ON card designed with the help of PPI, comparator, decoder, buffer and latches. An integrated DTMF decoder type MV8870 decodes the tone dialing codes received via the telephone line. The telephone line interface consists of two parts: one to detect the ring signals that enable the unit to answer the call at the right t and another to receive and transmit tones via telephone line.

6.4 Features

- Low power consumption
- Adjustable acquisition and release times
- Central office quality and performance
- Power-down and inhibit modes
- Single 5-volt power supply
- Dial tone suppression.

APPLICATIONS

- Telephone switch equipment
- Mobile radio
- Remote control
- Remote data entry.

6.5 Functional Description

The M-8870 operating functions include a band split filter that separates the high and low tones of the received pair, and a digital decoder that verifies both the frequency and duration of the received tones before passing the resulting 4-bit code to the output bus.

FILTER

The low and high group tones are separated by applying the dual tone signal to the inputs of two 9th order switched capacitor band pass filters with bandwidths that corresponds to the bands enclosing the low and high group tones. The filter also incorporates notches at 350 and 440Hz, providing excellent dial tone rejection. A single order switched capacitor section that smoothers the signals prior to limiting follows each filter output. High gain comparator provided with hysteresis to prevent detection of unwanted low-level signals and noise performs signal limiting. The comparator outputs provide full rail logic swings at the frequencies of the incoming tones.

DECODER

The M-8870 decoder uses a digital counting technique to determine the frequencies of the limited tones and to verify that they correspond to standard DTMF frequencies. Complex averaging algorithm is used to protect against tone simulation by extraneous signals while tolerating small frequency variations the algorithm ensures an optimal combination of immunity to talk off and tolerance to interfacing signals and noise. When the detector recognizes the simultaneous presence of two valid tones, it raises the early steering flag (EST). Any subsequent loss of signal condition will cause EST to fall.

STEERING CIRCUIT

Before a decoded tone pair is registered, the receiver checks for valid signal duration. This check is performed by an external RC time constant driven by EST. Logic high on EST causes Vcc to raise as capacitor discharges. Provided that the signal condition is maintained for the validation period Vcc reaches the threshold (Vtst) of the steering logic to register the tone pair thus latching its corresponding 4-bit code into the output latch.

At this point, the GT output is activated and drives Vcc to Vdd. GT continues to drive high as long as EST remains high. Finally, after a short delay to allow the output latch to settle, the delayed steering output flag (StD) goes high, signaling that received tone pair has been registered. The contents of the output latch are made available on the four-bit output bus by raising the three-state control input (OE) to logic high.

The steering circuit works in reverse to validate the inter digit pause between signals. Thus as well as rejecting signals too short to be considered valid, the receiver will tolerate signal interruptions too short to be considered a valid pause. This capability, together with the ability to select the steering time constants externally, allows the designer to tailor performance to meet a wide variety of system requirements.

GUARD TIME ADJUSTMENT:

Where independent selections of receive and pauses are not required the simple steering circuit is applicable. Component values are chosen according to the formula:

$$\boxed{\begin{array}{l} Trec=tdp+Tgtp \\ \quad Tgtp=0.67rc \end{array}}$$

The value of tdp is a parameter of the device and Trec is the minimum signal duration to be recognized by the receiver. A value for C of 0.1μF is recommended for most applications, leaving r to be selected by the designer. For example suitable value of R for a Trec of 40ms would be 300k ohm. The timing requirements for most telecommunications are satisfied with this arrangement.

Different steering arrangements may be used to select independently the guard times for tone present (Tgtp) and tone absent (Tgta). This may be necessary to meet system specifications that place both accept and reject limits on both tone and interdigit pause.

Guard time adjustments also allow the designer to tailor system parameters such as talk off and noise immunity. Increasing to tailor system parameters such as talk off and noise immunity. Increasing trec improves talk off performance, since it reduces the probability that tones simulated by speech will maintain signal condition long enough to be registered. On the other hand a relatively short trec with a long to do would be appropriate for extremely noisy environment where acquisition time and immunity to dropouts fast would be required.

INPUT CONFIGURATION:

The input arrangement of the M-8870 provides a differential input operational amplifier as well as bias source to bias the inputs at mid rail. Provision is made for connection of a feedback resistor to the op-amp output (GS) for gain adjustments.

In a single ended configuration, the input pins are connected with the op-amp connected for unity gain and Vref biasing the input at ½ VDD. Adjustable gain configuration is possible with the help of the feedback resistor.

***CLOCK CIRCUIT*:**

The internal clock circuit is completed with the addition of a standard 3.579545mHZ color burst crystal. The crystal can be connected to a single M-8870 or to a series of M-8870's. a single crystal can be used to connect a series of M-8870 by coupling the oscillator output of each M-8870 through a 30pF capacitor to the oscillator input of the next M-8870.

PIN FUNCTIONS:

IN+: Non-inverting input connected to the front-end of the differential amplifier.

IN -: Inverting input. Connected to the front-end of the differential amplifier

GS: Gain select. It gives access to output of front-end amplifier for connection of feedback resistor.

VREF: Reference voltage output. May be used to bias the inputs at mid rail.

INH*: Inhibits detection of tones representing keys A, B,C and D

PD*: Power down. Logic high powers down the device and inhibits the oscillator

OSC1: Clock input. 3.579545MHZ crystal connected between these pins complete the internal oscillator.

OSC2: Clock input. 3.579545MHZ crystal connected between these pins complete the internal oscillator.

VSS Negative power supply (normally connected to 0v)

OE: Three state output enable (input). Logic high enables the outputs Q1, Q4. Internal pull up.

Q1, Q2: Three state outputs. When enabled by OE,

Q3, Q4: Provides the code corresponding to the last valid tone pair received.

StD : Delayed steering output. Presents logic high when a received tone pair has been registered and the output latch is updated. Returns to logic low when the voltage on St/GT falls below Vtsi.

Est : Early steering output presents logic high immediately when the digital algorithm detects a recognizable tone pair. Any momentary loss of signal condition will cause Est to return to logic low.

St/GT: Steering input/guard time output voltage greater than VTSt detected at St cause the device to register the detected tone pair and update the output latch. A voltage less than VTSt free the device to accept a new tone pair. The GT output acts to reet the external steering time constant and its state is a function of Est and the voltage on St.

VDD: Positive power supply.

7.0 M-8870DTMF RECEIVER:

The Teltone M-8870 is a full DTMF receiver that integrates both band split filter and decoder functions into a single 18-pin DIP or SOIC package. Manufactured using state-of-the –art CMOS process technology, the M-8870 offers low power consumption (35 mW max) and precise data handling. Its filter section uses switched capacitor technology for both the high and low group filters and for dial tone rejection. Its decoder uses digital counting techniques to detect and decode all 16 DTMF tone pairs into a 4-bit code. External on-chip differential input amplifier, clock generator, and latched tri-state interface bus. Minimal external components required include a low-cost 3.579545 MHz color burst crystal, a timing resistor, and a timing capacitor.

The new M-8870-02 provides a "power-down" option which, then enabled, drops consumption to less than 0.5mW. The -02 versions can also inhibit the decoding of fourth column digits.

7.1 Features:

- Low power consumption
- Adjustable acquisition and release times
- Central office quality and performance
- Power – down and inhibit modes (-0.2 version)
- Single 5 volt power supply
- Dial tone suppression

APPLICATIONS:

- Telephone switch equipment
- Mobile radio
- Remote control
- Remote data entry

7.2 **Functional Description:**

M-8870 operating functions includes a band split filter that separate the high and low tones of the received pair, and a digital decoder that verifies both the frequency and duration of the received tones before passing the resulting 4-bit code to the output bus.

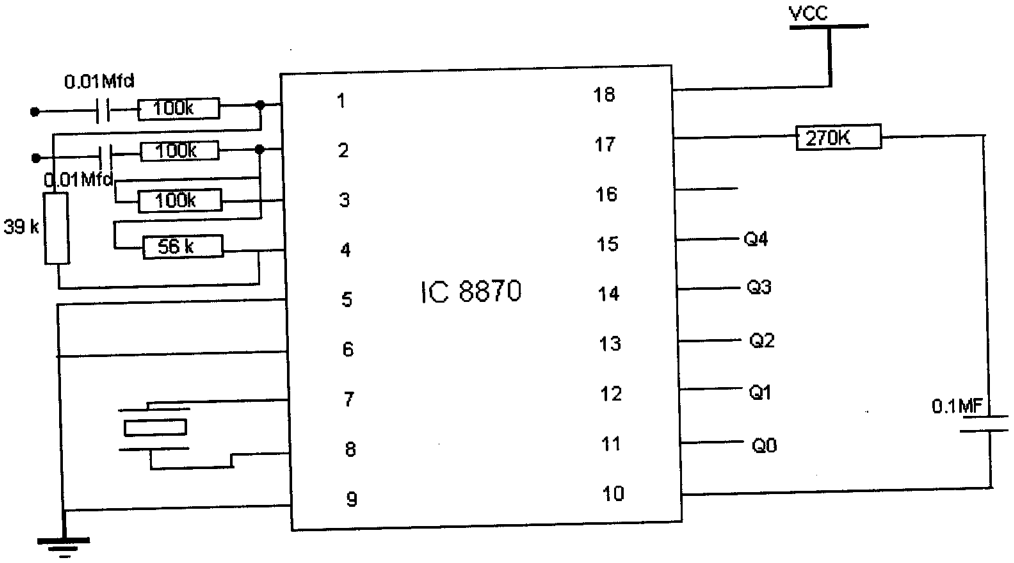
Table 5 Tone Decoding

F LOW	F HIGH	KEY (ref.)	OE	Q4	Q3	Q2	Q1

697	**1336**	**2**	**H**	**0**	**0**	**1**	**0**
697	**1477**	**3**	**H**	**0**	**0**	**1**	**1**
770	**1209**	**4**	**H**	**0**	**1**	**0**	**0**
770	**1336**	**5**	**H**	**0**	**1**	**0**	**1**
770	**1477**	**6**	**H**	**0**	**1**	**1**	**0**
852	**1209**	**7**	**H**	**0**	**1**	**1**	**1**
852	**1336**	**8**	**H**	**1**	**0**	**0**	**0**
852	**1477**	**9**	**H**	**1**	**0**	**0**	**1**
941	**1336**	**0**	**H**	**1**	**0**	**1**	**0**
941	**1209**	**.**	**H**	**1**	**0**	**1**	**1**
941	**1477**	**#**	**H**	**1**	**1**	**0**	**0**
697	**1633**	**A**	**H**	**1**	**1**	**0**	**1**
770	**1633**	**B**	**H**	**1**	**1**	**1**	**0**
852	**1633**	**C**	**H**	**1**	**1**	**1**	**1**
941	**1633**	**D**	**H**	**0**	**0**	**0**	**0**
ANY	**ANY**	**ANY**	**L**	**Z**	**Z**	**Z**	**Z**

H = High L = Low Z = High Impedance

DTMF reciver circuit



8.0 ULTRASONIC TRANSMITTER

The ultrasonic transmitter circuit consists of NAND gate.

NAND gate:

This is the 2 input NAND gate. Let the input signals be A and B and the output voltage be Y. The output will be 1 if A is a 0 or B is a 0 and the output will be 0 if A and B are 1.

The truth table of NAND gate is

Input		Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

8.1 Circuit Description:

Two NAND gates are used in this circuit. One of the inputs to the NAND gate is +5V. Consider the other input to the NAND is 0. The output will be 1. When the output is 1, the capacitor gates charged. During the next cycle both the inputs to the NAND gate is 1 and the output goes to 0. The output of the first NAND gate is given to the second NAND gate. The two inputs to the NAND gate are tied together.

The output of this second NAND gate is used for transmission using ultrasonic sensor.

9.0 ULTRASONIC RECEIVER

9.1 Introduction

The ultrasonic receiver circuit consists of

1. Inverting amplifier
2. Rectifier & Filter
3. Comparator
4. Switching circuit

INVERTING AMPLIFIER:

The inverting amplifier is the most widely used of all the circuits. The output voltage V_{OUT} is fed back to the inverting input terminal through the R_f-R_1 network where R_f is the feedback resistor. Input signal V_i is applied to the inverting input terminal through R_1 and non-inverting input terminal of op-amp is grounded.

The gain of the inverting amplifier (closed loop gain) is

$$A = V_o/V_i$$

$$= -R_f/R_1$$

The negative sign indicates a phase shift of 180^0 between V_i and V_0. The value of R_1 should be kept fairly large to avoid loading effect.

RECTIFIER & FILTER:

Diode 1N4148 is used for rectification. The diode allows only one direction of conduction through the device (forward-bias), and a reverse-bias condition will result in the open-circuit state for the device. When the voltage across the device is greater than the threshold, the diode conducts. When conduction is established the resistance of the diode will be specified. Capacitor is used for filter.

COMPARATOR:

A comparator is a circuit, which compares a signal voltage applied at one input of an op-amp with a known reference voltage at the other input. It is basically an open loop op-amp with an output $\pm V_{sat}$ (= V_{cc}). It may be seen that the change in the output state takes place with an increment in input V_1 of only 2 mV. This is the uncertainty region where output cannot be directly defined. There are basically two types of comparators:

1. Non-inverting comparator
2. Inverting comparator

In case of non-inverting comparator a fixed reference voltage V_{ref} is applied to (-) input and a time varying signal V_i is applied to (+) input. The output voltage is at $-V_{sat}$ for $V_i < V_{sat}$ and V_o goes to + V_{sat} for $V_i > V_{ref.}$

In case of inverting comparator a fixed reference voltage V_{ref} is applied to (+) input and a time varying signal V_i is applied to (-) input. The output voltage is at $+V_{sat}$ for $V_i < V_{sat}$ and V_o goes to $-V_{sat}$ for $V_i > V_{ref.}$

SWITCHING CIRCUIT:

Many solid-state devices are also used in power control applications, and the simplest of these is the discrete bipolar transistor, which is usually used in the switching mode. In the case of the NPN transistor the switch load is wired between collector and supply positive, and in the case of PNP device it is wired between collector and the 0V. In both cases the switch-driving signal is applied to base via R_1, which has a typical resistance about twenty times greater than the load resistance value.

In the NPN circuit Transistor Q1 is cut off (acting like an open switch), with its output at the positive supply voltage value, with zero input signal applied, but can be driven to saturation (so that it acts like a closed switch and passes current from collector to emitter) by applying a large positive input voltage, under which condition the output equals Q1's saturation voltage value (typically 200mV to 600 mV).

The action of the PNP circuit is the reverse of that described above, and Q1 is driven to saturation (with its output a few hundred millivolts below the supply voltage value) and passes current from emitter to collector with zero input drive voltage applied, and is cut off (with its output at zero volts) when the input equals the positive supply rail value.

9.2 Circuit Description:

At the receiving side, ultrasonic sensor is used to receive the transmitting signal. The received signal is used as input to the inverting amplifier. The amplified signal is rectified and filtered and given as input to the comparator. The other input to the comparator is the reference voltage. The comparator compares these two voltages and the output of the comparator is +12V or-12V. The output of the comparator is given to the switching circuit. Here the transistor acts as a switch. When the voltage across the B-E junction of the transistor is less than 0.7V, the transistor is in OFF condition. The output voltage taken across the collector of the transistor is +5V. When the voltage across the B-E junction of the transistor is greater than 0.7V, the transistor is in ON condition. The output voltage taken across the collector of the transistor is 0V.

10.0 RELAY CONTROL

10.1 introduction:

RELAYS:

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

When the controlling current flows through the coil, the soft iron core is magnetized and attracts the L-shaped soft iron armature. This rocks on its pivot and opens, closes or changes over, the electrical contacts in the circuit being controlled it closes the contacts.

The current needed to operate a relay is called the pull-in current and the dropout current in the coil when the relay just stops working. If the coil resistance R of a relay is 185 Ω and its operating voltage V is 12V, the pull-in current I is given by:

$$I = V/R$$

$$= 12V/185\Omega$$

$$= 0.065A = 65mA$$

10.2 Reed Switches:

Relays operate comparatively slowly and for fast switching of a signal circuit, e.g. in a telephone exchange, reed switches are used. The reeds are thin strips of easily magnetizable and demagnetizable material. They are sealed in a glass tube containing an inert gas such as nitrogen to reduce corrosion of the contacts.

The switch is operated either by bringing a magnet near or by passing a current through a coil surrounding it. In both cases reeds become magnetized attract each other and on touching they complete the circuit connected to the terminals. They separate when the magnet is removed or the current stops flowing in the coil.

When the changeover reed switch operates, the reed is attracted from the non-magnetic contact to the magnetic one. Protection of transistor-controlled relays and reed switches. When the current in the coil of a reed switch falls to zero, a large voltage is induced in the coil due to its inductance. This voltage could damage any transistor used to control the current in the coil. However if a diode is connected in reverse bias for the supply voltage it offers an easy path to the induced voltage and stops it building up to a high value.

10.3 Earth-Leakage (OR Residual Current) Circuit Breaker:

This is sometimes present as a safety device in mains electrical circuits. In one variety, current passes to earth through a relay-type 'trip coil' when for example; the metal case of the appliance becomes 'live' due to a fault. As a result the rod in the coil opens the switch, which can be set to break the circuit before the case rises above say 25V.

In our project we are using an ON-OFF control valve i.e. solenoid valve. The advantage of using ON-OFF control valve is

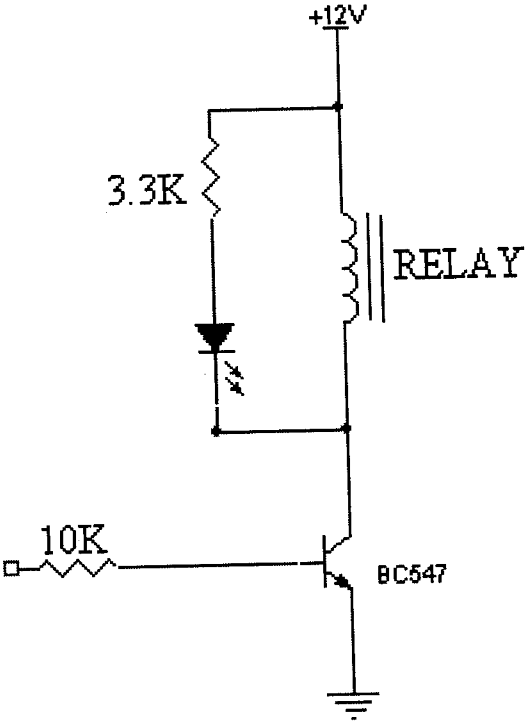
1. There is no dead time
2. There is no transfer lag.

10.4 Circuit Diagram Description:

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

1. Normally closed
2. Normally opened

We are using normally opened type relay. When the controller output from the PC is high the transistor will be in the ON state, so relay is energized. When the controller output from the PC is low the transistor will be in the OFF state, so relay is de-energized the valve will open. When the relay is de-energized the valve will close. So according to the controller output the valve will open or close and thus level is maintained.



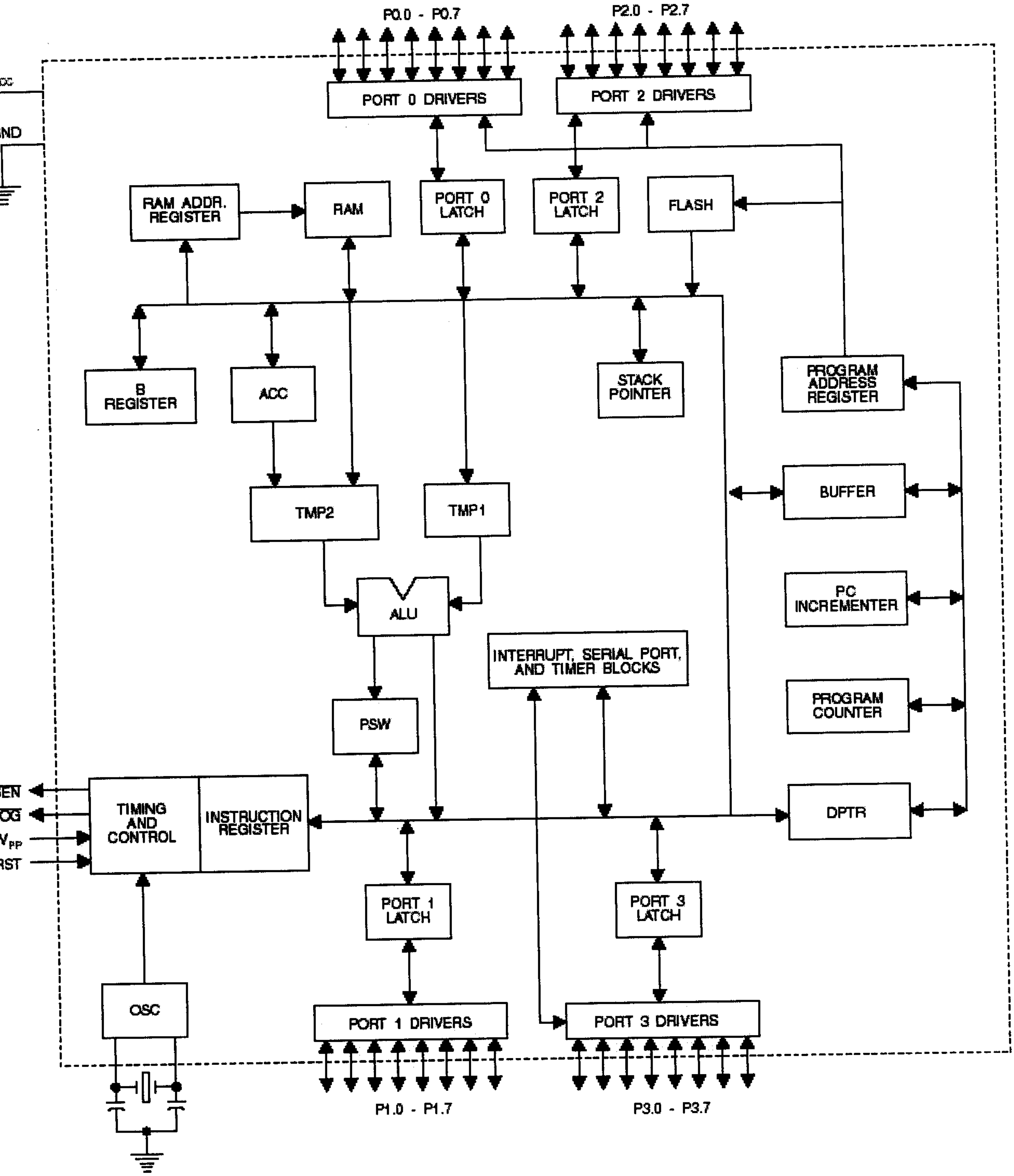
S

11.0 MICROCONTROLLER-ATMEL 89C51

11.1 Features

- Compatible with MCS-51™ Products
- 4K Bytes of In-System Reprogrammable Flash Memory– Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low-power Idle and Power-down Modes

11.2 Block diagram



11.3 Introduction

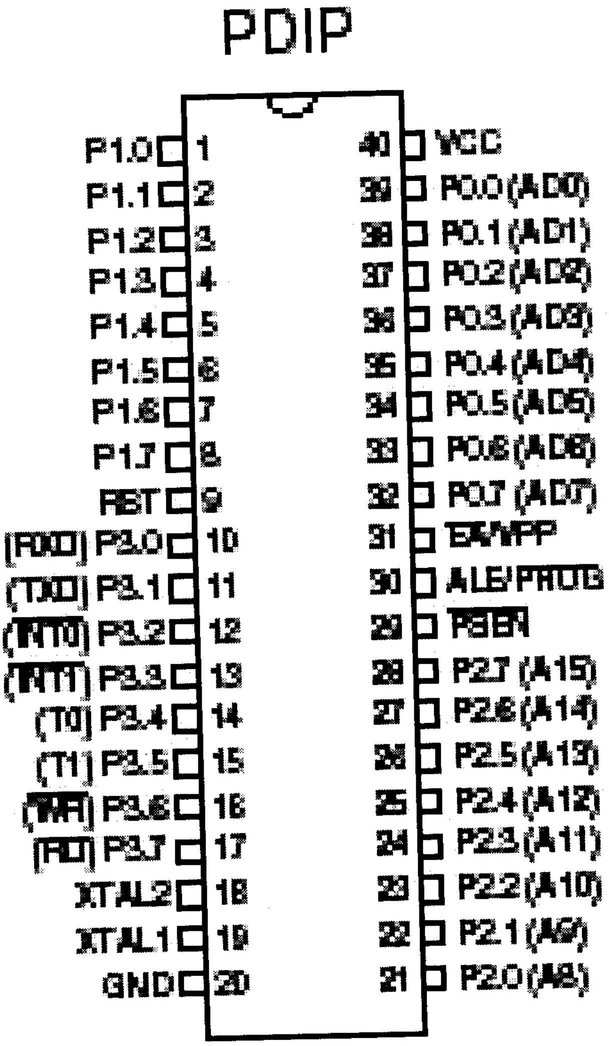
The AT89C51 provides the following standard features: 4K bytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry.

In addition, the AT89C51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power-down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

DESCRIPTION

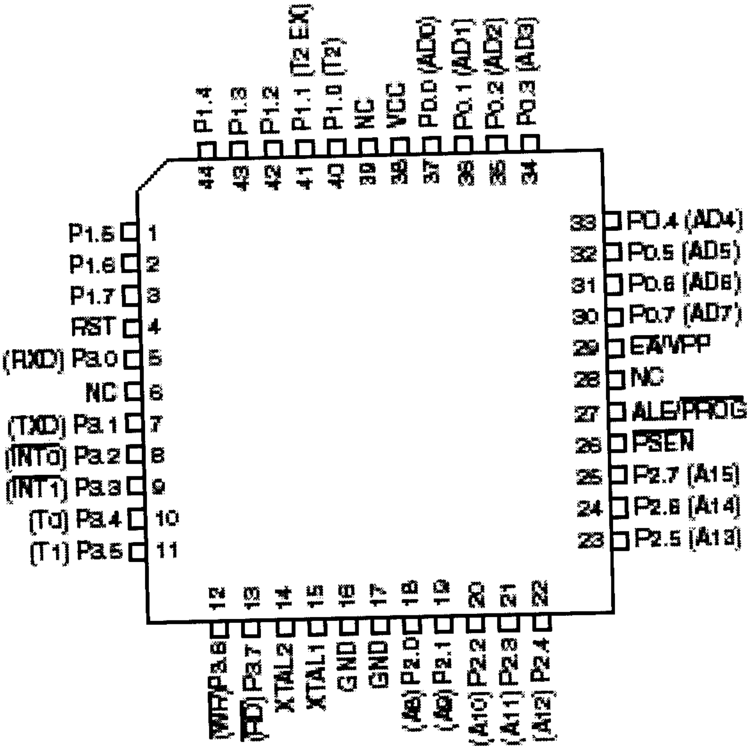
The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flashon a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications

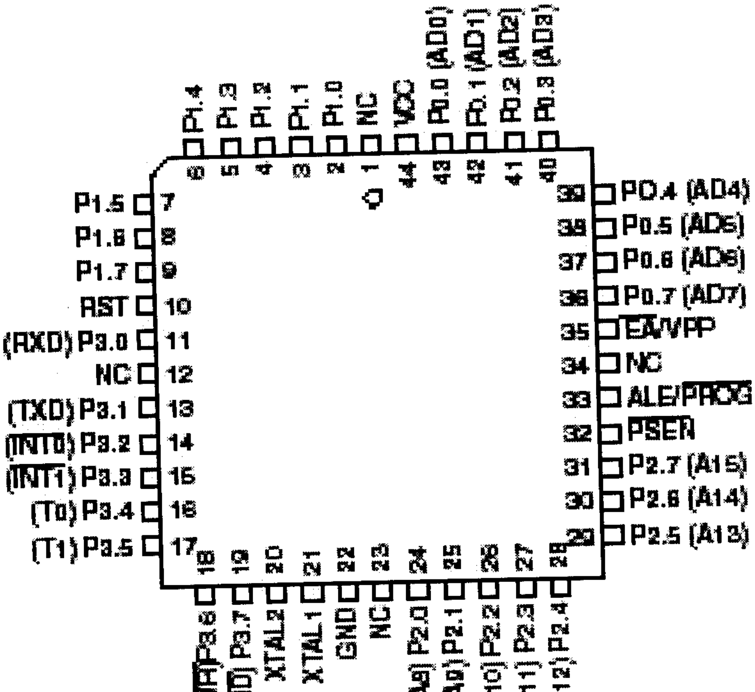
11.4 Pin Configurations



OTHER PACKAGES

PQFP/TQFP





Pin Functions:

VCC- Supply voltage.

GND- Ground.

PORT 0

Port 0 is an 8-bit open-drain bi-directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high impedance inputs. Port 0 may also be configured to be the multiplexed low order address/data bus during accesses to external program and data memory. In this mode P0 has internal pull-ups. Port 0 also receives the code bytes during Flash programming, and outputs the code bytes during program verification. External pull-ups are required during program verification.

PORT 1

Port 1 is an 8-bit bi-directional I/O port with internal pullups. The Port 1 output buffers can ink/source four TTL inputs. When 1s are written to Port 1 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL) because of the internal pullups. Port 1 also receives the low-order address bytes during Flash programming and verification.

PORT 2

Port 2 is an 8-bit bi-directional I/O port with internal pullups. The Port 2 output buffers can ink/source four TTL inputs. When 1s are written to Port 2 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (IIL) because of the internal pullups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that uses 16-bit addresses (MOVX @ DPTR). In this application, it uses strong internal pull-ups when emitting 1s. During accesses to external data memory that uses 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register. Port

2 also receives the high-order address bits and some control signals during Flash programming and verification.

PORT 3

Port 3 is an 8-bit bi-directional I/O port with internal pullups. The Port 3 output buffers can ink/source four TTL inputs. When 1s are written to Port 3 pins they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL) because of the pullups. Port 3 also serves the functions of various special features of the AT89C51 as listed below:

Port Pin	**Alternate Functions**
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	INT0 (external interrupt 0)
P3.3	INT1 (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

Port 3 also receives some control signals for Flash programming and verification. RST Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/PROG

Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input (PROG) during Flash programming.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory. If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

PSEN

Program Store Enable is the read strobe to external program memory. When the AT89C51 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

EA/VPP

External Access Enable. EA must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, EA will be internally latched on reset. EA should be strapped to VCC for internal program executions. This pin also receives the 12-volt programming enable voltage (VPP) during Flash programming, for parts that require 12-volt VPP.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

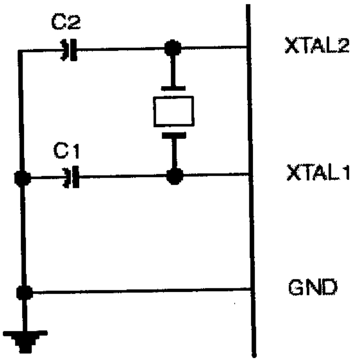
Idle Mode

In idle mode, the CPU puts itself to sleep while all the onchip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset. It should be noted that when idle is terminated by a hard ware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Status of External Pins During Idle and Power-down Modes

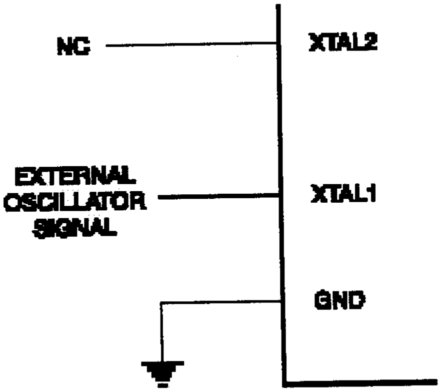
Mode	Program Memory	ALE	$\overline{\text{PSEN}}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Figure 1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



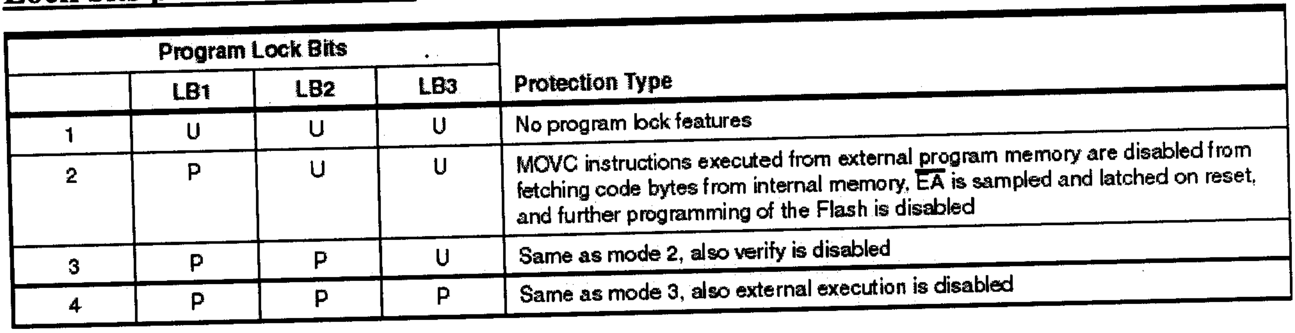
11.5 Power-down Mode

In the power-down mode, the oscillator is stopped, and the instruction that invokes power-down is the last instruction executed. The on-chip RAM and Special Function Registers Registers retain their values until the power-down mode is terminated. The only exit from power-down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before VCC is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Program Memory Lock Bits:

On the chip are three lock bits which can be left unprogrammed (U) or can be programmed the logic level at the EA pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of EA be in agreement with the current logic level at that pin in order for the device to function properly.

Lock bits protection modes:

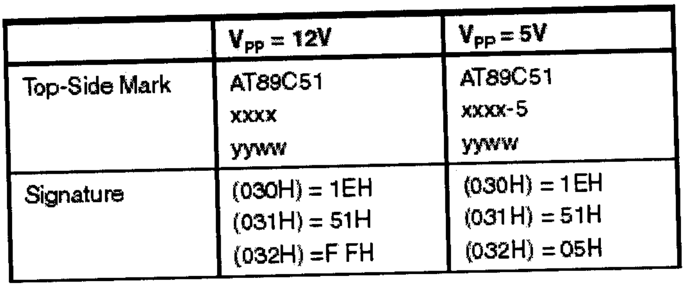


Program Lock Bits				
	LB1	LB2	LB3	Protection Type
1	U	U	U	No program bck features
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash is disabled
3	P	P	U	Same as mode 2, also verify is disabled
4	P	P	P	Same as mode 3, also external execution is disabled

11.6 Programming the Flash:

The AT89C51 is normally shipped with the on-chip Flash memory array in the erased state (that is, contents = FFH) and ready to be programmed. The programming interface accepts either a high-voltage (12-volt) or a low-voltage (VCC) program enable signal. The low-voltage programming mode provides a convenient way to program the AT89C51 inside the user's system, while the high-voltage programming mode is compatible with conventional third party Flash or EPROM programmers.

The AT89C51 is shipped with either the high-voltage or low-voltage programming mode enabled. The respective top-side marking and device signature codes are listed in the following table.



	V_{PP} = 12V	V_{PP} = 5V
Top-Side Mark	AT89C51 xxxx yyww	AT89C51 xxxx-5 yyww
Signature	(030H) = 1EH (031H) = 51H (032H) =F FH	(030H) = 1EH (031H) = 51H (032H) = 05H

The AT89C51 code memory array is programmed byte-by byte in either programming mode. To program any nonblank byte in the on-chip Flash Memory, the entire memory must be erased using the Chip Erase Mode.

11.7 Programming Algorithm:

Before programming AT89C51, the address, data and control signals should set up according to the Flash programming mode table Figure 3 and Figure 4. To program the AT89C51, take following steps.

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise EA/VPP to 12V for the high-voltage programming mode.
5. Pulse ALE/PROG once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

Data Polling: The AT89C51 features Data Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written datum on PO.7. Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. Data Polling may begin any time after a write cycle has been initiated. **Ready/Busy:** The progress of byte programming can also be monitored by the RDY/BSY output signal. P3.4 is pulled low after ALE goes high during programming to indicate BUSY. P3.4 is pulled high again when programming is done to indicate READY.

Figure 3. Programming the Flash

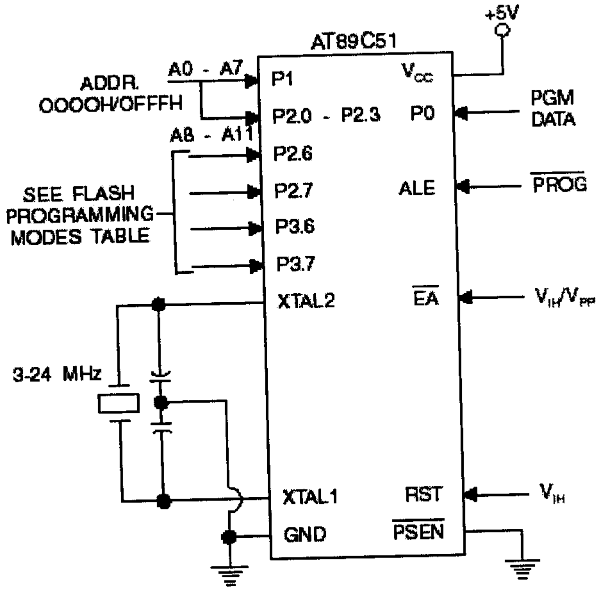
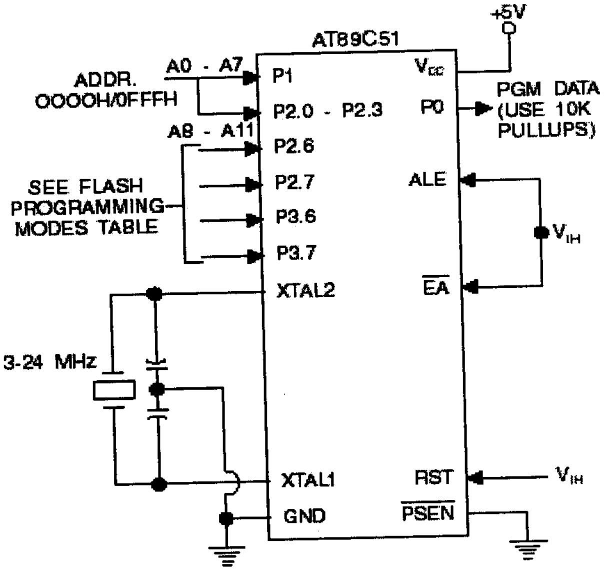


Figure 4. Verifying the Flash



Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/PROG low for 10 ms. The code array is written with all

“1”s. The chip erase operation must be executed before the code memory can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H, 031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(030H) = 1EH indicates manufactured by Atmel

(031H) = 51H indicates 89C51

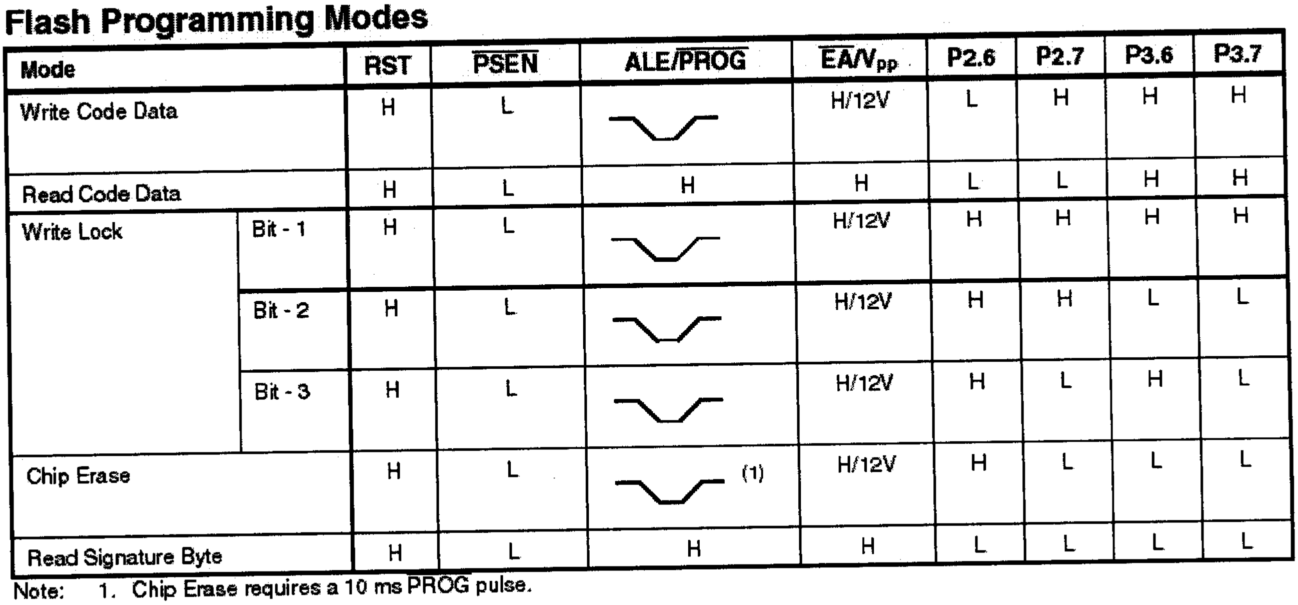
(032H) = FFH indicates 12V programming

(032H) = 05H indicates 5V programming

11.8 Programming Interface:

Every code byte in the Flash array can be written and the entire array can be erased by using the appropriate combination of control signals. The write operation cycle is selftimed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series.



Flash Programming Modes

Mode		RST	PSEN	ALE/PROG	EA/V_{pp}	P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L	⎺╲_╱⎺	H/12V	L	H	H	H
Read Code Data		H	L	H	H	L	L	H	H
Write Lock	Bit - 1	H	L	⎺╲_╱⎺	H/12V	H	H	H	H
	Bit - 2	H	L	⎺╲_╱⎺	H/12V	H	H	L	L
	Bit - 3	H	L	⎺╲_╱⎺	H/12V	H	L	H	L
Chip Erase		H	L	⎺╲_╱⎺ (1)	H/12V	H	L	L	L
Read Signature Byte		H	L	H	H	L	L	L	L

Note: 1. Chip Erase requires a 10 ms PROG pulse.

11.8 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 are not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.;

PROGRAMMING CODE

12.0 PROGRAMMING CODES:

dtmf_9key.SRC generated from: dtmf_9key.c

```
$NOMOD51
NAMEDTMF_9KEY

P0      DATA 080H
P1      DATA 090H
P2      DATA 0A0H

P3      DATA 0B0H
T0      BIT     0B0H.4
AC      BIT     0D0H.6
T1      BIT     0B0H.5

EA      BIT     0A8H.7
ndata   BIT     0B0H.4
IE      DATA 0A8H
RD      BIT     0B0H.7
ES      BIT     0A8H.4
IP      DATA 0B8H
RI      BIT     098H.0
INT0    BIT     0B0H.2
CY      BIT     0D0H.7

TI      BIT     098H.1
INT1    BIT     0B0H.3
relay   BIT     0B0H.6
PS      BIT     0B8H.4
SP      DATA 081H
OV      BIT     0D0H.2
WR      BIT     0B0H.6
SBUF  DATA 099H

PCON DATA 087H
SCON DATA 098H
TMODDATA 089H
TCON DATA 088H
IE0     BIT     088H.1
IE1     BIT     088H.3
B       DATA 0F0H
ACC     DATA 0E0H

ET0     BIT     0A8H.1
ET1     BIT     0A8H.3
```

```
TF1    BIT    088H.7
RB8    BIT    098H.2
TH0    DATA 08CH
EX0    BIT    0A8H.0
IT0    BIT    088H.0
TH1    DATA 08DH
TB8    BIT    098H.3
EX1    BIT    0A8H.2
IT1    BIT    088H.2
P      BIT    0D0H.0

SM0    BIT    098H.7
TL0    DATA 08AH
SM1    BIT    098H.6
TL1    DATA 08BH
SM2    BIT    098H.5
PT0    BIT    0B8H.1
PT1    BIT    0B8H.3
RS0    BIT    0D0H.3
TR0    BIT    088H.4
RS1    BIT    0D0H.4
TR1    BIT    088H.6
PX0    BIT    0B8H.0
PX1    BIT    0B8H.2
DPH    DATA 083H
DPL    DATA 082H
REN    BIT    098H.4
RXD    BIT    0B0H.0
TXD    BIT    0B0H.1
F0     BIT    0D0H.5
object BIT    0B0H.5
PSW    DATA 0D0H

?PR?main?DTMF_9KEY   SEGMENT CODE
?PR?timer0?DTMF_9KEY SEGMENT CODE

?PR?timer1?DTMF_9KEY SEGMENT CODE
?PR?obj?DTMF_9KEY    SEGMENT CODE
?PR?delay?DTMF_9KEY  SEGMENT CODE

?C_INITSEG         SEGMENT CODE
?DT?DTMF_9KEY         SEGMENT DATA
        EXTRN        CODE (?C_STARTUP)
        PUBLIC       t
        PUBLIC       s
        PUBLIC       r
        PUBLIC       reg1
        PUBLIC       l
        PUBLIC       k
```

```
    PUBLIC      i
    PUBLIC      g
    PUBLIC      dtmf
    PUBLIC      delay
    PUBLIC      obj
    PUBLIC      timer1
    PUBLIC      timer0
    PUBLIC      main

    RSEG  ?DT?DTMF_9KEY
    dtmf:  DS   1
      g:  DS   1
      i:  DS   1
      j:  DS   1
      k:  DS   2
      l:  DS   1
    reg1:  DS   1
      r:  DS   1
      s:  DS   1
      t:  DS   1

     RSEG  ?C_INITSEG
     DB     001H
     DB     l
     DB     000H

     DB     001H
     DB     reg1
     DB     000H

     DB     001H
     DB     g
     DB     000H

; #pragma src;
; #include<reg51.h>
; void del();
; void delay();
; void obj();
; void fir();
; sbit ndata=P3^4;
; sbit object=P3^5;
; sbit relay=P3^6;
; unsigned char i,j,r,s,t,dtmf,l=0,reg1=0,g=0;
; unsigned int k;
; main()

        RSEG  ?PR?main?DTMF_9KEY
main:
```

```
                        ; SOURCE LINE # 12
; {
                        ; SOURCE LINE # 13
; ndata=1;
                        ; SOURCE LINE # 14
        SETB  ndata
; object=1;
                        ; SOURCE LINE # 15
        SETB  object
; relay=0;
                        ; SOURCE LINE # 16
        CLR   relay
; delay();
                        ; SOURCE LINE # 17
        LCALL           delay
; EA=1;
                        ; SOURCE LINE # 18
        SETB  EA
; TH0=0x00;
                        ; SOURCE LINE # 19
        CLR   A
        MOV   TH0,A
; TL0=0x00;
                        ; SOURCE LINE # 20
        MOV   TL0,A
; TH1=0x00;
                        ; SOURCE LINE # 21
        MOV   TH1,A
; TL1=0x00;
                        ; SOURCE LINE # 22
        MOV   TL1,A
; TMOD=0x11;
                        ; SOURCE LINE # 23
        MOV   TMOD,#011H
; ET0=1;
                        ; SOURCE LINE # 24
        SETB  ET0
; ET1=1;
                        ; SOURCE LINE # 25
        SETB  ET1
; TR0=1;
                        ; SOURCE LINE # 26
        SETB  TR0
; TR1=1;
                        ; SOURCE LINE # 27
        SETB  TR1
; while(1)
                        ; SOURCE LINE # 28
; lw:
                        ; SOURCE LINE # 30
?main?lw:
; if(ndata==0)
                        ; SOURCE LINE # 31
        JNB    ndata,?main?lw
; {goto lw;}
                        ; SOURCE LINE # 32
; le:
                        ; SOURCE LINE # 33
?main?le:
; if(ndata==1)
                        ; SOURCE LINE # 34
        JB     ndata,?main?le
; {goto le;}
                        ; SOURCE LINE # 35
?C0006:
; dtmf=P3&0x0f;
                        ; SOURCE LINE # 36
        MOV  A,P3
        ANL  A,#0FH
        MOV  dtmf,A
; delay();
                        ; SOURCE LINE # 37
        LCALL       delay
; ser:
                        ; SOURCE LINE # 38
?main?ser:
;  if(dtmf==0x04)
                        ; SOURCE LINE # 39
        MOV  A,dtmf
        XRL  A,#04H
        JNZ  ?C0008
;  {
                        ; SOURCE LINE # 40
;   for(k=0;k<=10000;k++)
                        ; SOURCE LINE # 41
        MOV  k,A
        MOV  k+01H,A
?C0009:
        SETB C
        MOV  A,k+01H
        SUBB A,#010H
        MOV  A,k
        SUBB A,#027H
        JNC  ?C0008
;  {
                        ; SOURCE LINE # 42
                        ; SOURCE LINE # 43
        MOV   P2,#055H
;  delay();
                        ; SOURCE LINE # 44
        LCALL         delay
;  if(dtmf!=0x04)
                        ; SOURCE LINE # 45
        MOV   A,dtmf
        CJNE  A,#04H,?main?ser
;  {
                        ; SOURCE LINE # 46
;    goto ser;
                        ; SOURCE LINE # 47
;  }
                        ; SOURCE LINE # 48
?C0012:
;  P2=0x99;
                        ; SOURCE LINE # 49
        MOV   P2,#099H
;  delay();
                        ; SOURCE LINE # 50
        LCALL         delay
;  if(dtmf!=0x04)
                        ; SOURCE LINE # 51
        MOV   A,dtmf
        CJNE  A,#04H,?main?ser
;  {
                        ; SOURCE LINE # 52
;    goto ser;
                        ; SOURCE LINE # 53
;  }
                        ; SOURCE LINE # 54
?C0013:
;  P2=0xaa;
                        ; SOURCE LINE # 55
        MOV   P2,#0AAH
;  delay();
                        ; SOURCE LINE # 56
        LCALL         delay
;  if(dtmf!=0x04)
                        ; SOURCE LINE # 57
        MOV   A,dtmf
        CJNE  A,#04H,?main?ser
;  {
                        ; SOURCE LINE # 58
;    goto ser;
                        ; SOURCE LINE # 59
;  }
                        ; SOURCE LINE # 60
;  P2=0x66;
                                ; SOURCE LINE # 61
        MOV  P2,#066H
;  delay();
                                ; SOURCE LINE # 62
        LCALL       delay
;  if(dtmf!=0x04)
                                ; SOURCE LINE # 63
        MOV  A,dtmf
        CJNE  A,#04H,?main?ser
;  {
                                ; SOURCE LINE # 64
;    goto ser;
                                ; SOURCE LINE # 65
;  }
                                ; SOURCE LINE # 66
; }
                                ; SOURCE LINE # 67
?C0011:
        INC     k+01H
        MOV   A,k+01H
        JNZ     ?C0009
        INC     k
?C0081:
        SJMP  ?C0009
; }
                                ; SOURCE LINE # 68
?C0008:
;
; if(dtmf==0x05)
                                ; SOURCE LINE # 70
        MOV   A,dtmf
        XRL    A,#05H
        JNZ     ?C0016
; {
                                ; SOURCE LINE # 71
; for(k=0;k<=10000;k++)
                                ; SOURCE LINE # 72
        MOV   k,A
        MOV   k+01H,A
?C0017:

        SETB  C
        MOV   A,k+01H
        SUBB  A,#010H
        MOV   A,k
        SUBB  A,#027H
        JNC    ?C0016
```

```
; P2=0x66;
                         ; SOURCE LINE # 74
      MOV  P2,#066H
; delay();
                         ; SOURCE LINE # 75
      LCALL        delay
; if(dtmf!=0x05)
                         ; SOURCE LINE # 76
      MOV  A,dtmf
      XRL  A,#05H
      JNZ  ?main?ser
; {
                         ; SOURCE LINE # 77
;  goto ser;
                         ; SOURCE LINE # 78
; }
                         ; SOURCE LINE # 79
?C0020:
; P2=0xaa;
                         ; SOURCE LINE # 80
      MOV  P2,#0AAH
; delay();
                         ; SOURCE LINE # 81
      LCALL        delay
; if(dtmf!=0x05)
                         ; SOURCE LINE # 82
      MOV  A,dtmf
      XRL  A,#05H
      JNZ  ?main?ser
; {
                         ; SOURCE LINE # 83
;  goto ser;
                         ; SOURCE LINE # 84
; }
                         ; SOURCE LINE # 85
?C0021:
; P2=0x99;
                         ; SOURCE LINE # 86
      MOV  P2,#099H
; delay();
                         ; SOURCE LINE # 87
      LCALL        delay
; if(dtmf!=0x05)
                         ; SOURCE LINE # 88
      MOV  A,dtmf
      XRL  A,#05H
      JZ   $ + 5H
      LJMP ?main?ser
```

```
;   goto ser;
                         ; SOURCE LINE # 90
; }
                         ; SOURCE LINE # 91
?C0022:
; P2=0x55;
                         ; SOURCE LINE # 92
        MOV  P2,#055H
; delay();
                         ; SOURCE LINE # 93
        LCALL        delay
; if(dtmf!=0x05)
                         ; SOURCE LINE # 94
        MOV  A,dtmf
        XRL  A,#05H
        JZ   $ + 5H
        LJMP ?main?ser
; {
                         ; SOURCE LINE # 95
;   goto ser;
                         ; SOURCE LINE # 96
; }
                         ; SOURCE LINE # 97
; }
                         ; SOURCE LINE # 98
?C0019:
        INC  k+01H
        MOV  A,k+01H
        JNZ  ?C0017
        INC  k
?C0082:
        SJMP ?C0017
; }
                         ; SOURCE LINE # 99
?C0016:
;
; if(dtmf==0x06)
                         ; SOURCE LINE # 101
        MOV  A,dtmf
        CJNE A,#06H,?C0024
; {
                         ; SOURCE LINE # 102
;  P1=0x00;
                         ; SOURCE LINE # 103
        CLR  A
        MOV  P1,A
;  P2=0x00;
                         ; SOURCE LINE # 104
```

```
                        ; SOURCE LINE # 105
        LJMP  ?main?ser
; }
                        ; SOURCE LINE # 106
?C0024:
;
; if(dtmf==0x07)
                        ; SOURCE LINE # 108
        MOV   A,dtmf
        XRL   A,#07H
        JNZ   ?C0025
; {
                        ; SOURCE LINE # 109
; for(k=0;k<=10000;k++)
                        ; SOURCE LINE # 110
        MOV   k,A
        MOV   k+01H,A
?C0026:
        SETB  C
        MOV   A,k+01H
        SUBB  A,#010H
        MOV   A,k
        SUBB  A,#027H
        JNC   ?C0025
; {
                        ; SOURCE LINE # 111
; P2=0x50;
                        ; SOURCE LINE # 112
        MOV   P2,#050H
; delay();
                        ; SOURCE LINE # 113
        LCALL       delay
; if(dtmf!=0x07)
                        ; SOURCE LINE # 114
        MOV   A,dtmf
        XRL   A,#07H
        JZ    $ + 5H
        LJMP  ?main?ser
; {
                        ; SOURCE LINE # 115
;  goto ser;
                        ; SOURCE LINE # 116
; }
                        ; SOURCE LINE # 117
?C0029:
; P2=0x90;
                        ; SOURCE LINE # 118
        MOV   P2,#090H
; delay();
```

```
        LCALL       delay
; if(dtmf!=0x07)
                        ; SOURCE LINE # 120
        MOV  A,dtmf
        XRL  A,#07H
        JZ   $ + 5H
        LJMP ?main?ser
; {
                        ; SOURCE LINE # 121
;   goto ser;
                        ; SOURCE LINE # 122
; }
                        ; SOURCE LINE # 123
?C0030:
; P2=0xa0;
                        ; SOURCE LINE # 124
        MOV  P2,#0A0H
; delay();
                        ; SOURCE LINE # 125
        LCALL       delay
; if(dtmf!=0x07)
                        ; SOURCE LINE # 126
        MOV  A,dtmf


        XRL  A,#07H
        JZ   $ + 5H
        LJMP ?main?ser
; {
                        ; SOURCE LINE # 127
;   goto ser;
                        ; SOURCE LINE # 128
; }
                        ; SOURCE LINE # 129
?C0031:
; P2=0x60;
                        ; SOURCE LINE # 130
        MOV  P2,#060H
; delay();
                        ; SOURCE LINE # 131
        LCALL       delay
; if(dtmf!=0x07)
                        ; SOURCE LINE # 132
        XRL   A,#07H
        JZ    $ + 5H
        LJMP  ?main?ser
; {
                        ; SOURCE LINE # 133
;   goto ser;
                        ; SOURCE LINE # 134
; }
                        ; SOURCE LINE # 135
; }
                        ; SOURCE LINE # 136
?C0028:
        INC   k+01H
        MOV   A,k+01H
        JNZ   ?C0026
        INC   k
?C0083:
        SJMP  ?C0026
; }
                        ; SOURCE LINE # 137
?C0025:
;
; if(dtmf==0x08)
                        ; SOURCE LINE # 139
        MOV   A,dtmf
        XRL   A,#08H
        JNZ   ?C0033
; {
                        ; SOURCE LINE # 140
; for(k=0;k<=10000;k++)
                        ; SOURCE LINE # 141
        MOV   k,A
        MOV   k+01H,A
?C0034:
        SETB  C
        MOV   A,k+01H
        SUBB  A,#010H
        MOV   A,k
        SUBB  A,#027H
        JNC   ?C0033
; {
                        ; SOURCE LINE # 142
; P2=0x05;
                        ; SOURCE LINE # 143
        MOV   P2,#05H
; delay();
                        ; SOURCE LINE # 144
        LCALL         delay
        MOV   A,dtmf
        XRL   A,#08H
        JZ    $ + 5H
        LJMP  ?main?ser
; {
                        ; SOURCE LINE # 146
;   goto ser;
                        ; SOURCE LINE # 147
; }
                        ; SOURCE LINE # 148
?C0037:
; P2=0x09;
                        ; SOURCE LINE # 149
        MOV   P2,#09H
; delay();
                        ; SOURCE LINE # 150
        LCALL           delay
; if(dtmf!=0x08)
                        ; SOURCE LINE # 151
        MOV   A,dtmf
        XRL   A,#08H
        JZ    $ + 5H
        LJMP  ?main?ser
; {
                        ; SOURCE LINE # 152
;   goto ser;
                        ; SOURCE LINE # 153
; }
                        ; SOURCE LINE # 154
?C0038:
; P2=0x0a;
                        ; SOURCE LINE # 155
        MOV   P2,#0AH
; delay();
                        ; SOURCE LINE # 156
        LCALL           delay
; if(dtmf!=0x08)
                        ; SOURCE LINE # 157
        MOV   A,dtmf
        XRL   A,#08H
        JZ    $ + 5H
        LJMP  ?main?ser
; {
                        ; SOURCE LINE # 158
;   goto ser;
                        ; SOURCE LINE # 159
; }
                        ; SOURCE LINE # 160
                        ; SOURCE LINE # 161
        MOV  P2,#06H
; delay();
                        ; SOURCE LINE # 162
        LCALL       delay
; if(dtmf!=0x08)
                        ; SOURCE LINE # 163
        MOV  A,dtmf
        XRL  A,#08H
        JZ   $ + 5H
        LJMP ?main?ser
; {
                        ; SOURCE LINE # 164
;  goto ser;
                        ; SOURCE LINE # 165
; }
                        ; SOURCE LINE # 166
; }
                        ; SOURCE LINE # 167
?C0036:
        INC  k+01H
        MOV  A,k+01H
        JNZ  ?C0034
        INC  k
?C0084:
        SJMP ?C0034



; }
                        ; SOURCE LINE # 168
?C0033:
;
; if(dtmf==0x09)
                        ; SOURCE LINE # 170
        MOV  A,dtmf
        XRL  A,#09H
        JNZ  ?C0041
; {
                        ; SOURCE LINE # 171
;  for(k=0;k<=10000;k++)
                        ; SOURCE LINE # 172
        MOV  k,A
        MOV  k+01H,A
?C0042:
        SETB C
        MOV  A,k+01H
        SUBB A,#010H
        MOV  A,k
```

```
      JNC    ?C0041
;  {
                          ; SOURCE LINE # 173
;  P1=0x05;
                          ; SOURCE LINE # 174
      MOV   P1,#05H
;  delay();
                          ; SOURCE LINE # 175
      LCALL        delay
;  if(dtmf!=0x09)
                          ; SOURCE LINE # 176
      MOV   A,dtmf
      XRL   A,#09H
      JZ    $ + 5H
      LJMP  ?main?ser
;  {
                          ; SOURCE LINE # 177
;    goto ser;
                          ; SOURCE LINE # 178
;  }
                          ; SOURCE LINE # 179
?C0045:
;  P1=0x09;
                          ; SOURCE LINE # 180
      MOV   P1,#09H
;  delay();
                          ; SOURCE LINE # 181
      LCALL        delay
;  if(dtmf!=0x09)
                          ; SOURCE LINE # 182
      MOV   A,dtmf
      XRL   A,#09H
      JZ    $ + 5H
      LJMP  ?main?ser
;  {
                          ; SOURCE LINE # 183
;    goto ser;
                          ; SOURCE LINE # 184
;  }
                          ; SOURCE LINE # 185
?C0046:
;  P1=0x0a;
                          ; SOURCE LINE # 186
      MOV   P1,#0AH
;  delay();
                          ; SOURCE LINE # 187
      LCALL        delay
;  if(dtmf!=0x09)
                          ; SOURCE LINE # 188
```

```
        XRL   A,#09H
        JZ    $ + 5H
        LJMP  ?main?ser
;  {
                          ; SOURCE LINE # 189
;   goto ser;
                          ; SOURCE LINE # 190
;  }
                          ; SOURCE LINE # 191
?C0047:
;  P1=0x06;
                          ; SOURCE LINE # 192
        MOV   P1,#06H
;  delay();
                          ; SOURCE LINE # 193
        LCALL        delay
;  if(dtmf!=0x09)
                          ; SOURCE LINE # 194
        MOV   A,dtmf
        XRL   A,#09H
        JZ    $ + 5H
        LJMP  ?main?ser
;  {
                          ; SOURCE LINE # 195
;   goto ser;
                          ; SOURCE LINE # 196
;  }
                          ; SOURCE LINE # 197
; }


                          ; SOURCE LINE # 198
?C0044:
        INC   k+01H
        MOV   A,k+01H
        JNZ   ?C0042
        INC   k
?C0085:
        SJMP  ?C0042
; }

                          ; SOURCE LINE # 199
?C0041:
;
;  if(dtmf==0x0b)
                          ; SOURCE LINE # 201
        MOV   A,dtmf
        XRL   A,#0BH
        JNZ   ?C0049
```

```
                        ; SOURCE LINE # 202
;  for(k=0;k<=10000;k++)
                        ; SOURCE LINE # 203
        MOV  k,A
        MOV  k+01H,A
?C0050:
        SETB C
        MOV  A,k+01H
        SUBB A,#010H
        MOV  A,k
        SUBB A,#027H
        JNC  ?C0049
 ;  {
                        ; SOURCE LINE # 204
 ;  P1=0x06;
                        ; SOURCE LINE # 205
        MOV  P1,#06H
 ;  delay();
                        ; SOURCE LINE # 206
        LCALL       delay
 ;  if(dtmf!=0x0b)
                        ; SOURCE LINE # 207
        MOV  A,dtmf
        XRL  A,#0BH
        JZ   $ + 5H
        LJMP ?main?ser
 ;  {
                        ; SOURCE LINE # 208
 ;   goto ser;

        ; SOURCE LINE # 209
 ;  }

                ; SOURCE LINE # 210
 ?C0053:
 ;  P1=0x0a;
                        ; SOURCE LINE # 211
        MOV  P1,#0AH
 ;  delay();
                        ; SOURCE LINE # 212
        LCALL       delay
 ;  if(dtmf!=0x0b)
                        ; SOURCE LINE # 213
        MOV  A,dtmf
        XRL  A,#0BH
        JZ   $ + 5H
        LJMP ?main?ser
 ;  {
```

```
                        ; SOURCE LINE # 214
;   goto ser;
                        ; SOURCE LINE # 215
;  }
                        ; SOURCE LINE # 216
?C0054:
;  P1=0x09;
                        ; SOURCE LINE # 217
        MOV   P1,#09H
;  delay();
                        ; SOURCE LINE # 218
        LCALL       delay
;  if(dtmf!=0x0b)
                        ; SOURCE LINE # 219
        MOV   A,dtmf
        XRL   A,#0BH
        JZ    $ + 5H
        LJMP  ?main?ser
;  {
                        ; SOURCE LINE # 220
;   goto ser;
                        ; SOURCE LINE # 221
;  }
                        ; SOURCE LINE # 222
?C0055:
;  P1=0x05;


                        ; SOURCE LINE # 223
        MOV   P1,#05H
;  delay();
                        ; SOURCE LINE # 224
        LCALL       delay
;  if(dtmf!=0x0b)
                        ; SOURCE LINE # 225
        MOV   A,dtmf
        XRL   A,#0BH
        JZ    $ + 5H
        LJMP  ?main?ser
;  {
                        ; SOURCE LINE # 226
;   goto ser;
                        ; SOURCE LINE # 227
;  }
                        ; SOURCE LINE # 228
; }
                        ; SOURCE LINE # 229
?C0052:
        INC   k+01H
        JNZ     ?C0050
        INC     k
?C0086:
        SJMP  ?C0050
; }
                            ; SOURCE LINE # 230
?C0049:
;
; if(dtmf==0x0a)
                            ; SOURCE LINE # 232
        MOV   A,dtmf
        CJNE  A,#0AH,?C0057
; {
                            ; SOURCE LINE # 233
;   relay=1;
                            ; SOURCE LINE # 234
        SETB  relay
;   goto ser;
                            ; SOURCE LINE # 235
        LJMP  ?main?ser
; }
                            ; SOURCE LINE # 236
?C0057:
;
; if(dtmf==0x0c)
                            ; SOURCE LINE # 238
        MOV   A,dtmf
        XRL   A,#0CH
        JZ    $ + 5H
        LJMP  ?main?lw
; {
                            ; SOURCE LINE # 239
;   relay=0;
                            ; SOURCE LINE # 240
        CLR    relay
;   goto ser;
                            ; SOURCE LINE # 241
        LJMP  ?main?ser
; END OF main

CSEG AT     0000BH
        LJMP  timer0


; }
;
; }
; }
;
    void timer0(void) interrupt 1
```

```
        RSEG  ?PR?timer0?DTMF_9KEY
        USING        0
timer0:
        PUSH  ACC
        PUSH  PSW
                              ; SOURCE LINE # 247
; {
;  TR0=0;
                              ; SOURCE LINE # 249
        CLR    TR0
;  if(l==0)
                              ; SOURCE LINE # 250
        MOV   A,l
        JNZ    ?C0060
;  {
                              ; SOURCE LINE # 251
;   if(ndata==0)
                              ; SOURCE LINE # 252
        JB       ndata,?C0060
;   {
                              ; SOURCE LINE # 253
;    l=1;
                              ; SOURCE LINE # 254
        MOV   l,#01H
;    goto retr;
                              ; SOURCE LINE # 255
        SJMP  ?timer0?retr
;   }

                              ; SOURCE LINE # 256

;  }
                              ; SOURCE LINE # 257
?C0060:
;  if(l==1)
                              ; SOURCE LINE # 258
        MOV   A,l
        CJNE  A,#01H,?timer0?retr
;  {
                              ; SOURCE LINE # 259
;   if(ndata==1)
                              ; SOURCE LINE # 260
        JNB    ndata,?timer0?retr
;   {
                              ; SOURCE LINE # 261
;    l=0;
                              ; SOURCE LINE # 262
        MOV   l,#00H
;    dtmf=P3&0x0f;
                              ; SOURCE LINE # 263
        ANL   A,#0FH
        MOV   dtmf,A
;   goto retr;
                          ; SOURCE LINE # 264
;  }
                          ; SOURCE LINE # 265
; }
                          ; SOURCE LINE # 266
; retr:
                          ; SOURCE LINE # 267
?timer0?retr:
; TH0=0x00;
                          ; SOURCE LINE # 268
        MOV   TH0,#00H
; TL0=0x00;
                          ; SOURCE LINE # 269
        MOV   TL0,#00H
; TR0=1;
                          ; SOURCE LINE # 270
        SETB  TR0
;}
                          ; SOURCE LINE # 271
        POP   PSW
        POP   ACC
        RETI
; END OF timer0

CSEG AT      0001BH
        LJMP  timer1


;
; void timer1(void) interrupt 3



        RSEG  ?PR?timer1?DTMF_9KEY
        USING        0
timer1:
        PUSH  ACC
        PUSH  B
        PUSH  DPH

        PUSH  DPL
        PUSH  PSW
        MOV   PSW,#00H
        PUSH  AR0
        PUSH  AR1
        PUSH  AR2
```

```
      PUSH  AR4
      PUSH  AR5
      PUSH  AR6
      PUSH  AR7
      USING        0
                   ; SOURCE LINE # 273
; {
;  TR1=0;
                   ; SOURCE LINE # 275
      CLR   TR1
;  obj();
                   ; SOURCE LINE # 276
      LCALL        obj
;  if(reg1==2)
                   ; SOURCE LINE # 277
      MOV   A,reg1
      CJNE  A,#02H,?C0066
;  {
                   ; SOURCE LINE # 278
;   reg1=0;
                   ; SOURCE LINE # 279
      MOV   reg1,#00H
;   P1=0x00;


                   ; SOURCE LINE # 280
      MOV   P1,#00H
;   P2=0x00;
                   ; SOURCE LINE # 281
      MOV   P2,#00H
;   dtmf=0x06;
                   ; SOURCE LINE # 282
      MOV   dtmf,#06H
;  }
                   ; SOURCE LINE # 283
?C0066:
;  TH1=0x00;
                   ; SOURCE LINE # 284
      MOV   TH1,#00H
;  TL1=0x00;
                   ; SOURCE LINE # 285
      MOV   TL1,#00H
;  TR1=1;
                   ; SOURCE LINE # 286
      SETB  TR1
; }

                   ; SOURCE LINE # 287
      POP   AR7
```

```
        POP    AR5
        POP    AR4
        POP    AR3
        POP    AR2
        POP    AR1
        POP    AR0
        POP    PSW
        POP    DPL
        POP    DPH
        POP    B
        POP    ACC
        RETI
; END OF timer1

;

; void obj()

          RSEG  ?PR?obj?DTMF_9KEY



 obj:
                           ; SOURCE LINE # 289
 ; {
                           ; SOURCE LINE # 290
 ;  if(g==0)
                           ; SOURCE LINE # 291
          MOV   A,g
          JNZ     ?C0068

  ;  {
                           ; SOURCE LINE # 292
  ;   if(object==1)
                           ; SOURCE LINE # 293
           JNB     object,?C0068
  ;   {


                           ; SOURCE LINE # 294
  ;    g=1;
                           ; SOURCE LINE # 295
            MOV   g,#01H
  ;    goto retr;
                           ; SOURCE LINE # 296
            RET
  ;   }
                           ; SOURCE LINE # 297
```

```
?C0068:
; if(g==1)
                          ; SOURCE LINE # 299
        MOV   A,g
        CJNE  A,#01H,?C0073
; {
                          ; SOURCE LINE # 300
;  if(object==0)
                          ; SOURCE LINE # 301
        JB      object,?C0073
;  {

                          ; SOURCE LINE # 302
;   g=0;
                          ; SOURCE LINE # 303
        CLR   A
        MOV   g,A
;   reg1++;
                          ; SOURCE LINE # 304
        INC    reg1
;   goto retr;
                          ; SOURCE LINE # 305
;  }
                          ; SOURCE LINE # 306
; }
                          ; SOURCE LINE # 307
; retr:
                          ; SOURCE LINE # 308

?obj?retr:
; ;
; }
                          ; SOURCE LINE # 310
?C0073:
        RET
; END OF obj

;
; void delay()


        RSEG  ?PR?delay?DTMF_9KEY
delay:
                          ; SOURCE LINE # 312
; {
                          ; SOURCE LINE # 313
;  for(i=0;i<=0x50;i++)
                          ; SOURCE LINE # 314
        CLR    A
```

```
?C0074:
; {
                        ; SOURCE LINE # 315
;   for(j=0;j<=0x50;j++)
                        ; SOURCE LINE # 316
        CLR   A
        MOV   j,A
?C0077:
;   {
                        ; SOURCE LINE # 317
;   }
                        ; SOURCE LINE # 318
        INC    j
        MOV   A,j
        CJNE  A,#051H,?C0077
; }
                        ; SOURCE LINE # 319
?C0076:
        INC    i
        MOV   A,i
        CJNE  A,#051H,?C0074
;}
                        ; SOURCE LINE # 320
?C0080:
        RET
; END OF delay

        END
```

13.0 MECHANICAL DESIGN

13.1 Introduction

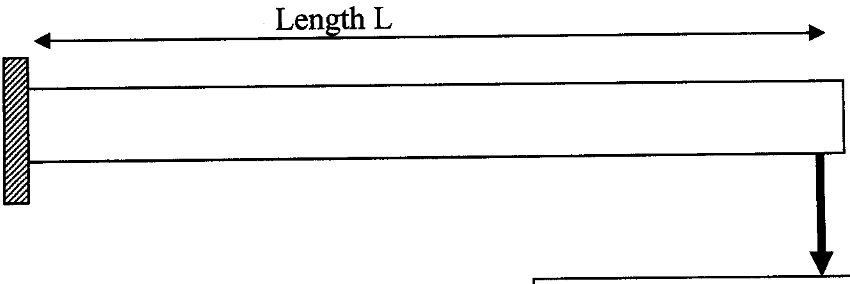
The Mechanical design is based on the knowledge of the properties of the material and the physical design. The design constrains for the project were

- Need for compact design
- Light in weight
- Place to accommodate all the hardware circuits

The Physical model of the robot was designed with the help of CAD software PROe-2000i[2] which enabled us to get the make a virtual model of the robot and make necessary changes. There was also necessity to calculate the necessary stress and other forces that act on the robot to select the Arm and the Motor for the drive.

The calculation for the arm dimensions are given below

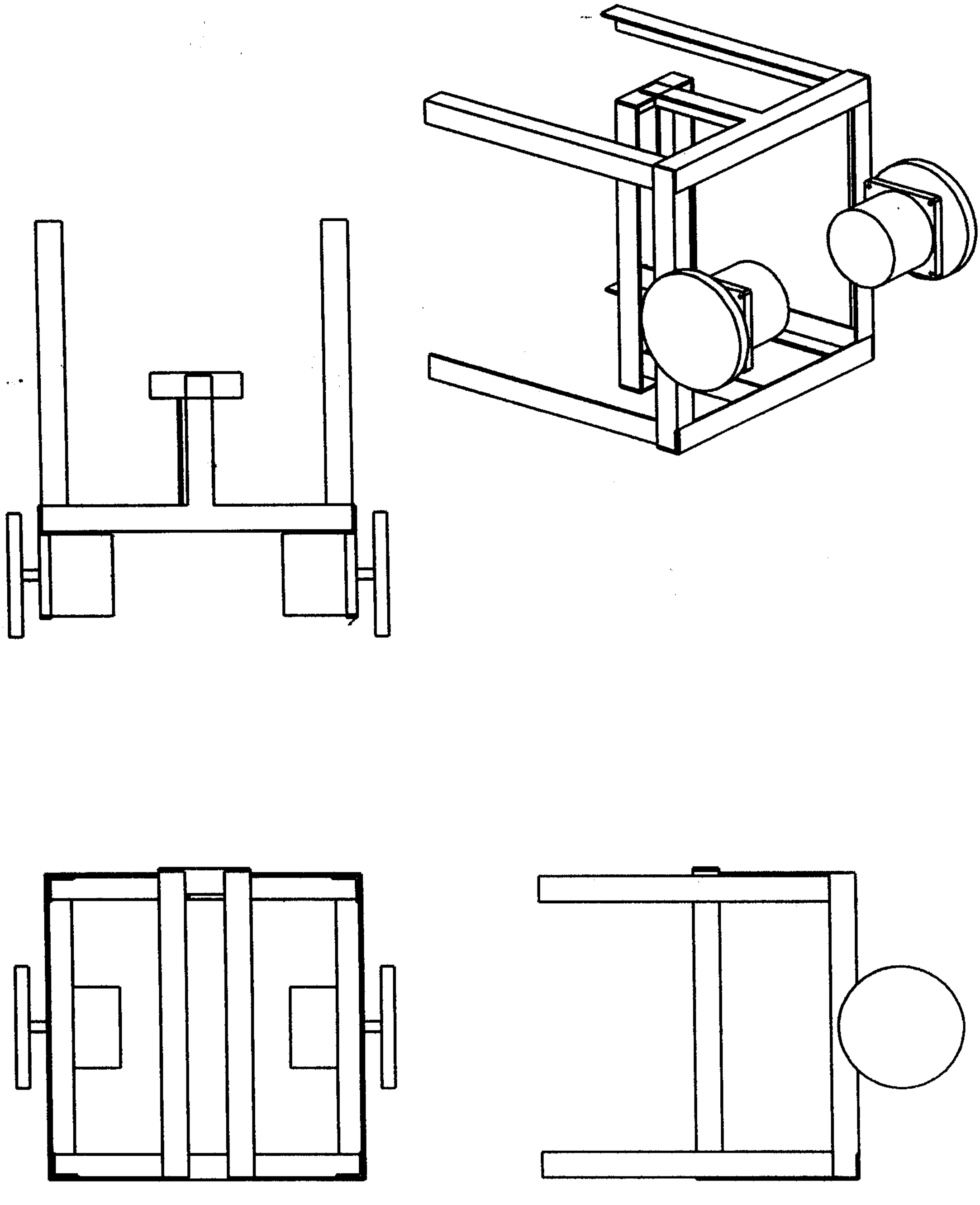
For the design let us consider the arm to be a cantilever fixed at one end and a force equal to the payload and the weight of the hanging portion is taken.



P=force of nuts + part of Arm hanging down(500+200=700gms approx)

We know that

Moment acting on the beam $M_{max} = P\ L = 700(9.81) * 200 = 1373400$



For the safe design we have

$$\frac{\mathbf{M}}{\mathbf{I}} = \frac{\mathbf{F}}{\mathbf{Y}} \longrightarrow \text{Equation 1}$$

where

M is M_{max}=1373400

F is the shear stress for the material (in our case the material is MS) with

F value = 360 n/mm^2

I moment of inertia for a rectangular plate given by the formula

$I = bd^3/12$ where d is breadth and b is the length and $I = 25\ d^3$

(here we assume the value of b and find d for safe value)

$Y = d/2$

When all these values are substituted is the equation we get the value of d = 76.3 mm

And we have 8mm for the design so the design is safe.

13.2 Calculation for the Drive Motor Torque

The next calculation for the specification for the stepper motor i.e. the torque necessary for the motor for this we calculate the weight of the robot which is about 3 kg which is the weight of all the exterior parts with no motors attached, so one the weight acting

Weight on one side = 3/4 = 0.75 kg

The action of force is about 6 inches from the centre of the motor

So the torque required for the motor is T = f * L = 0.75 * 6 = 4.5 Kgf

In our case the motors have 3 Kgf rating on one side so the totally 6 Kgf torque is generated which is enough to move the motor

CONCLUSION

CONCLUSION

The main objective was to develop a full fledged intelligent Robot with low cost and have a good implementation in the Industry, but right now the intelligence of the Robot is limited by certain ways. The Robot is capable of moving with the human help. The Robot has following freedom of movement

- Forward
- Reverse
- Left
- Right
- Movement of Arm up and Down
- Relay On/Off

The Robot has capability of being programmed in such a way that when the location of the part is defined in terms of x and y direction the Robot can pick it up and drop at the specified location in the room. It can also repeat the process number of times when they are to be used in the assembly line automation areas. The number of axis is now limited to two which can be increased to 4 –Axis. The Robot can also be converted to a wire free system by use of on-board power system and FM based-remote control panel with key bouncing circuit.

When all these are implemented to the present model it would be a Industrial Robot which can be readily made to work in those environment.

APPENDIX

DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 5.0V ± 20% (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC} - 0.1	V
V_{IL1}	Input Low-voltage ($\overline{EA}$)		-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[1] (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
V_{OL1}	Output Low-voltage[1] (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
V_{OH}	Output High-voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
V_{OH1}	Output High-voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, VCC = 5V ± 10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pull-down Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz		20	mA
		Idle Mode, 12 MHz		5	mA
	Power-down Mode[2]	V_{CC} = 6V		100	µA
		V_{CC} = 3V		40	µA

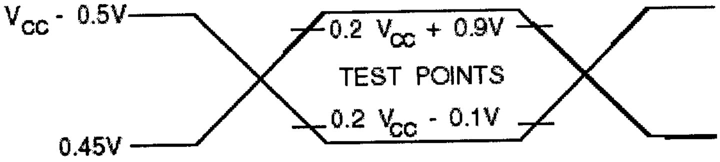
Notes:
1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 10 mA
   Maximum I_{OL} per 8-bit port: Port 0: 26 mA
   Ports 1, 2, 3: 15 mA
   Maximum total I_{OL} for all output pins: 71 mA
   If I_{OL} exceeds the test condition, 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.

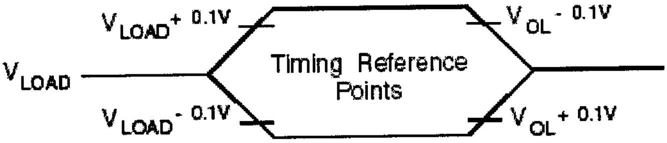
External Program and Data Memory Characteristics

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



NOTE: 1. AC Inputs during testing are driven at VCC - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at VIH min. for a logic 1 and VIL max. for a logic 0

Float Waveforms(1)



NOTE: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when 100 mV change from the loaded VOH/VOL level occurs.

Mnemonic		Description	Byte	Oscillator Period
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,R_n	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@R_i	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	R_n,A	Move Accumulator to register	1	12
MOV	R_n,direct	Move direct byte to register	2	24
MOV	R_n,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,R_n	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@R_i	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@R_i,A	Move Accumulator to indirect RAM	1	12
MOV	@R_i,direct	Move direct byte to indirect RAM	2	24
MOV	@R_i,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@R_i	Move External RAM (8-bit addr) to Acc	1	24
DATA TRANSFER (continued)				

Mnemonic		Description	Byte	Oscillator Period
MOVX	A,@DPTR	Move Exernal RAM (16-bit addr) to Acc	1	24
MOVX	@R_i,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,R_n	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@R_i	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@R_i	Exchange low-order Digit indirect RAM with Acc	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				

Mnemonic		Description	Byte	Oscillator Period
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,R_n	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@R_i	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	R_n,A	Move Accumulator to register	1	12
MOV	R_n,direct	Move direct byte to register	2	24
MOV	R_n,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,R_n	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@R_i	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@R_i,A	Move Accumulator to indirect RAM	1	12
MOV	@R_i,direct	Move direct byte to indirect RAM	2	24
MOV	@R_i,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@R_i	Move External RAM (8-bit addr) to Acc	1	24
DATA TRANSFER (continued)				

Mnemonic		Description	Byte	Oscillator Period
MOVX	A,@DPTR	Move Exernal RAM (16-bit addr) to Acc	1	24
MOVX	@R_i,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,R_n	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@R_i	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@R_i	Exchange low-order Digit indirect RAM with Acc	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				

BIBLIOGRAPHY

BIBLIOGRAPHY

Industrial Robotics

Technology, Programming and Application

- By Mikell P Groover McGraw Hill Inc Edition 1986

Robotics for Engineers

-By Yoram Koren McGraw-Hill, New York 1985

Let́ us C

- By Yashwant H Kanitkar Tata McGraw Hill 2nd edition

Programming notes and reference

- http://www.amtel.com

Power IC Data Book

- National Semiconductors