

# **MULTIUTILITY UNMANNED VEHICLE**

## **A PROJECT REPORT**

*Submitted by*

P.Gokul Rajesh

71201205015

M.Jude Vimal

71201205021

T.Karthikeyan

71201205023

*in partial fulfillment for the award of the degree  
of*

## **BACHELOR OF TECHNOLOGY**

**IN**

## **INFORMATION TECHNOLOGY**

**KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**

**ANNA UNIVERSITY : CHENNAI 600 025**

**APRIL 2005**

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “MULTIUTILITY UNMANNED VEHICLE” is the bonafide work of “P.Gokul Rajesh, M.Jude Vimal, T.Karthikeyan” who carried out the project work under my supervision.

  
SIGNATURE

**Dr. S. THANGASWAMY**

**HEAD OF THE DEPARTMENT**

Computer Science and Engineering  
Kumaraguru College of Technology  
Coimbatore - 641006.

  
SIGNATURE

**Mrs.N.Suganthi**


**SUPERVISOR**

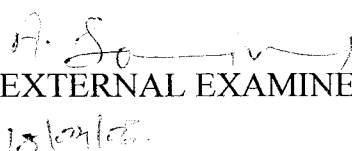
Senior Lecturer

Information Technology  
Kumaraguru College of Technology  
Coimbatore - 641006.

The candidates with University Register Nos. **71201205015**, **71201205021** and **71201205023** were examined by us in the project viva-voce examination held on

.....18-04-2005.....

  
INTERNAL EXAMINER

  
EXTERNAL EXAMINER

## ACKNOWLEDGEMENT

We are extremely grateful to **Dr.K.K.Padmanabhan**, B.Sc.(Engg.), M.Tech., Ph.D., Principal, Kumaraguru College of Technology for having given us a golden opportunity to embark on this project.

We are deeply obliged to **Dr.S.Thangaswamy**, Head of the Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks to our course co-coordinator **Mr. Bhaskar**, Assistant professor, Department of Information Technology and our guide **Mrs.N.Suganthi**, Senior Lecturer, Department of Information Technology for their helpful guidance and valuable support given to us throughout this project.

We thank the teaching and non-teaching staff of our Department for providing us the technical support in the duration of our project.

We also thank all of our friends who helped us to complete this project successfully.

## ABSTRACT

The need for wireless systems grows today in an exponential way because of its high significance. Our project is also centered on transmission and reception of signals in wireless fashion.

Exploring the space has become the top notch of today, for this we need robots that are capable of being controlled from the ground station with the ability to capture images with the help of a camera, and transmit them to the control room. Also, these robots are used to perform a variety of functions like,

- House safe system
- Fire fighter
- Exploring underground

The robot we have developed, which has a camera affixed at its top, could be used as a prototype for many industrial applications, in particular those that need the help of these robots, where men can't work.

A single FM transmitter cum receiver circuit is used for transmitting control signals from PC and to receive the same in the robot. These signals are decoded and given to the microcontroller 89C51. Two stepper motors are used to move the vehicle, and another stepper motor to enable the camera rotate through 360 degrees and capture the whole surrounding. The user interface is developed using Visual Basic, which the user uses to give control signals to the vehicle.

## TABLE OF CONTENTS

CHAPTER	TITLE	PAGE No.
	ABSTRACT	v
	LIST OF TABLES	viii
	LIST OF FIGURES	ix
	LIST OF SYMBOLS AND ABBREVIATIONS	x
1.	INTRODUCTION	1
	1.1 Wireless systems	1
	1.1.1 Fundamentals of wireless communication	1
	1.1.2 Frequency Modulation	1
	1.1.2.1 Comparison of am to fm	2
	1.1.2.2 FM applications	3
	1.1.3 DTMF	4
	1.1.3.1 DTMF usage	5
	1.1.3.2 Composition of DTMF signals	5
	1.1.3.3 How to transmit DTMF	6
	1.1.3.4 How to decode DTMF	6
	1.2 Printed Circuit Board	8
	1.2.1 PCB designing process	8
	1.3 Stepper motor	10
	1.3.1 Types of stepper motors	11
	1.3.1.1 Variable reluctance	11
	1.3.1.2 Permanent magnet	12
2.	TRANSMITTER SECTION	14
	2.1 Power supply	16
	2.1.1 Transformer	17

	<b>2.2 Interface circuit</b>	<b>18</b>
	<b>2.3 Encoder</b>	<b>20</b>
	<b>2.4 Encoder relay</b>	<b>21</b>
	<b>2.5 Transmitter</b>	<b>23</b>
<b>3.</b>	<b>RECEIVER SECTION</b>	<b>24</b>
	<b>3.1 FM Receiver:</b>	<b>26</b>
	<b>3.1.1 Description</b>	<b>26</b>
	<b>3.1.2 Features</b>	<b>26</b>
	<b>3.1.3 Functions</b>	<b>27</b>
	<b>3.2 Decoder</b>	<b>28</b>
	<b>3.3 Microcontroller</b>	<b>30</b>
	<b>3.3.1 Power modes of AT89C51</b>	<b>31</b>
	<b>3.3.2 Memory organization</b>	<b>32</b>
	<b>3.3.2.1 Program memory</b>	<b>32</b>
	<b>3.3.2.2 Data memory</b>	<b>33</b>
	<b>3.4 Driving unit</b>	<b>35</b>
	<b>3.5 Camera</b>	<b>36</b>
	<b>3.5.1 Description</b>	<b>37</b>
	<b>3.5.2 Features</b>	<b>37</b>
<b>4.</b>	<b>CONCLUSION</b>	<b>38</b>
<b>5.</b>	<b>APPENDICES</b>	<b>39</b>
	<b>5.1 Appendix 1 – SAMPLE CODE</b>	<b>39</b>
	<b>5.2 Appendix 2 - SNAP SHOTS</b>	<b>62</b>
	<b>5.3 Appendix 3 - DATA SHEETS</b>	<b>63</b>
<b>6.</b>	<b>REFERENCES</b>	<b>79</b>

## LIST OF TABLES

<b>Table No.</b>	<b>Title</b>	<b>Page No.</b>
<b>1.a</b>	<b>Frequency Table</b>	<b>5</b>
<b>1.b</b>	<b>List of Frequencies</b>	<b>7</b>
<b>3.a</b>	<b>pecification of WS – 809 AS CMOS Camera</b>	<b>37</b>

## LIST OF FIGURES

<b>Fig No.</b>	<b>Title</b>	<b>Page No.</b>
1.a	Depiction of Wideband FM and Narrowband FM	4
1.b	Sample of a Tone Frequency	7
1.c	Elements of a Stepper Motor	10
1.d	Stepper motor circuit	11
1.e	Variable Reluctance Motor	12
1.f	Permanent Magnet Motor	13
2.a	Block Diagram of Transmitter Section	14
2.b	Circuit Diagram of Transmitter Section	15
2.c	Block Diagram of Power Supply	16
2.d	Circuit Diagram of Power Supply	17
2.e	Parallel Port Interface	18
2.f	Encoder Schematic	20
2.g	Encoder Relay Schematic	21
2.h	Simple Illustration of Relay	22
2.i	FM Transmitter	23
3.a	Block Diagram of Receiver Section	24
3.b	Circuit Diagram of Receiver Section	25
3.c	FM Receiver	26
3.d	Decoder Schematic	28
3.e	Program Memory	33
3.f	Interfacing of Driving Unit and Motor	35
3.g	WS – 809 AS CMOS Camera	36
3.h	Video Signals – Receiver	36



## LIST OF SYMBOLS AND ABBREVIATIONS

<b>AC</b>	Alternating Current
<b>AM</b>	Amplitude Modulation
<b>BCD</b>	Binary Coded Decimal
<b>dB</b>	decibel
<b>DC</b>	Direct Current
<b>DR</b>	Deviation Ratio
<b>DTMF</b>	Dual Tone Multi Frequency
<b>FM</b>	Frequency Modulation
<b>GHz</b>	Gigahertz
<b>KHz</b>	Kilohertz
<b>MHz</b>	Megahertz
<b>NBFM</b>	Narrow Band Frequency Modulation
<b>PCB</b>	Printed Circuit Board
<b>PSW</b>	Program Status Word
<b>RF</b>	Radio Frequency
<b>Rms</b>	root mean square
<b>SFR</b>	Special Function Register
<b>WFM</b>	Wideband Frequency Modulation

# **1. INTRODUCTION**

## **1.1 WIRELESS SYSTEMS**

### **1.1.1 FUNDAMENTALS OF WIRELESS COMMUNICATION**

A wireless communication system deals with two directions, a transmitting direction and a receiving direction. Normally, the size of the antenna must be as large as one fourth of the wavelength of the signal to be transmitted or received to get enough efficiency. For this reason, the original signal (normally the voice) with a large wavelength must be transferred to a higher frequency (smaller wavelength) to downsize the antenna. At the transmitting end, the original signal is imposed on a locally generated radio frequency (RF) signal called a carrier. This process is called modulation.

This carrier signal, along with the information signal imposed on it, is then radiated by the antenna. At the receiving end, the signal is picked up by another antenna and fed into a receiver where the desired carrier with the imposed information signal is selected from among all of the other signals impinging on the antenna. The information signal (e.g., voice) is then extracted from the carrier in a process referred to as demodulation.

### **1.1.2 FREQUENCY MODULATION**

Frequency Modulation (FM) is a technique used to mix (encode) an information-carrying signal onto a much higher sine wave carrier frequency, so that the signal can be transmitted over long distances as a radio wave. Time varying changes in the information signal cause the instantaneous value of the carrier frequency to shift from its nominal or centre frequency, by an amount proportional to the amplitude of the signal ' this results in positive and negative

deviations in frequency. It is these shifts of frequency that are detected in the receiver, and which are demodulated to reveal the original signal.

Analogue or digital signals can be modulated onto a carrier wave using this type of frequency modulation, and in the case of analogue signals the frequency deviations will vary in a continuous process. Digital FM is implemented differently, and in this case the carrier frequency is shifted abruptly to any of a number of new fixed carrier frequencies, based on a binary (to the power of 2) system. The number of different levels of digitization that are required for digital FM, i.e. the number of discrete frequencies used, will depend on the bandwidth needed for the information to be transmitted.

Wireless communications commonly use narrowband FM, and this can cause frequency deviations from the carrier centre frequency of up to about 5 kHz. FM is often encountered by the general public in the form of FM radio broadcasting on the 88-108 MHz frequency band, which supports high fidelity sound and stereophonic reproduction. The size of FM radios has decreased to the point where it is now possible to find mobile phones that include an integrated FM radio built into the handset. FM communication is less liable to noise interference than amplitude modulation (AM), the other common type of modulation used for radio communications and broadcasting.

#### **1.1.2.1 COMPARISON OF AM TO FM**

It is universally agreed that FM radio stations have better quality sound than AM radio stations. Part of the reason for this is the noise immunity introduced by the non-linear modulation. Another reason is that the bandwidth for FM stations is 15 kHz, whereas AM stations are only allowed 5 kHz.

Also, FM receivers can have aerials which are half the wavelength of the transmitted carrier (due to the higher frequency of operation). This allows more signal power to be received than the AM case, where aerials would need to be many times longer.

### 1.1.2.2 FM APPLICATIONS

FM applications are divided into two broad categories:

- Wideband FM (WFM)
- Narrowband FM (NBFM)

The primary difference between the two types of FM is the number of sidebands in the modulated signal. Wideband FM has a large number (theoretically infinite) number of sidebands. Narrowband FM has only a single pair of significant sidebands.

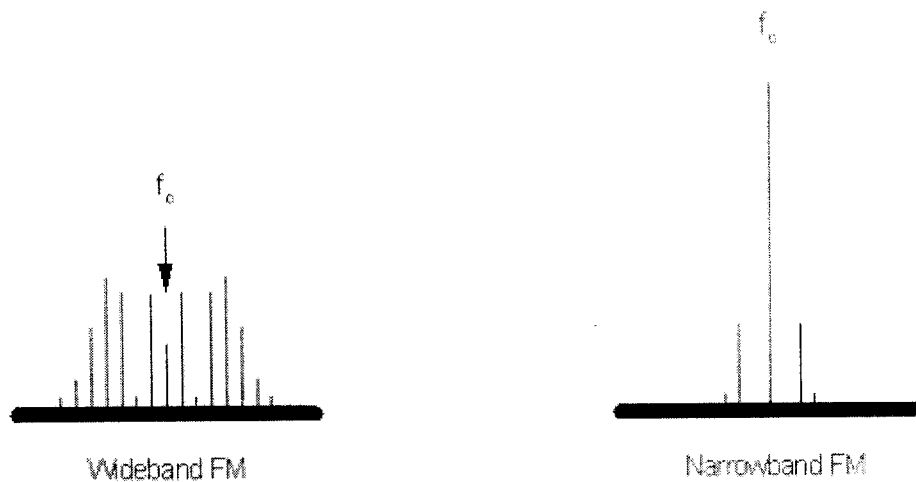
It is possible to determine if a particular FM signal will be wide or narrow band by looking at a quantity called the Deviation Ratio (DR). It is defined as the ratio of the maximum deviation of the FM signal to the maximum modulating frequency:

$$DR = \frac{\delta}{f_{max}}$$

The DR is also the modulation index of the highest modulating frequency. If the  $DR \geq 1.0$  the modulation is called wideband FM (WFM). If the  $DR < 1.0$ , the modulation is narrow band FM (NBFM).

One of the drawbacks of wideband FM is the large bandwidth required. Commercial FM broadcasting requires 150 KHz of bandwidth to transmit a 15 KHz audio signal, 5 times the bandwidth required for an AM signal.

The figure below compares the spectra of a WFM signal ( $DR = 5$ ) and a NBFM signal ( $DR = 0.5$ ). The separation between sidebands is equal to the modulating frequency. Thus the bandwidth for NBFM is  $2 * f_m$ , which is the same as for AM. However, for WFM, the bandwidth is approximately  $2N * f_m$ , where  $N$  = the number of sidebands.



**Fig 1.a** Depiction of Wideband FM and Narrowband FM

FM is used in the following applications:

1. Non-commercial broadcasting from 88 – 90 MHz (WFM)
2. Commercial broadcasting from 90 – 108 MHz (WFM)
3. Television audio (WFM)
4. Public Service communications (police, fire departments, etc.) from 30 – 50 MHz, 136-174 MHz, 450-470 MHz, and 800 MHz (NBFM)
5. Amateur Radio Service Communications 29.5 – 29.7 MHz, 52 – 54 MHz, 144 – 148 MHz, 222 – 225 MHz, 440 – 450 MHz, 902 MHz, 1240 – 1300 MHz, and other frequencies above 2.3 GHz (NBFM)
6. Point-to-point microwave links used by telecommunications companies (this is very wideband FM – the deviation of the carrier can be 10 MHz or more)

### 1.1.3 DTMF

DTMF (Dual-tone Multi Frequency) is a tone composed of two sine waves of given frequencies. Individual frequencies are chosen so that it is quite easy to design frequency filters, and so that they can easily pass through telephone lines (where the maximum guaranteed bandwidth extends from about 300 Hz to 3.5 kHz). DTMF was not intended for data transfer; it is designed for control signals

only. With standard decoders, it is possible to signal at a rate of about 10 "beeps" (=5 bytes) per second. DTMF standards specify 50ms tone and 50ms space duration. For shorter lengths, synchronization and timing becomes very tricky.

### 1.1.3.1 DTMF USAGE:

DTMF is the basis for voice communications control. Modern telephony uses DTMF to dial numbers, configure telephone exchanges (switchboards), and so on. Occasionally, simple floating codes are transmitted using DTMF - usually via a CB transceiver (27 MHz). It is used to transfer information between radio transceivers, in voice mail applications, etc.

Almost any mobile (cellular) phone is able to generate DTMF after establishing connection. If your phone can't generate DTMF, you can use a stand-alone "dialer". DTMF was designed so that it is possible to use acoustic transfer, and receive the codes using standard microphone.

### 1.1.3.2 COMPOSITION OF DTMF SIGNALS

The table shows how to compose any DTMF code. Each code, or "beep", consists of two simultaneous frequencies mixed together (added amplitudes). Standards specify 0.7% typical and 1.5% maximum tolerance. The higher of the two frequencies may have higher amplitude (be "louder") of 4 dB max. This shift is called a "twist". If the twist is equal to 3 dB, the higher frequency is 3 dB louder. If the lower frequency is louder, the twist is negative.

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

**Table 1.a** Frequency Table

This table resembles a matrix keyboard. The X and Y coordinates of each code give the two frequencies that the code is composed of. Notice that there are 16 codes; however, common DTMF dialers use only 12 of them. The "A" through "D" is "system" codes. Most end users won't need any of those; they are used to configure phone exchanges or to perform other special functions.

### **1.1.3.3 HOW TO TRANSMIT DTMF**

Most often, dedicated telephony circuits are used to generate DTMF (for example, MT8880). On the other hand, a microprocessor can do it, too. Just connect a RC filter to two output pins, and generate correct tones via software. However, getting the correct frequencies often requires usage of a suitable Xtal for the processor itself - at the cost of non-standard cycle length, etc. So, this method is used in simple applications only.

### **1.1.3.4 HOW TO DECODE DTMF**

It is not easy to detect and recognize DTMF with satisfactory precision. Often, dedicated integrated circuits are used, although a functional solution for DTMF transmission and receiving by a microprocessor exists. It is rather complicated, so it is used only marginally. Most often, a MT 8870 or compatible circuit would be used.

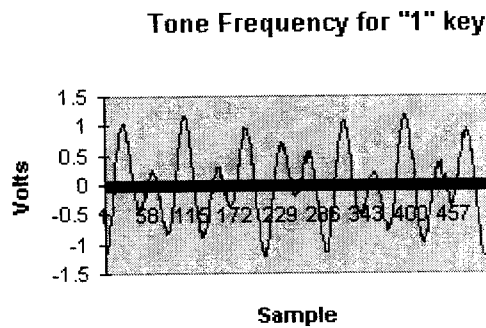
Most decoders detect only the rising edges of the sine waves. So, DTMF generated by rectangular pulses and RC filters works reliably. The mentioned MT 8870 uses two 6th order band pass filters with switched capacitors. These produce nice clean sine waves even from distorted inputs, with any harmonics suppressed.

In DTMF there are 16 distinct tones. Each tone is the sum of two frequencies: one from a low and one from a high frequency group. There are four different frequencies in each group.

DTMF Row/Column Frequencies	
<b>LOW-FREQUENCIES</b>	
ROW #	FREQUENCY (HZ)
R1: ROW 0	697
R2: ROW 1	770
R3: ROW 2	852
R4: ROW 3	941
<b>HIGH-FREQUENCIES</b>	
COL #	FREQUENCY (HZ)
C1: COL 0	1209
C2: COL 1	1336
C3: COL 2	1477
C4: COL 3	1633

**Table 1.b** List of Frequencies

The graph as in fig 1.b is a captured screen from an oscilloscope. It is a plot of the tone frequency for the "1" key:



**Fig 1.b** Sample of a Tone Frequency



## 1.2 PRINTED CIRCUIT BOARD

### 1.2.1 PCB DESIGNING PROCESS

A PCB is designed by performing the following steps:

- Drawing the circuit in a graph
- Drilling
- Transferring the diagram to the board
- Etching
- Placing components
- Soldering

The first step is to draw the necessary circuit on an inch graph sheet which involves providing provisions to place components with minimal spacing between each other. The spots where components need to be placed are marked by points. The circuit is completed by joining the points as per the circuit diagram.

With the graph drawn successfully, it is placed over the copper clad sheet and the points in the graph are drilled. The sheet is scrubbed using a file and is then cleanly washed.

Once the drilling process is completed, the diagram on the graph is transferred to the copper coated sheet using markers. The drilled spots are neatly marked and are connected as in the graph.

The next phase is etching. Here the sheet is placed in a solution of anhydrous ferric chloride mixed with water. A little of dilute hydrochloric acid added to the solution quickens the etching process. Etching is the process of removing the copper coating from the sheet except in the places coated by marker. Once done,

the sheet is washed to remove the marker on it. Then it is dried and cleaned carefully.

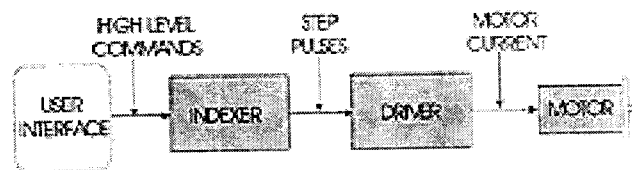
With the guide of the circuit, the components are placed in the holes made by drilling. Once placed in proper position, the components are then soldered with each other and to the sheet itself. With the completion of soldering, the ICs are placed in their holders and the PCB is fully completed.

## 1.3 STEPPER MOTOR

Motion Control, in electronic terms, means to accurately control the movement of an object based on speed, distance, load, inertia or a combination of all these factors. There are numerous types of motion control systems, including; Stepper Motor, Linear Step Motor, DC Brush, Brushless, Servo, Brushless Servo and more. This document will concentrate on Step Motor technology.

In Theory, a Stepper motor is a marvel in simplicity. It has no brushes, or contacts. Basically it's a synchronous motor with the magnetic field electronically switched to rotate the armature magnet around.

A Stepping Motor System consists of three basic elements, often combined with some type of user interface (Host Computer, PLC or Dumb Terminal).



**Fig 1.c** Elements of a Stepper Motor

The Indexer (or Controller) is a microprocessor capable of generating step pulses and direction signals for the driver. In addition, the indexer is typically required to perform many other sophisticated command functions.

The Driver (or Amplifier) converts the indexer command signals into the power necessary to energize the motor windings. There are numerous types of drivers, with different current/amperage ratings and construction technology. Not all drivers are suitable to run all motors, so when designing a Motion Control System the driver selection process is critical.

The Step Motor is an electromagnetic device that converts digital pulses into mechanical shaft rotation. Advantages of step motors are low cost, high reliability,

high torque at low speeds and a simple, rugged construction that operates in almost any environment. The main disadvantages in using a step motor is the resonance effect often exhibited at low speeds and decreasing torque with increasing speed.

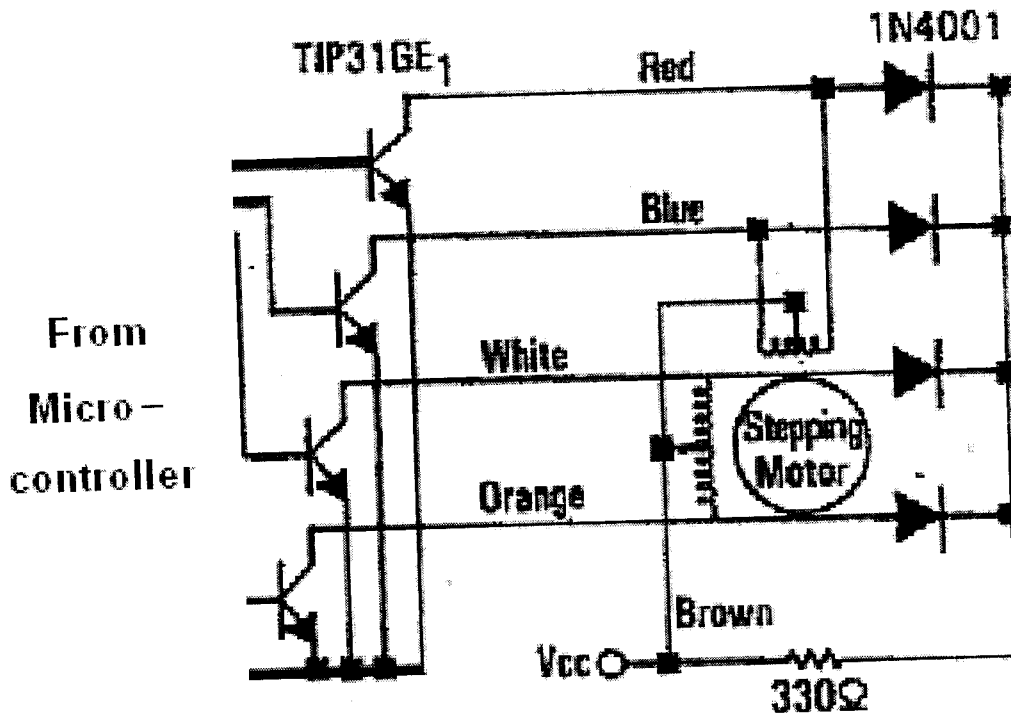


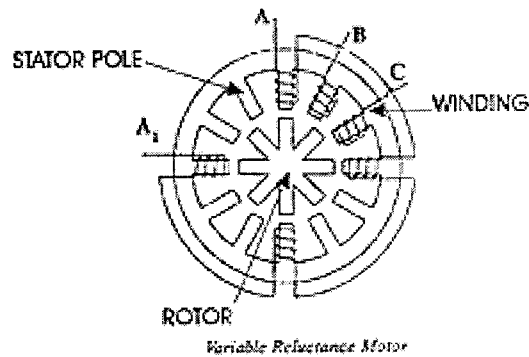
Fig 1.d Stepper motor circuit

### 1.3.1 TYPES OF STEPPER MOTORS

There are basically three types of stepper motors; variable reluctance, permanent magnet and hybrid. They differ in terms of construction based on the use of permanent magnets and/or iron rotors with laminated steel stators.

#### 1.3.1.1 VARIABLE RELUCTANCE

The variable reluctance motor does not use a permanent magnet. As a result, the motor rotor can move without constraint or "detent" torque. This type of construction is good in non industrial applications that do not require a high degree of motor torque, such as the positioning of a micro slide .

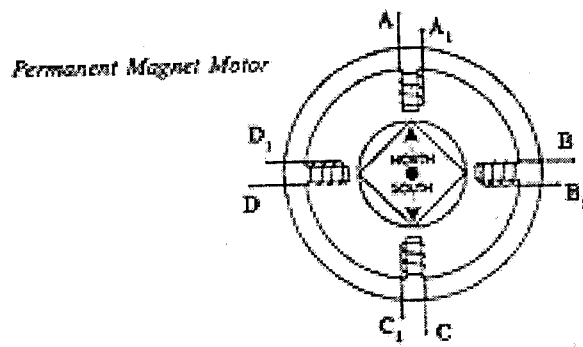


**Fig 1.e** Variable Reluctance Motor

The variable reluctance motor in the above illustration has four "stator pole sets" (A, B, C, D), set 15 degrees apart. Current applied to pole A through the motor winding causes a magnetic attraction that aligns the rotor (tooth) to pole A. Energizing stator pole B causes the rotor to rotate 15 degrees in alignment with pole B. This process will continue with pole C and back to A in a clockwise direction. Reversing the procedure (C to A) would result in a counterclockwise rotation.

### 1.3.1.2 PERMANENT MAGNET

The permanent magnet motor, also referred to as a "canstack" motor, has, as the name implies, a permanent magnet rotor. It is a relatively low speed, low torque device with large step angles of either 45 or 90 degrees. Its simple construction and low cost make it an ideal choice for non industrial applications, such as a line printer print wheel positioner.



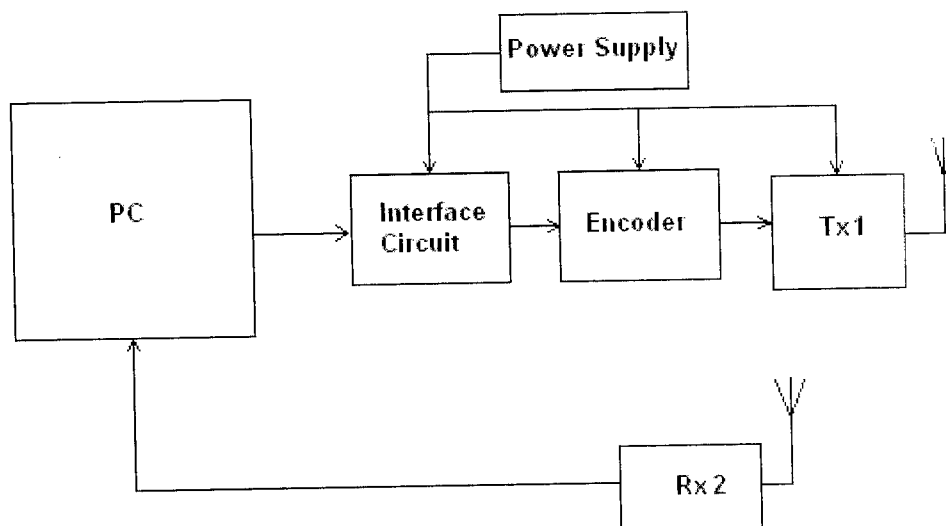
**Fig 1.f** Permanent Magnet Motor

Unlike the other stepping motors, the PM motor rotor has no teeth and is designed to be magnetized at a right angle to its axis. The above illustration shows a simple, 90 degree PM motor with four phases (A-D). Applying current to each phase in sequence will cause the rotor to rotate by adjusting to the changing magnetic fields. Although it operates at fairly low speed the PM motor has a relatively high torque characteristic.

## 2. TRANSMITTER SECTION

At the system side the following are the PCBs, as depicted in figure 1, with VLSI chips affixed on them to perform specific functions, enabling the user to control the vehicle from a pc, whose signals are processed and then transmitted to the vehicle as control signals.

- Power supply
- Interface Circuit
- Encoder
- Transmitter
- Receiver

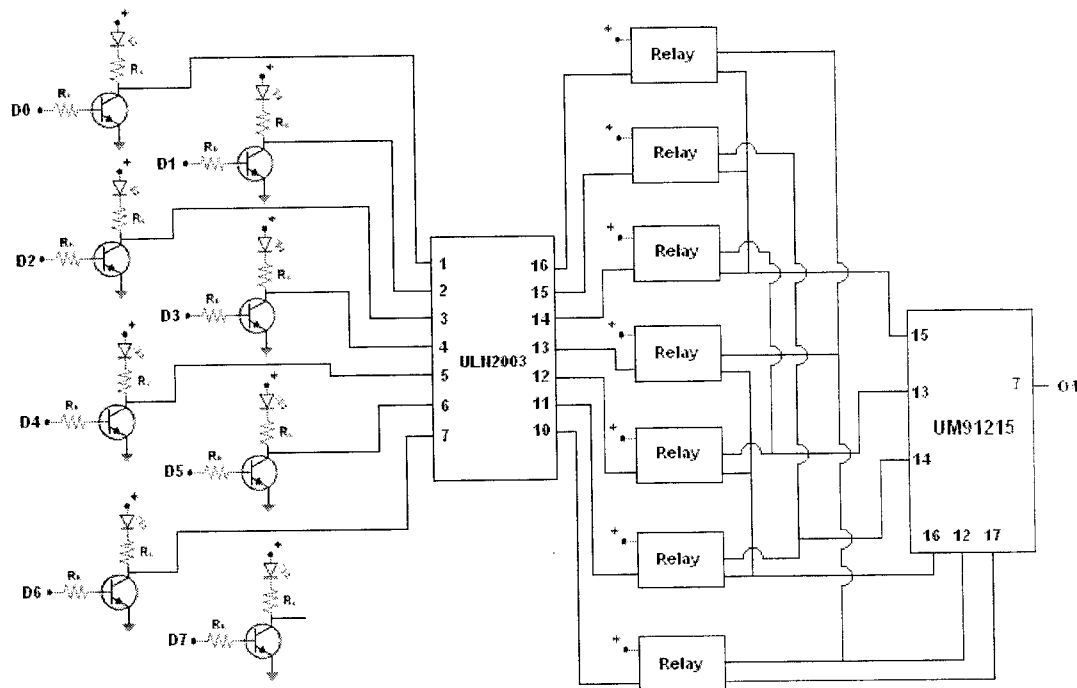


**Fig 2.a** Block Diagram of Transmitter Section

The commands from the user are converted into respective 8 bit data. In our case only one bit is high (1) and the rest low (0) to denote a particular user command. Hence, 00000001, 00000010, 00000100, 00001000, 00010000, 00100000, 01000000, 10000000 are the possibilities to denote a command. This data is fed to the parallel port output pins of the computer, from which is extracted to the interfacing circuit. Here the signals are of very low voltage, since the electronic

components detect signals around 5V, these low voltage signals are boosted up using transistors. The boosted signals are then fed to IC ULN2003 which in turn drives relays connected to them. A matrix arrangement is made using relays that resemble DTMF matrix. Based on the pin that receives high voltage from the computer a particular relay is switched ON at a time, which drives a row and a column of the matrix short, which in turn when connected to UM91215 encoder produces a tone unique to every bit driven high. This tone is fed to a FM transmitter where it is modulated using a carrier of 105MHz and is transmitted through an antenna.

Simultaneously a video receiver captures video signal transmitted by a camera from the robot and in turn sends them to a video capture card placed in the computer. Windows ® operating system has built-in dll files for reading this signal and reproducing the video as captured by the camera. The application program uses this dll files to play the captured video on screen.

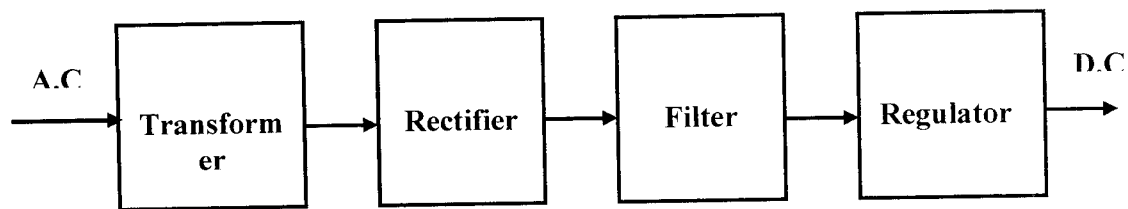


**Fig 2.b** Circuit diagram of transmitter section



## 2.1 POWER SUPPLY

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

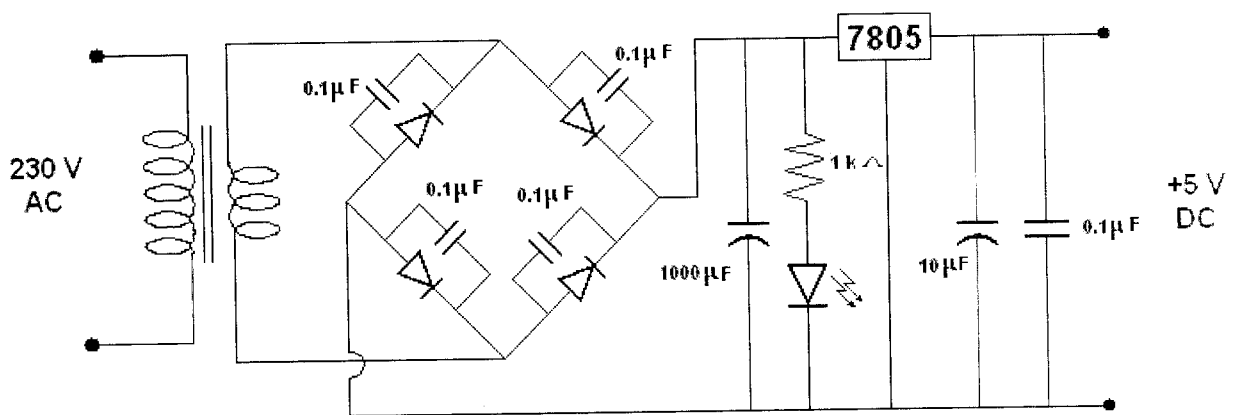


**Fig 2.c** Block Diagram of Power Supply

The main things used in the power supply unit are Transformer, Rectifier, Filter, and Regulator. The block diagram of the power supply unit is shown in the above block diagram. The 230V ac supply is converted into 12V ac supply through the transformer. The output of the transformer has the same frequency as in the input ac power. This ac power is converted into dc power through the diodes. Here the bridge diode is used to convert the ac supply to the dc power supply. This converted dc power supply has the ripple content and for the normal circuit operation the ripple content of the dc power supply should be as low as possible. Because the ripple content of the power supply will reduce the life of circuit.

So to reduce the ripple content of the dc power supply, the filter is used. The filter is nothing but the large value capacitance. The output waveform of the filter capacitance will almost be the straight line.

This filtered output will not be the regulated voltage. For the normal operation of the circuit it should have the regulated output. Specifically for the microcontroller IC regulated constant 5V output voltage should be given. For this purpose 78xx regulator should be used in the circuit. In that number of IC, the 8 represents the positive voltage and if it is 9, it will represent the negative voltage. The xx represents the voltage. If it is 7805, it represent 5V regulator, and if it is 7812, it represent 12V regulator. Thus the regulated constant output can be obtained. The brief description of the above blocks is follows.



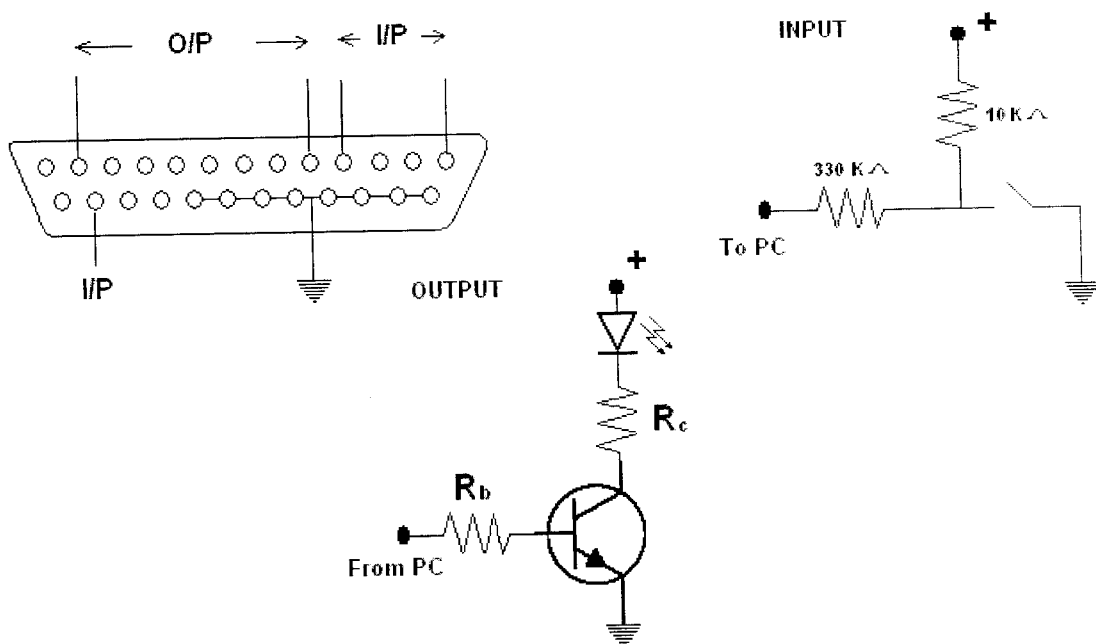
**Fig 2.d** Circuit Diagram of Power Supply

### 2.1.1 TRANSFORMER

A transformer is a static (or stationary) piece of which electric power in one circuit is transformed into electric power of the same frequency in another circuit. It can raise or lower the voltage in a circuit but with a corresponding decrease or increase in current. It works with the principle of mutual induction. In our project we are using step down transformer for providing a necessary supply for the electronic circuits. In our project we are using a 230/12 v transformer.

## 2.2 INTERFACE CIRCUIT

The model we have suggested being designed for wireless mode of operation, the function of the interfacing circuit comes to the fore. This board stands as the mediator between the system and the transmitter circuit in the transmitter end i.e., the instructions that are got from the user to control the vehicle are in the form that could be read by the computer which are not compatible with the microcontroller, so these instructions need to be converted to a suitable form that could be properly interpreted at the receiver end that is by the microcontroller and should carry out the function specified in the right way.



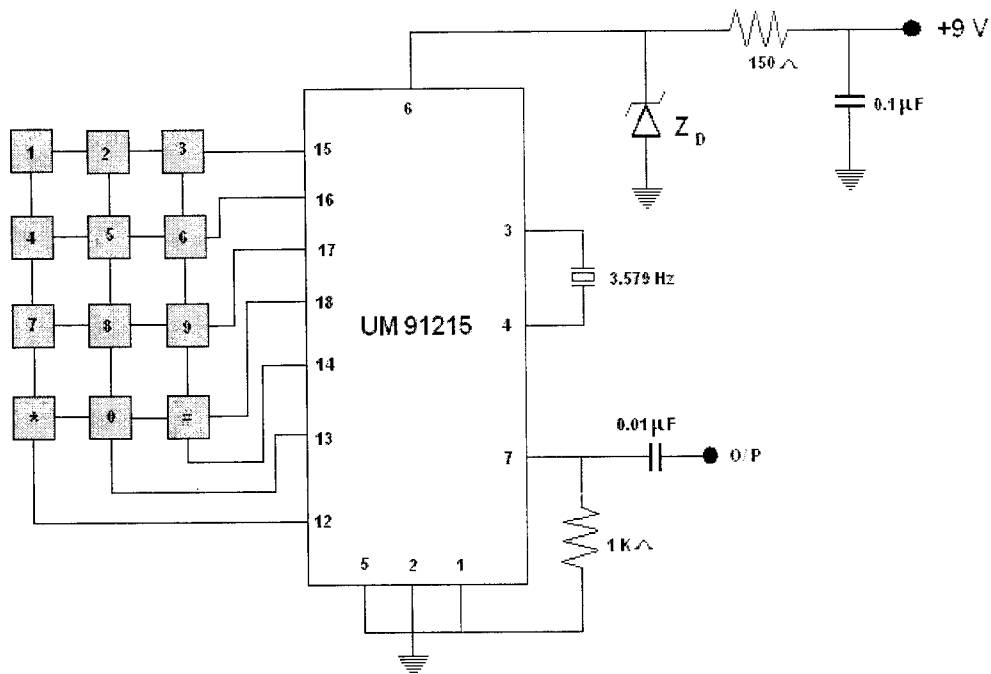
**Fig 2.e** Parallel Port Interface

Of the 25 pins in the parallel port, the pins from 2 to 9 are used for output and the pins from 10 to 13 and 15 are used for input. The signals to and from these pins are felt even if a light load passes by. For this purpose we use npn junction diodes, in which when a signal is passed, a small current passes by, which is called as the base current. As this current flows the collector current flows through the emitter, as a result the control signal from the user is fed to the encoder relay as electronic signals. The input pins are to receive the signals from the vehicle. In the same way,

the output pins are to transmit the signals that emanate from the computer, which are the control signals to the vehicle that are caused by the user. The parallel port is connected to switches on the transmitter side for testing purpose.

## 2.3 ENCODER

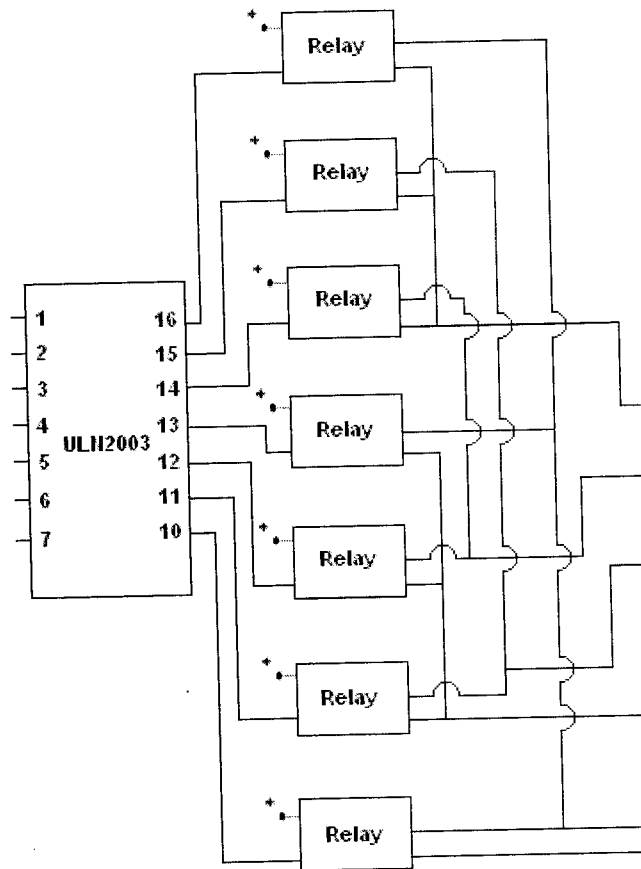
The UM91215 is a single chip, silicon gate, CMOS integrated circuit with an on-chip oscillator for a 3.68MHz crystal or ceramic resonator. It provides dialing pulse (DP) or dual tone multi-frequency (DTMF) dialing. A standard 4\*4 matrix keyboard can be used to support either DP or DTMF modes. Up to 32 digits can be saved in the on-chip RAM for redialing. In the DTMF mode, minimum tone duration and minimum inter tone pause provide for rapid dialing. Maximum tone duration is dependent upon the key depression time in manual dialing.



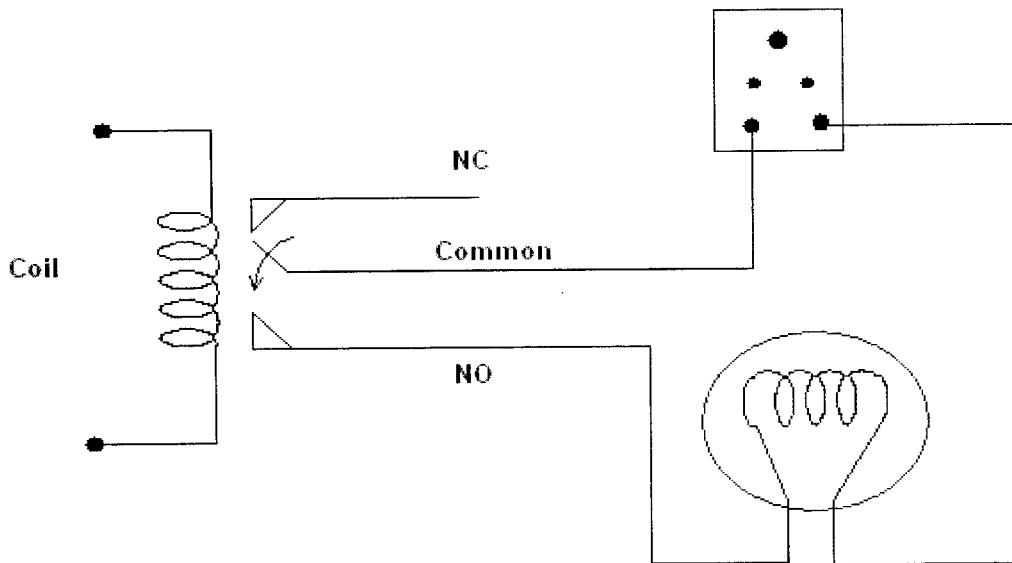
**Fig 2.f** Encoder Schematic

## 2.4 ENCODER RELAY

The ULN2003, high voltage, high current Darlington array each containing seven open collector Darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.



**Fig 2.g** Encoder Relay Schematic



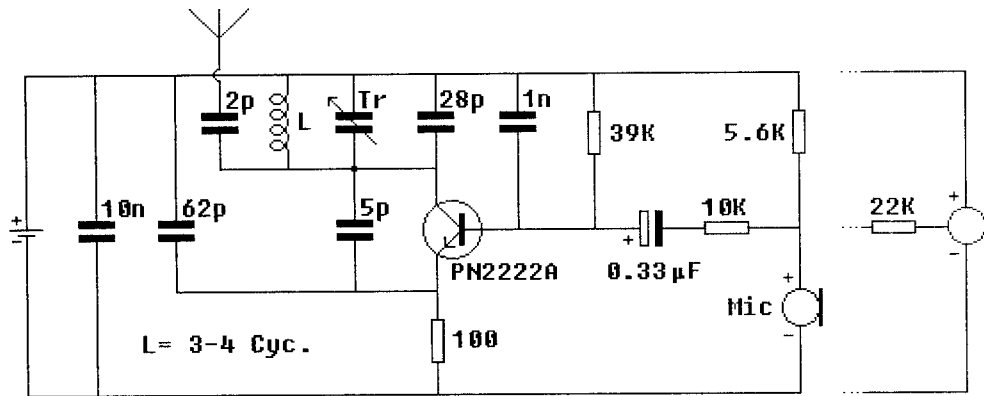
**Fig 2.h** Simple Illustration of Relay

The ULN2003, used have 16 pin plastic DIP packages with a copper lead frame to reduce thermal resistance.

As illustrated in fig 2.h the Normally Close (NC) line is closed with the common line when the relay is OFF. But when current flows through the coil due to electromagnetic effect the common line deflects and closes with the Normally Open (NO) line, thus completing any circuit connected to it. Thus, relay is used to switch high voltage circuits connected between NO and common, ON and OFF, using a low voltage passed through the coil. Hence, a relay is used as an electrical switch.

## 2.5 TRANSMITTER

An ordinary transmitter that alters the frequency of the carrier signal in accordance to the frequency of the modulating signal is used, as the frequency modulated signals are stronger than amplitude modulated signals and also these signals are less subjected to interferences caused by other obstacles in the atmosphere.



Simple FM Transmitter, 30/50 meters range, operates with 1.5V

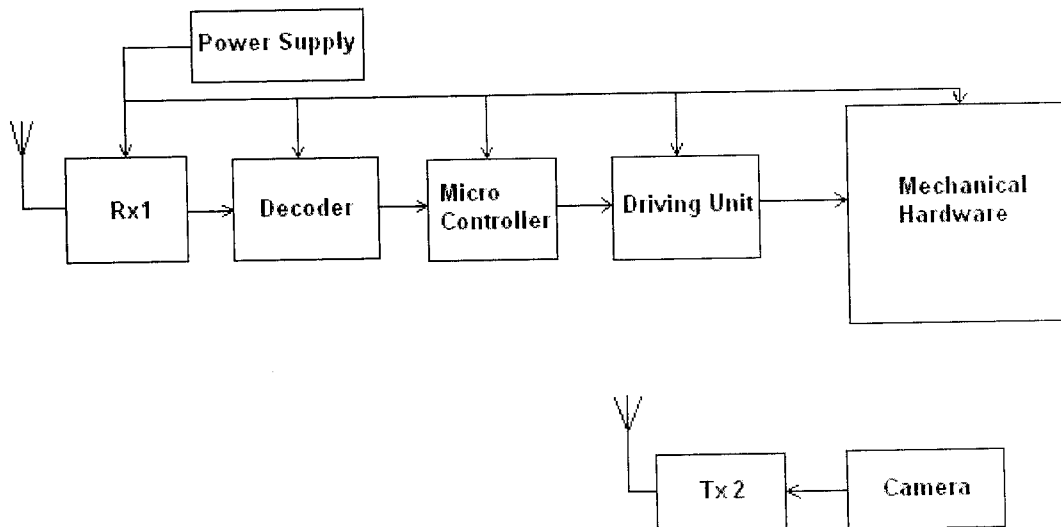
Fig 2.i FM Transmitter

The distance between the transmitter and the receiver could be elapsed to a maximum distance of 30 to 50 meters in the absence of any obstruction for better quality in reception. Also, the transmitter cum receiver is available as a pair, which avoids the need for perfection in winding up of the coils, thereby increasing the perception of reception. Thus, the control signals to the vehicle, in the form of a tone generated at a particular frequency, with respect to the signal being generated is transmitted from the system side. This form of signal is got from the encoder where a crystal is used to generate the tone at the particular frequency, in accordance to the signal being transmitted.



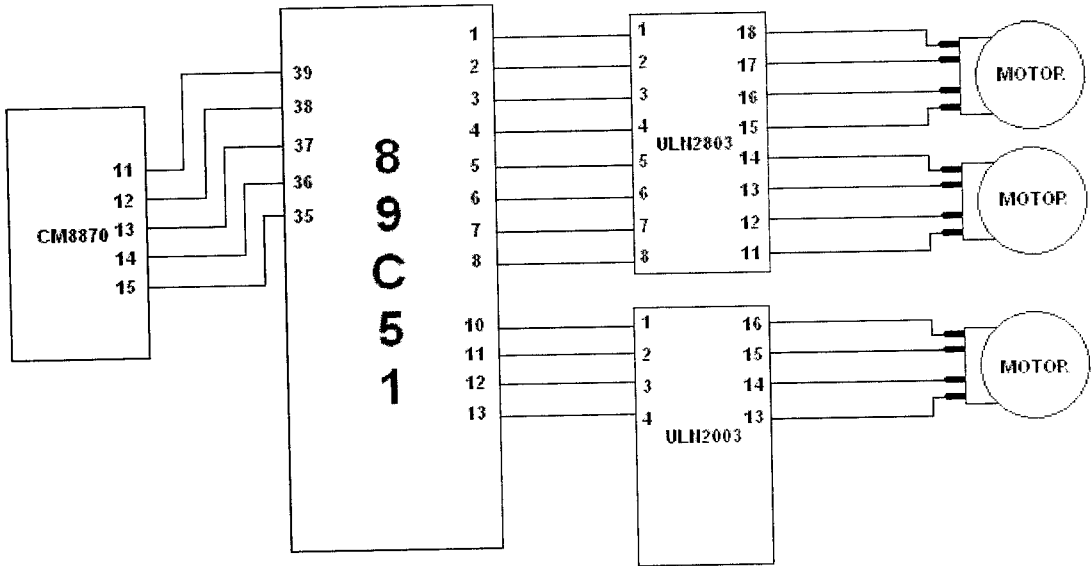
### 3. RECEIVER SECTION

The FM signals received by the receiver are demodulated to recover the DTMF tones. This tone is fed as input to CM8870 which is the decoder IC used here. The output of the decoder is a 4 bit BCD data that represent the respective bit that was high in the transmitting end. This 4 bits along with an additional bit representing reception of a DTMF signal is given as input to 89C51 micro controller's port 0. The pins from port 1 and port 3 are connected to ULN2803 and ULN2003 ICs respectively. Naturally we use the ports 1 and 3 as output ports and port 0 as input port. The microcontroller is programmed suitably to identify the command received as input and send output signals to assist the motion of the stepper motors connected to ULN2803 and ULN2003. Out of the two motors connected to ULN2803 one is used for forward and reverse motion of the vehicle, the other for the rotary motion of the vehicle. The motor connected to ULN2003 is used to bring about the rotary motion of the camera.



**Fig 3.a** Block Diagram of Receiver Section

Simultaneously the camera connected to the vehicle captures live video, modulates the same in the order of GHz and transmits them through an antenna connected to the camera.



**Fig 3.b** Circuit diagram of receiver section

### 3.1 FM RECEIVER:

FM Receiver consists of CXA1619AS, a one-chip FM radio IC It receives the transmitted signals from the transmitter and demodulates to give the DTMF tones.

#### 3.1.1 DESCRIPTION

CXA1619AM/AS is a one-chip FM/AM radio IC designed for radio-cassette tape recorders and headphone tape recorders, and has the following functions.

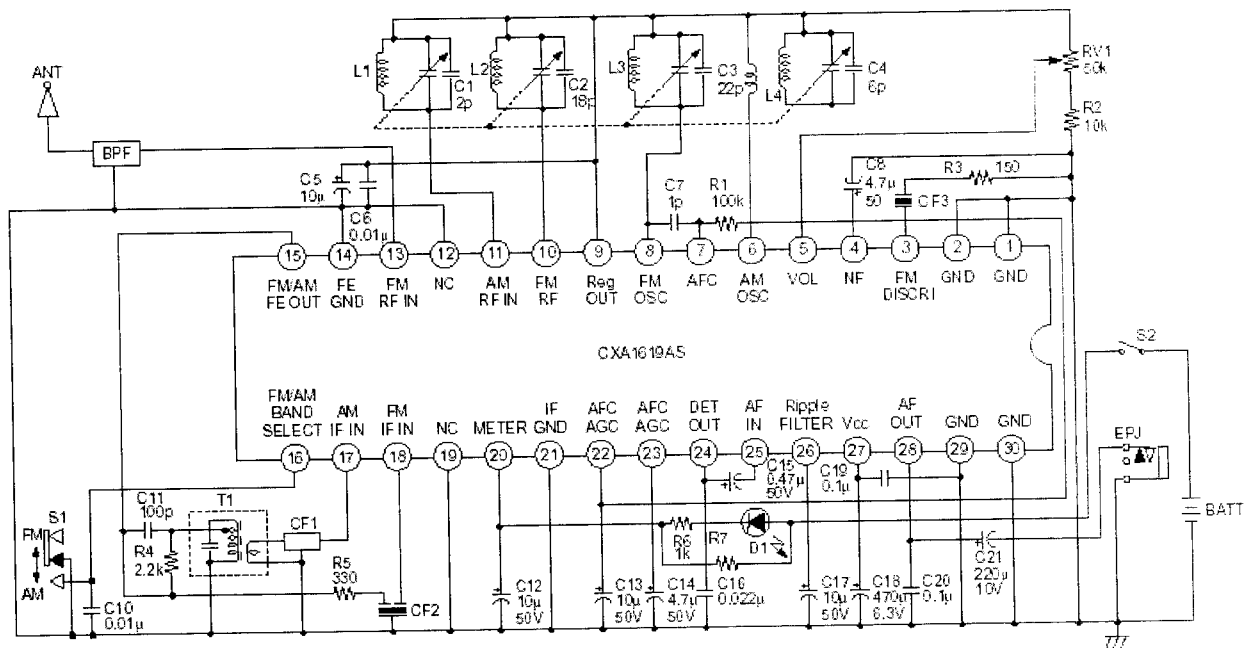


Fig 3.c FM Receiver

#### 3.1.2 FEATURES

- Small number of peripheral components.
- Low current consumption (VCC=3 V)
  - For FM : ID=5.8 mA (Typ.)
  - For AM : ID=4.7 mA (Typ.)
- Built-in FM/AM select switch.
- Large output of AF amplifier.

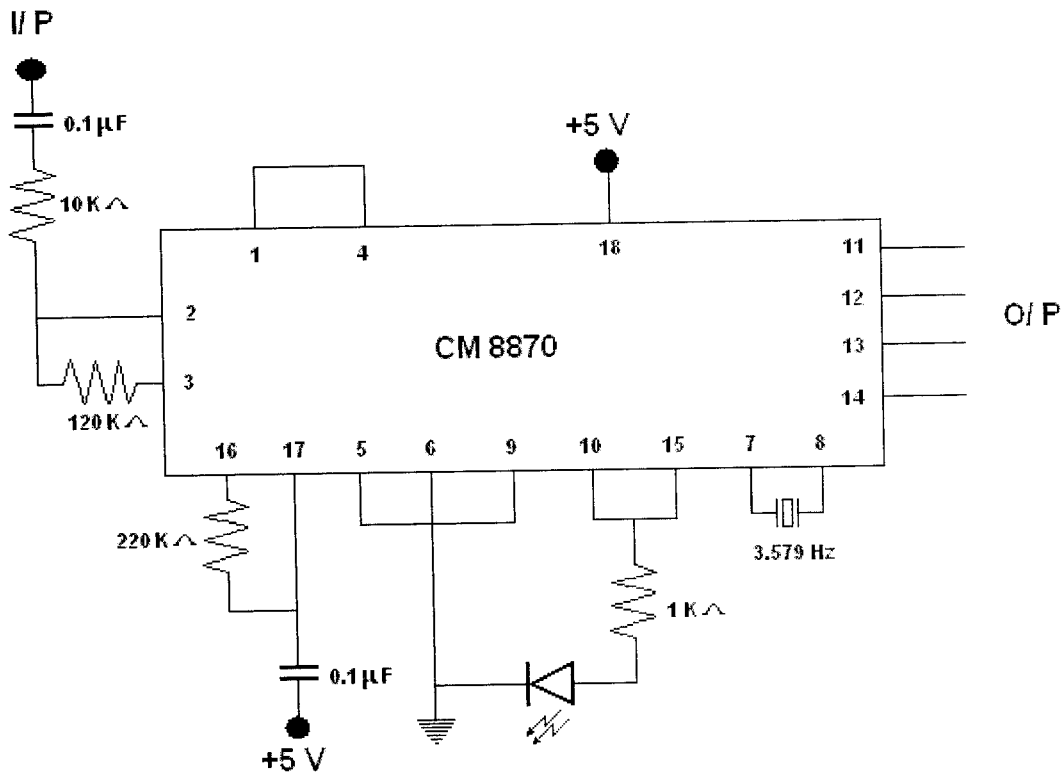
### **3.1.3 FUNCTIONS**

#### **FM section**

- RF amplifier, Mixer and OSC (incorporating AFC variable capacitor).
- IF amplifier
- Quadrature detection
- Tuning LED driver AM section
- RF amplifier, Mixer and OSC (with RF AGC)
- IF amplifier (with IF AGC)
- Detector
- Tuning LED driver AF section
- Electronic volume control

### 3.2 DECODER

The function of the decoder is to obtain the transmitted signals from the receiver, which is received in the form of tones, each tone being differentiated by its frequency, and the decoder does the conversion of these tones to signals in a form that could be processed by the microcontroller. Thereby, the decoder enables the microcontroller to identify the transmitted information and react upon in accordance to the control signals being transmitted and received.



**Fig 3.d** Decoder Schematic

The chip used for this purpose is an 18 pin IC CM8870, which uses either a quartz crystal or ceramic resonators to decode the signal received. The internal structure of the IC consists of a band split filter that separates the high and low tones of the received pair, which is followed by a digital decode(counting) which verifies both the frequency and the duration of the received tones before passing the resultant 4-bit code. The decoder thus by using digital counting techniques for detection and decoding, converts all 16 DTMF tone pairs into 4bit code.

The CM8870 provides full DTMF receiver capability by integrating both the band split filter and digital decoder functions into a single 18-pin DIP. The CM8870 is manufactured using state-of-the-art CMOS process technology for low power consumption and precise data handling. The filter section uses a switched capacitor technique for both high and low group filters and dial tone rejection.

### 3.3 MICROCONTROLLER

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 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 Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

The features of AT89C51 are,

- Compatible with MCS-51 products
- On - Chip Flash Program Memory
- Fully static operations: 0Hz to 24MHz
- 128 bytes internal RAM
- Two 16 bit timers
- 5 interrupt sources
- 32 I/O lines
- Programmable serial channel
- Low power idle and power down modes
- 8 Bit CPU optimized for control applications
- Extensive Boolean processing (Single - bit Logic) Capabilities.
- On - Chip Data RAM
- Bi-directional and Individually Addressable I/O Lines
- Multiple 16-Bit Timer/Counters
- Full Duplex UART
- Multiple Source / Vector / Priority Interrupt Structure
- On - Chip Oscillator and Clock circuitry.
- On - Chip EEPROM
- SPI Serial Bus Interface

- Watch Dog Timer

The AT89C51 is designed with static logic for operation, down to zero frequency and supports two software selectable power saving modes.

### **3.3.1 POWER MODES OF AT89C51**

The CMOS circuitry provides support to save the power, with the help of the two software-invited reduced power modes, provided by Atmel's Flash micro controllers. The following are the operation modes of the AT89C51 microcontroller,

- ✓ Idle Mode
- ✓ Power-down Mode

In the latter mode of power saving, the CPU is stopped from functioning, while allowing the RAM, counters/timers, serial ports and interrupt system to continue functioning. With the CPU being turned off in this mode, while the RAM and other peripherals still functioning, the current drawn from the power supply is about 15% lower than the current that is drawn when the device is fully active.

The Power-down Mode behaves differently, by saving the contents of the RAM and also it freezes the oscillator, disabling all other chip functions until the next hardware reset. Thus in this power saving mode, the device is made to draw less than 15 Micro Amps, which could be even lowered to 0.6 Micro Amos, by suspending all the on-chip activities, while the RAM on the chip holding the data continuously.

Another notable feature of the microcontroller is the Power on Reset. The circuit holds the RST pin high for an amount of time which depends on the capacitor value and the rate at which it charges when power is turned on.



To ensure a valid reset, the RST pin must be held high long enough to allow the oscillator to start up plus two machine cycles. On power up, Vcc should rise within approximately 10ms. The oscillator start-up time depends on the oscillator frequency. For a 10MHz crystal, the start-up time is typically 1ms. With the given circuit, reducing Vcc quickly to 0 causes the RST pin voltage to momentarily fall below 0V. However, this voltage is internally limited to 1 and will not harm the device

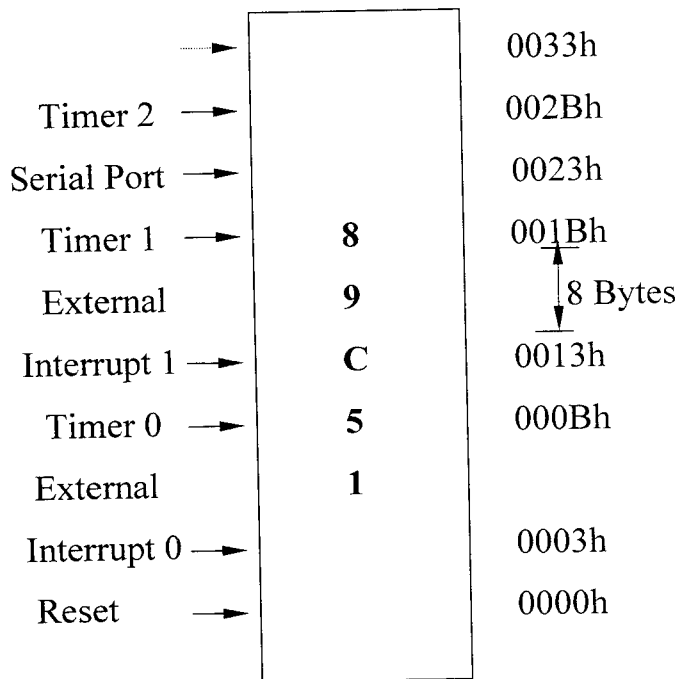
### **3.3.2 MEMORY ORGANIZATION**

The important point about the organization of memory in AT89C51 is the separation of data memory and program memory. This microcontroller has separate address spaces for program and data memory. Because the program and data memory are logically separated, it causes the data memory to be accessed by 8 bit addresses, which can be stored and manipulated by an 8 bit processor more quickly.

#### **3.3.2.1 PROGRAM MEMORY**

The processor begins the execution of the programs in the memory from the address 0000h. Every interrupt is assigned a separate address space in the memory i.e., the interrupt service routines for every external interrupt is assigned a fixed address space and therefore, when an interrupt occurs, the control of program execution is switched to the location, in accordance with the program that's being executed or to the service routine of the interrupt that has occurred, based on the priority assigned.

The service routines for the interrupts are placed at a location in the memory, and these addresses are stored in the processor so that whenever an interrupt occurs, the processor switches the control automatically.



**Fig 3.e Program Memory**

Program memory can only be read. There can be up to 64K bytes of directly addressable program memory. The read strobe for external program memory is the Program Store Enable Signal (PSEN) Data memory occupies a separate address space from program memory. Up to 64K bytes of external memory can be directly addressed in the external data memory space. The CPU generates read and write signals, RD and Wr, during external data memory accesses. External program memory and external data memory can be combined by applying the RD and PSEN signals to the inputs of AND gate and using the output of the gate as the read strobe to the external program/data memory.

### 3.3.2.2 DATA MEMORY

The Internal Data memory is divided into three blocks namely,

- ❖ The lower 128 Bytes of Internal RAM.
- ❖ The Upper 128 Bytes of Internal RAM.
- ❖ Special Function Register.

Internal Data memory Addresses are always 1 byte wide, which implies an address space of only 256 bytes. However, the addressing modes for internal RAM can in fact accommodate 384 bytes. Direct addresses higher than 7Fh access one memory space and indirect addresses higher than 7Fh access a different Memory Space.

The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) Select, which register bank, is in use. This architecture allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing.

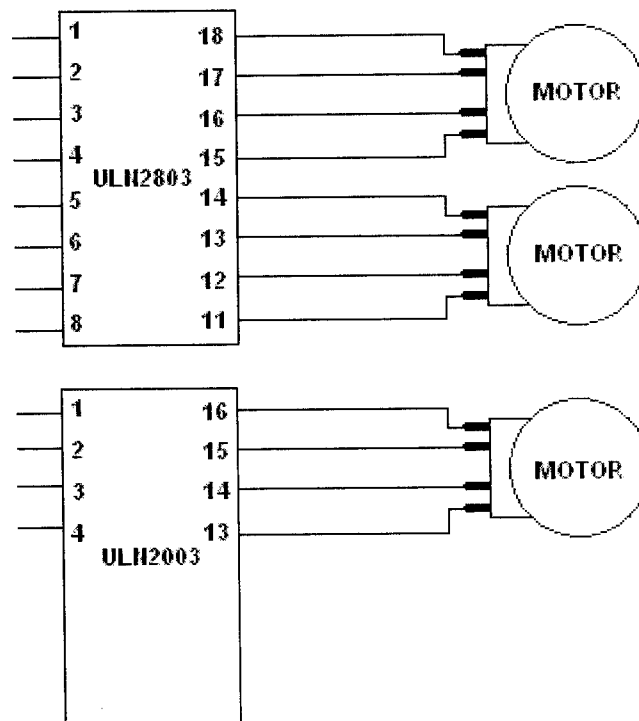
The next 16-bytes above the register banks form a block of bit addressable memory space. The micro controller instruction set includes a wide selection of single - bit instructions and this instruction can directly address the 128 bytes in this area. These bit addresses are 00h through 7Fh. either direct or indirect addressing can access all of the bytes in lower 128 bytes. Indirect addressing can only access the upper 128. The upper 128 bytes of RAM are only in the devices with 256 bytes of RAM.

The Special Function Register includes Ports latches, timers, peripheral controls etc., direct addressing can only access these register. In general, all Atmel micro controllers have the same SFRs at the same addresses in SFR space as the AT89C51 and other compatible micro controllers. However, upgrades to the AT89C51 have additional SFRs. Sixteen addresses in SFR space are both byte and bit Addressable. The bit Addressable SFRs are those whose address ends in 000B. The bit addresses in this area are 80h through FFh.

### 3.4 DRIVING UNIT

The ULN2003, high voltage, high current Darlington array each containing seven open collector Darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

The ULN2003, used have 16 pin plastic DIP packages with a copper lead frame to reduce thermal resistance.

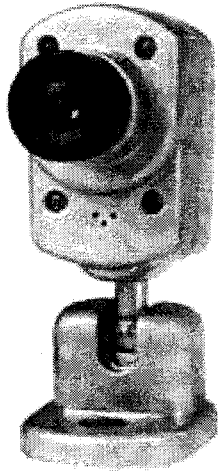


**Fig 3.f** Interfacing of Driving Unit and Motor

The signals from the microcontroller are fed to ULN2003 and are then used to drive the stepper motors.

### 3.5 CAMERA

#### WS-809AS (CMOS Camera)



WS-809AS Color / wireless cam  
WS-809B B/W Camera  
380 TV Lines  
Metal Case

Fig 3.g WS – 809 AS CMOS Camera



Fig 3.h Video signals – Receiver

### 3.5.1 DESCRIPTION

High quality Sound, Super high resolution and sensitivity ,Integrated lens can be adjusted ,Low consumption and excellent stabilization ,Compact and easy to install Fit for the security system of stores, banks, hotels, school, hospitals, factories and apartments.

	<b>Color</b>	<b>B/W</b>
<b>Picture Element</b>	PAL: 628X582 NTSC:510X492	EIA(NTSC):320X240 CCIR(PAL):352X268
<b>Electronic Shutter</b>	1/60-1/15000's	1/50 - 1/6000
<b>S/N Ratio</b>	48dB	46dB
<b>Horizontal Resolution</b>	382 TV LINES	380 TV LINES
<b>Video Output</b>	1Vp-p 75ohm	1Vp-p 75ohm
<b>Min Illumination</b>	2LUX	0.2LUX
<b>Backlight Opensation</b>	AUTO	AUTO
<b>Operating Temp</b>	-10;æ--+50;æ	-10;æ-- +50;æ
<b>Power Supply</b>	DC6--12V	DC6--12V
<b>Power Consumption</b>	50mA	55mA (with infrared ray)
<b>IC</b>	OV7910	OV5116

**Table 3.a** Specification of WS – 809 AS CMOS Camera

### 3.5.2 FEATURES

1. High quality Sound
2. Super high resolution and sensitivity
3. Integrated lens can be adjusted
4. Low consumption and excellent stabilization
5. Compact and easy to install
6. Fit for the security system of stores, banks, hotels, school, hospitals, factories and apartments.

## **4. CONCLUSION**

With the incursion of wireless systems, applications and services in our day to day life, the model of a robot we have designed aids in the research field. Also, with further enhancements more and more applications could be embedded in our model by changing few of its present components, so that it would provide support for the new applications. When such enhancements are done, our project could be used as a model to build a robot in real life. Exploration of space involves designing of robots to scan the information about the destination; this project would well serve this purpose. Also, our project could be used in 'home security' system, where the vehicle could be controlled by a system connected to the vehicle through an intermediate system, connected in a network. In the same way our project with little modification could be use as a model to build a robot, which could be used to scrutinize real time industrial applications.

Thus our project increases mobility of service in addition to the idea of the model we have suggested.

## 5. APPENDICES

### 5.1 APPENDIX 1 – SAMPLE CODE

#### MICROCONTROLLER PROGRAM

```
fwd          equ 01h    ;forward
rev          equ 02h    ;reverse
right       equ 03h    ;right
left        equ 04h    ;left
cright      equ 05h    ;camera right
cleft       equ 06h    ;camera left
```

```
sense       equ 84h
```

```
light       equ 0b6h
```

```
org 0000h
```

```
mov p3,#00h
mov p1,#00h
mov p0,#0ffh
```

```
mov a,#00h
```

```
l1          nop
           mov p3,#00h
           jnb sense,l1
```

```
           mov a,p0
           anl a,#0fh
```

```
           mov p3,a
```

```
           cjne a,#fwd,n1
```

```
           mov p1,#59h
           acall frdelay
           mov p1,#53h
           acall frdelay
           mov p1,#56h
           acall frdelay
           mov p1,#5ch
```



```

acall frdelay

mov p1,#00h

n1      cjne a,#rev,n2

        mov p1,#5ch
        acall frdelay
        mov p1,#56h
        acall frdelay
        mov p1,#53h
        acall frdelay
        mov p1,#59h
        acall frdelay

        mov p1,#00h

n2      cjne a,#right,n3

        mov p1,#90h
        acall delay
        mov p1,#30h
        acall delay
        mov p1,#60h
        acall delay
        mov p1,#0c0h
        acall delay

        ;mov p1,#00h

n3      cjne a,#left,n4

        mov p1,#0c0h
        acall delay
        mov p1,#60h
        acall delay
        mov p1,#30h
        acall delay
        mov p1,#90h
        acall delay

        ;mov p1,#00h

n4      cjne a,#cleft,n5

```

```
mov p3,#01h
setb light
acall delay
```

```
mov p3,#02h
setb light
acall delay
```

```
mov p3,#04h
setb light
acall delay
```

```
mov p3,#08h
setb light
acall delay
```

```
clr light
```

```
;mov p1,#00h
```

```
n5      cjne a,#cright,n6
```

```
mov p3,#08h
setb light
acall delay
```

```
mov p3,#04h
setb light
acall delay
```

```
mov p3,#02h
setb light
acall delay
```

```
mov p3,#01h
setb light
acall delay
```

```
clr light
```

```
;mov p1,#00h
```

```
n6      ajmp l1
```

```
frdelay  mov r0,#01h
```

```
frin2      mov r1,#10h
frin1      mov r2,#70h
frwait     djnz r2,frwait
           djnz r1,frin1
           djnz r0,frin2
           ret
```

```
delay     mov r0,#01h
in2       mov r1,#00h
in1       mov r2,#00h
dwait     djnz r2,dwait
           djnz r1,in1
           djnz r0,in2
           ret
```

## SAMPLE VISUAL BASIC CODE

### Mainform.frm

Option Explicit

```
Private Sub Command3_MouseDown(Index As Integer, Button As Integer, Shift  
As Integer, x As Single, y As Single)  
Out &H378, 2 ^ Index  
End Sub
```

```
Private Sub Command3_MouseUp(Index As Integer, Button As Integer, Shift As  
Integer, x As Single, y As Single)  
Out &H378, 0  
End Sub
```

```
Private Sub Form_Load()  
    Dim lpszName As String * 100  
    Dim lpszVer As String * 100  
    Dim Caps As CAPDRIVERCAPS  
    capGetDriverDescriptionA 0, lpszName, 100, lpszVer, 100  
    lwndC = capCreateCaptureWindowA(lpszName, WS_CAPTION Or  
WS_THICKFRAME Or WS_VISIBLE Or WS_CHILD, 0, 0, 160, 120, Me.hwnd,  
0)  
    SetWindowText lwndC, lpszName  
    capSetCallbackOnStatus lwndC, AddressOf MyStatusCallback  
    capSetCallbackOnError lwndC, AddressOf MyErrorCallback  
    If capDriverConnect(lwndC, 0) Then  
        capDriverGetCaps lwndC, VarPtr(Caps), Len(Caps)  
        If Caps.fHasDlgVideoSource = 0 Then mnuSource.Enabled = False  
        If Caps.fHasDlgVideoFormat = 0 Then mnuFormat.Enabled = False  
        If Caps.fHasDlgVideoDisplay = 0 Then mnuDisplay.Enabled = False
```

```
capPreviewScale lwndC, True
capPreviewRate lwndC, 66
capPreview lwndC, True
ResizeCaptureWindow lwndC
```

```
End If
```

```
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
```

```
capSetCallbackOnError lwndC, vbNull
capSetCallbackOnStatus lwndC, vbNull
capSetCallbackOnYield lwndC, vbNull
capSetCallbackOnFrame lwndC, vbNull
capSetCallbackOnVideoStream lwndC, vbNull
capSetCallbackOnWaveStream lwndC, vbNull
capSetCallbackOnCapControl lwndC, vbNull
```

```
End Sub
```

```
Private Sub mnuSource_Click()
```

```
capDlgVideoSource lwndC
```

```
End Sub
```

```
Private Sub mnuStart_Click()
```

```
Dim sFileName As String
```

```
Dim CAP_PARAMS As CAPTUREPARMS
```

```
capCaptureGetSetup lwndC, VarPtr(CAP_PARAMS),
```

```
Len(CAP_PARAMS)
```

```
CAP_PARAMS.dwRequestMicroSecPerFrame = (1 * (10 ^ 6)) / 30 ' 30
```

```
Frames per second
```

```
CAP_PARAMS.fMakeUserHitOKToCapture = True
```

```
CAP_PARAMS.fCaptureAudio = False
```

```
capCaptureSetSetup lwndC, VarPtr(CAP_PARAMS),  
Len(CAP_PARAMS)  
sFileName = "C:\myvideo.avi"  
capCaptureSequence lwndC  
capFileSaveAs lwndC, sFileName  
End Sub
```

## Vbavicap.bas

```
Public Const WM_USER = &H400
```

```
Type POINTAPI
```

```
    x As Long
```

```
    y As Long
```

```
End Type
```

```
Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd  
As Long, ByVal wParam As Long, ByVal lParam As  
Long) As Long
```

```
Declare Function SendMessageS Lib "user32" Alias "SendMessageA" (ByVal  
hwnd As Long, ByVal wParam As Long, ByVal lParam As Integer, ByVal lParam  
As String) As Long
```

```
Public Const WM_CAP_START = WM_USER
```

```
Public Const WM_CAP_GET_CAPSTREAMPTR = WM_CAP_START + 1
```

```
Public Const WM_CAP_SET_CALLBACK_ERROR = WM_CAP_START + 2
```

```
Public Const WM_CAP_SET_CALLBACK_STATUS = WM_CAP_START + 3
```

```
Public Const WM_CAP_SET_CALLBACK_YIELD = WM_CAP_START + 4
```

```
Public Const WM_CAP_SET_CALLBACK_FRAME = WM_CAP_START + 5
```

```
Public Const WM_CAP_SET_CALLBACK_VIDEOSTREAM =  
WM_CAP_START + 6
```

```
Public Const WM_CAP_SET_CALLBACK_WAVESTREAM =  
WM_CAP_START + 7
```

```
Public Const WM_CAP_GET_USER_DATA = WM_CAP_START + 8
```

```
Public Const WM_CAP_SET_USER_DATA = WM_CAP_START + 9
```

```
Public Const WM_CAP_DRIVER_CONNECT = WM_CAP_START + 10
```

Public Const WM\_CAP\_DRIVER\_DISCONNECT = WM\_CAP\_START + 11

Public Const WM\_CAP\_DRIVER\_GET\_NAME = WM\_CAP\_START + 12

Public Const WM\_CAP\_DRIVER\_GET\_VERSION = WM\_CAP\_START + 13

Public Const WM\_CAP\_DRIVER\_GET\_CAPS = WM\_CAP\_START + 14

Public Const WM\_CAP\_FILE\_SET\_CAPTURE\_FILE = WM\_CAP\_START + 20

Public Const WM\_CAP\_FILE\_GET\_CAPTURE\_FILE = WM\_CAP\_START + 21

Public Const WM\_CAP\_FILE\_ALLOCATE = WM\_CAP\_START + 22

Public Const WM\_CAP\_FILE\_SAVEAS = WM\_CAP\_START + 23

Public Const WM\_CAP\_FILE\_SET\_INFOCHUNK = WM\_CAP\_START + 24

Public Const WM\_CAP\_FILE\_SAVEDIB = WM\_CAP\_START + 25

Public Const WM\_CAP\_EDIT\_COPY = WM\_CAP\_START + 30

Public Const WM\_CAP\_SET\_AUDIOFORMAT = WM\_CAP\_START + 35

Public Const WM\_CAP\_GET\_AUDIOFORMAT = WM\_CAP\_START + 36

Public Const WM\_CAP\_DLG\_VIDEOFORMAT = WM\_CAP\_START + 41

Public Const WM\_CAP\_DLG\_VIDEOSOURCE = WM\_CAP\_START + 42

Public Const WM\_CAP\_DLG\_VIDEODISPLAY = WM\_CAP\_START + 43

Public Const WM\_CAP\_GET\_VIDEOFORMAT = WM\_CAP\_START + 44

Public Const WM\_CAP\_SET\_VIDEOFORMAT = WM\_CAP\_START + 45

Public Const WM\_CAP\_DLG\_VIDEOCOMPRESSION = WM\_CAP\_START +  
46

Public Const WM\_CAP\_SET\_PREVIEW = WM\_CAP\_START + 50

Public Const WM\_CAP\_SET\_OVERLAY = WM\_CAP\_START + 51

Public Const WM\_CAP\_SET\_PREVIEWRATE = WM\_CAP\_START + 52

Public Const WM\_CAP\_SET\_SCALE = WM\_CAP\_START + 53

Public Const WM\_CAP\_GET\_STATUS = WM\_CAP\_START + 54



Public Const WM\_CAP\_SET\_SCROLL = WM\_CAP\_START + 55

Public Const WM\_CAP\_GRAB\_FRAME = WM\_CAP\_START + 60

Public Const WM\_CAP\_GRAB\_FRAME\_NOSTOP = WM\_CAP\_START + 61

Public Const WM\_CAP\_SEQUENCE = WM\_CAP\_START + 62

Public Const WM\_CAP\_SEQUENCE\_NOFILE = WM\_CAP\_START + 63

Public Const WM\_CAP\_SET\_SEQUENCE\_SETUP = WM\_CAP\_START + 64

Public Const WM\_CAP\_GET\_SEQUENCE\_SETUP = WM\_CAP\_START + 65

Public Const WM\_CAP\_SET\_MCI\_DEVICE = WM\_CAP\_START + 66

Public Const WM\_CAP\_GET\_MCI\_DEVICE = WM\_CAP\_START + 67

Public Const WM\_CAP\_STOP = WM\_CAP\_START + 68

Public Const WM\_CAP\_ABORT = WM\_CAP\_START + 69

Public Const WM\_CAP\_SINGLE\_FRAME\_OPEN = WM\_CAP\_START + 70

Public Const WM\_CAP\_SINGLE\_FRAME\_CLOSE = WM\_CAP\_START + 71

Public Const WM\_CAP\_SINGLE\_FRAME = WM\_CAP\_START + 72

Public Const WM\_CAP\_PAL\_OPEN = WM\_CAP\_START + 80

Public Const WM\_CAP\_PAL\_SAVE = WM\_CAP\_START + 81

Public Const WM\_CAP\_PAL\_PASTE = WM\_CAP\_START + 82

Public Const WM\_CAP\_PAL\_AUTOCREATE = WM\_CAP\_START + 83

Public Const WM\_CAP\_PAL\_MANUALCREATE = WM\_CAP\_START + 84

Public Const WM\_CAP\_SET\_CALLBACK\_CAPCONTROL =

WM\_CAP\_START + 85

Public Const WM\_CAP\_END = WM\_CAP\_SET\_CALLBACK\_CAPCONTROL

Type CAPDRIVERCAPS

wDeviceIndex As Long  
fHasOverlay As Long  
fHasDlgVideoSource As Long  
fHasDlgVideoFormat As Long  
fHasDlgVideoDisplay As Long  
fCaptureInitialized As Long  
fDriverSuppliesPalettes As Long  
hVideoIn As Long  
hVideoOut As Long  
hVideoExtIn As Long  
hVideoExtOut As Long

End Type

Type CAPSTATUS

uiImageWidth As Long  
uiImageHeight As Long  
fLiveWindow As Long  
fOverlayWindow As Long  
fScale As Long  
ptScroll As POINTAPI  
fUsingDefaultPalette As Long  
fAudioHardware As Long  
fCapFileExists As Long  
dwCurrentVideoFrame As Long  
dwCurrentVideoFramesDropped As Long  
dwCurrentWaveSamples As Long  
dwCurrentTimeElapsedMS As Long  
hPalCurrent As Long  
fCapturingNow As Long  
dwReturn As Long

wNumVideoAllocated As Long

wNumAudioAllocated As Long

End Type

Type CAPTUREPARMS

dwRequestMicroSecPerFrame As Long

fMakeUserHitOKToCapture As Long

wPercentDropForError As Long

fYield As Long

dwIndexSize As Long

wChunkGranularity As Long

fUsingDOSMemory As Long

wNumVideoRequested As Long

fCaptureAudio As Long

wNumAudioRequested As Long

vKeyAbort As Long

fAbortLeftMouse As Long

fAbortRightMouse As Long

fLimitEnabled As Long

wTimeLimit As Long

fMCIControl As Long

fStepMCIDevice As Long

dwMCIStartTime As Long

dwMCIStopTime As Long

fStepCaptureAt2x As Long

wStepCaptureAverageFrames As Long

dwAudioBufferSize As Long

fDisableWriteCache As Long

End Type

Type CAPINFOCHUNK

fccInfoID As Long

lpData As Long

cbData As Long

End Type

Type VIDEOHDR

lpData As Long

dwBufferLength As Long

dwBytesUsed As Long

dwTimeCaptured As Long

dwUser As Long

dwFlags As Long

dwReserved(3) As Long

End Type

Declare Function capCreateCaptureWindowA Lib "avicap32.dll" ( \_

ByVal lpszWindowName As String, \_

ByVal dwStyle As Long, \_

ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight

As Integer, \_

ByVal hWndParent As Long, ByVal nID As Long) As Long

Declare Function capGetDriverDescriptionA Lib "avicap32.dll" ( \_

ByVal wDriver As Integer, \_

ByVal lpszName As String, \_

ByVal cbName As Long, \_

ByVal lpszVer As String, \_

ByVal cbVer As Long) As Boolean

Public Const IDS\_CAP\_BEGIN = 300

Public Const IDS\_CAP\_END = 301

Public Const IDS\_CAP\_INFO = 401  
Public Const IDS\_CAP\_OUTOFMEM = 402  
Public Const IDS\_CAP\_FILEEXISTS = 403  
Public Const IDS\_CAP\_ERRORPALOPEN = 404  
Public Const IDS\_CAP\_ERRORPALSAVE = 405  
Public Const IDS\_CAP\_ERRORDIBSAVE = 406  
Public Const IDS\_CAP\_DEFAVIEXT = 407  
Public Const IDS\_CAP\_DEFPALEXT = 408  
Public Const IDS\_CAP\_CANTOPEN = 409  
Public Const IDS\_CAP\_SEQ\_MSGSTART = 410  
Public Const IDS\_CAP\_SEQ\_MSGSTOP = 411  
Public Const IDS\_CAP\_VIDEDITERR = 412  
Public Const IDS\_CAP\_READONLYFILE = 413  
Public Const IDS\_CAP\_WRITEERROR = 414  
Public Const IDS\_CAP\_NODISKSPACE = 415  
Public Const IDS\_CAP\_SETFILESIZE = 416  
Public Const IDS\_CAP\_SAVEASPERCENT = 417  
Public Const IDS\_CAP\_DRIVER\_ERROR = 418  
Public Const IDS\_CAP\_WAVE\_OPEN\_ERROR = 419  
Public Const IDS\_CAP\_WAVE\_ALLOC\_ERROR = 420  
Public Const IDS\_CAP\_WAVE\_PREPARE\_ERROR = 421  
Public Const IDS\_CAP\_WAVE\_ADD\_ERROR = 422  
Public Const IDS\_CAP\_WAVE\_SIZE\_ERROR = 423  
Public Const IDS\_CAP\_VIDEO\_OPEN\_ERROR = 424  
Public Const IDS\_CAP\_VIDEO\_ALLOC\_ERROR = 425  
Public Const IDS\_CAP\_VIDEO\_PREPARE\_ERROR = 426  
Public Const IDS\_CAP\_VIDEO\_ADD\_ERROR = 427  
Public Const IDS\_CAP\_VIDEO\_SIZE\_ERROR = 428  
Public Const IDS\_CAP\_FILE\_OPEN\_ERROR = 429  
Public Const IDS\_CAP\_FILE\_WRITE\_ERROR = 430

```

Public Const IDS_CAP_RECORDING_ERROR = 431
Public Const IDS_CAP_RECORDING_ERROR2 = 432
Public Const IDS_CAP_AVI_INIT_ERROR = 433
Public Const IDS_CAP_NO_FRAME_CAP_ERROR = 434
Public Const IDS_CAP_NO_PALETTE_WARN = 435
Public Const IDS_CAP_MCI_CONTROL_ERROR = 436
Public Const IDS_CAP_MCI_CANT_STEP_ERROR = 437
Public Const IDS_CAP_NO_AUDIO_CAP_ERROR = 438
Public Const IDS_CAP_AVI_DRAWDIB_ERROR = 439
Public Const IDS_CAP_COMPRESSOR_ERROR = 440
Public Const IDS_CAP_AUDIO_DROP_ERROR = 441
Public Const IDS_CAP_STAT_LIVE_MODE = 500
Public Const IDS_CAP_STAT_OVERLAY_MODE = 501
Public Const IDS_CAP_STAT_CAP_INIT = 502
Public Const IDS_CAP_STAT_CAP_FINI = 503
Public Const IDS_CAP_STAT_PALETTE_BUILD = 504
Public Const IDS_CAP_STAT_OPTPAL_BUILD = 505
Public Const IDS_CAP_STAT_I_FRAMES = 506
Public Const IDS_CAP_STAT_L_FRAMES = 507
Public Const IDS_CAP_STAT_CAP_L_FRAMES = 508
Public Const IDS_CAP_STAT_CAP_AUDIO = 509
Public Const IDS_CAP_STAT_VIDEOCURRENT = 510
Public Const IDS_CAP_STAT_VIDEOAUDIO = 511
Public Const IDS_CAP_STAT_VIDEOONLY = 512
Function capSetCallbackOnError(ByVal lwnd As Long, ByVal lpProc As Long)
As Boolean
    capSetCallbackOnError = SendMessage(lwnd,
WM_CAP_SET_CALLBACK_ERROR, 0, lpProc)
End Function

```

```
Function capSetCallbackOnStatus(ByVal hwnd As Long, ByVal lpProc As Long)
As Boolean
```

```
    capSetCallbackOnStatus = SendMessage(hwnd,
WM_CAP_SET_CALLBACK_STATUS, 0, lpProc)
```

```
End Function
```

```
Function capSetCallbackOnYield(ByVal hwnd As Long, ByVal lpProc As Long)
As Boolean
```

```
    capSetCallbackOnYield = SendMessage(hwnd,
WM_CAP_SET_CALLBACK_YIELD, 0, lpProc)
```

```
End Function
```

```
Function capSetCallbackOnFrame(ByVal hwnd As Long, ByVal lpProc As Long)
As Boolean
```

```
    capSetCallbackOnFrame = SendMessage(hwnd,
WM_CAP_SET_CALLBACK_FRAME, 0, lpProc)
```

```
End Function
```

```
Function capSetCallbackOnVideoStream(ByVal hwnd As Long, ByVal lpProc As
Long) As Boolean
```

```
    capSetCallbackOnVideoStream = SendMessage(hwnd,
WM_CAP_SET_CALLBACK_VIDESTREAM, 0, lpProc)
```

```
End Function
```

```
Function capSetCallbackOnWaveStream(ByVal hwnd As Long, ByVal lpProc As
Long) As Boolean
```

```
    capSetCallbackOnWaveStream = SendMessage(hwnd,
WM_CAP_SET_CALLBACK_WAVESTREAM, 0, lpProc)
```

```
End Function
```

```
Function capSetCallbackOnCapControl(ByVal hwnd As Long, ByVal lpProc As
Long) As Boolean
```

```
    capSetCallbackOnCapControl = SendMessage(hwnd,
WM_CAP_SET_CALLBACK_CAPCONTROL, 0, lpProc)
```

```
End Function
```

```

Function capSetUserData(ByVal hwnd As Long, ByVal IUser As Long) As
Boolean
    capSetUserData = SendMessage(hwnd, WM_CAP_SET_USER_DATA, 0, IUser)
End Function

Function capGetUserData(ByVal hwnd As Long) As Long
    capGetUserData = SendMessage(hwnd, WM_CAP_GET_USER_DATA, 0, 0)
End Function

Function capDriverConnect(ByVal hwnd As Long, ByVal i As Integer) As Boolean
    capDriverConnect = SendMessage(hwnd, WM_CAP_DRIVER_CONNECT, i, 0)
End Function

Function capDriverDisconnect(ByVal hwnd As Long) As Boolean
    capDriverDisconnect = SendMessage(hwnd,
WM_CAP_DRIVER_DISCONNECT, 0, 0)
End Function

Function capDriverGetName(ByVal hwnd As Long, ByVal szName As Long,
ByVal wSize As Integer) As Boolean
    capDriverGetName = SendMessage(hwnd, YOURCONSTANTMESSAGE,
wSize, szName)
End Function

Function capDriverGetVersion(ByVal hwnd As Long, ByVal szVer As Long,
ByVal wSize As Integer) As Boolean
    capDriverGetVersion = SendMessage(hwnd,
WM_CAP_DRIVER_GET_VERSION, wSize, szVer)
End Function

Function capDriverGetCaps(ByVal hwnd As Long, ByVal s As Long, ByVal wSize
As Integer) As Boolean
    capDriverGetCaps = SendMessage(hwnd, WM_CAP_DRIVER_GET_CAPS,
wSize, s)
End Function

```



```

Function capFileSetCaptureFile(ByVal lwnd As Long, szName As String) As
Boolean
    capFileSetCaptureFile = SendMessageS(lwnd,
WM_CAP_FILE_SET_CAPTURE_FILE, 0, szName)
End Function

Function capFileGetCaptureFile(ByVal lwnd As Long, ByVal szName As Long,
wSize As String) As Boolean
    capFileGetCaptureFile = SendMessageS(lwnd,
WM_CAP_FILE_SET_CAPTURE_FILE, wSize, szName)
End Function

Function capFileAlloc(ByVal lwnd As Long, ByVal dwSize As Long) As Boolean
    capFileAlloc = SendMessage(lwnd, WM_CAP_FILE_ALLOCATE, 0, dwSize)
End Function

Function capFileSaveAs(ByVal lwnd As Long, szName As String) As Boolean
    capFileSaveAs = SendMessageS(lwnd, WM_CAP_FILE_SAVEAS, 0, szName)
End Function

Function capFileSetInfoChunk(ByVal lwnd As Long, ByVal lpInfoChunk As
Long) As Boolean
    capFileSetInfoChunk = SendMessage(lwnd,
WM_CAP_FILE_SET_INFOCHUNK, 0, lpInfoChunk)
End Function

Function capFileSaveDIB(ByVal lwnd As Long, ByVal szName As Long) As
Boolean
    capFileSaveDIB = SendMessage(lwnd, WM_CAP_FILE_SAVEDIB, 0,
szName)
End Function

Function capEditCopy(ByVal lwnd As Long) As Boolean
    capEditCopy = SendMessage(lwnd, WM_CAP_EDIT_COPY, 0, 0)
End Function

```

```

Function capSetAudioFormat(ByVal hwnd As Long, ByVal s As Long, ByVal
wSize As Integer) As Boolean
    capSetAudioFormat = SendMessage(hwnd, WM_CAP_SET_AUDIOFORMAT,
wSize, s)
End Function
Function capGetAudioFormat(ByVal hwnd As Long, ByVal s As Long, ByVal
wSize As Integer) As Long
    capGetAudioFormat = SendMessage(hwnd, WM_CAP_GET_AUDIOFORMAT,
wSize, s)
End Function
Function capGetAudioFormatSize(ByVal hwnd As Long) As Long
    capGetAudioFormatSize = SendMessage(hwnd,
WM_CAP_GET_AUDIOFORMAT, 0, 0)
End Function
Function capDlgVideoFormat(ByVal hwnd As Long) As Boolean
    capDlgVideoFormat = SendMessage(hwnd, WM_CAP_DLG_VIDEOFORMAT,
0, 0)
End Function
Function capDlgVideoSource(ByVal hwnd As Long) As Boolean
    capDlgVideoSource = SendMessage(hwnd, WM_CAP_DLG_VIDEOSOURCE,
0, 0)
End Function
Function capDlgVideoDisplay(ByVal hwnd As Long) As Boolean
    capDlgVideoDisplay = SendMessage(hwnd, WM_CAP_DLG_VIDEODISPLAY,
0, 0)
End Function
Function capDlgVideoCompression(ByVal hwnd As Long) As Boolean
    capDlgVideoCompression = SendMessage(hwnd,
WM_CAP_DLG_VIDEOCOMPRESSION, 0, 0)
End Function

```

Function capGetVideoFormat(ByVal hwnd As Long, ByVal s As Long, ByVal  
wSize As Integer) As Long

capGetVideoFormat = SendMessage(hwnd, WM\_CAP\_GET\_VIDEOFORMAT,  
wSize, s)

End Function

Function capGetVideoFormatSize(ByVal hwnd As Long) As Long

capGetVideoFormatSize = SendMessage(hwnd,  
WM\_CAP\_GET\_VIDEOFORMAT, 0, 0)

End Function

Function capSetVideoFormat(ByVal hwnd As Long, ByVal s As Long, ByVal  
wSize As Integer) As Boolean

capSetVideoFormat = SendMessage(hwnd, WM\_CAP\_SET\_VIDEOFORMAT,  
wSize, s)

End Function

Function capPreview(ByVal hwnd As Long, ByVal f As Boolean) As Boolean

capPreview = SendMessage(hwnd, WM\_CAP\_SET\_PREVIEW, f, 0)

End Function

Function capPreviewRate(ByVal hwnd As Long, ByVal wMS As Integer) As  
Boolean

capPreviewRate = SendMessage(hwnd, WM\_CAP\_SET\_PREVIEWRATE,  
wMS, 0)

End Function

Function capOverlay(ByVal hwnd As Long, ByVal f As Boolean) As Boolean

capOverlay = SendMessage(hwnd, WM\_CAP\_SET\_OVERLAY, f, 0)

End Function

Function capPreviewScale(ByVal hwnd As Long, ByVal f As Boolean) As Boolean

capPreviewScale = SendMessage(hwnd, WM\_CAP\_SET\_SCALE, f, 0)

End Function

Function capGetStatus(ByVal hwnd As Long, ByVal s As Long, ByVal wSize As  
Integer) As Boolean

```

    capGetStatus = SendMessage(lwnd, WM_CAP_GET_STATUS, wSize, s)
End Function
Function capSetScrollPos(ByVal lwnd As Long, ByVal lpP As Long) As Boolean
    capSetScrollPos = SendMessage(lwnd, WM_CAP_SET_SCROLL, 0, lpP)
End Function
Function capGrabFrame(ByVal lwnd As Long) As Boolean
    capGrabFrame = SendMessage(lwnd, WM_CAP_GRAB_FRAME, 0, 0)
End Function
Function capGrabFrameNoStop(ByVal lwnd As Long) As Boolean
    capGrabFrameNoStop = SendMessage(lwnd,
WM_CAP_GRAB_FRAME_NOSTOP, 0, 0)
End Function
Function capCaptureSequence(ByVal lwnd As Long) As Boolean
    capCaptureSequence = SendMessage(lwnd, WM_CAP_SEQUENCE, 0, 0)
End Function
Function capCaptureSequenceNoFile(ByVal lwnd As Long) As Boolean
    capCaptureSequenceNoFile = SendMessage(lwnd,
WM_CAP_SEQUENCE_NOFILE, 0, 0)
End Function
Function capCaptureStop(ByVal lwnd As Long) As Boolean
    capCaptureStop = SendMessage(lwnd, WM_CAP_STOP, 0, 0)
End Function
Function capCaptureAbort(ByVal lwnd As Long) As Boolean
    capCaptureAbort = SendMessage(lwnd, WM_CAP_ABORT, 0, 0)
End Function
Function capCaptureSingleFrameOpen(ByVal lwnd As Long) As Boolean
    capCaptureSingleFrameOpen = SendMessage(lwnd,
WM_CAP_SINGLE_FRAME_OPEN, 0, 0)
End Function
Function capCaptureSingleFrameClose(ByVal lwnd As Long) As Boolean

```

```

    capCaptureSingleFrameClose = SendMessage(lwnd,
WM_CAP_SINGLE_FRAME_CLOSE, 0, 0)
End Function
Function capCaptureSingleFrame(ByVal lwnd As Long) As Boolean
    capCaptureSingleFrame = SendMessage(lwnd, WM_CAP_SINGLE_FRAME, 0,
0)
End Function
Function capCaptureGetSetup(ByVal lwnd As Long, ByVal s As Long, ByVal
wSize As Integer) As Boolean
    capCaptureGetSetup = SendMessage(lwnd,
WM_CAP_GET_SEQUENCE_SETUP, wSize, s)
End Function
Function capCaptureSetSetup(ByVal lwnd As Long, ByVal s As Long, ByVal
wSize As Integer) As Boolean
    capCaptureSetSetup = SendMessage(lwnd,
WM_CAP_SET_SEQUENCE_SETUP, wSize, s)
End Function
Function capSetMCIDeviceName(ByVal lwnd As Long, ByVal szName As Long)
As Boolean
    capSetMCIDeviceName = SendMessage(lwnd, WM_CAP_SET_MCI_DEVICE,
0, szName)
End Function
Function capGetMCIDeviceName(ByVal lwnd As Long, ByVal szName As Long,
ByVal wSize As Integer) As Boolean
    capGetMCIDeviceName = SendMessage(lwnd,
WM_CAP_GET_MCI_DEVICE, wSize, szName)
End Function
Function capPaletteOpen(ByVal lwnd As Long, ByVal szName As Long) As
Boolean
    capPaletteOpen = SendMessage(lwnd, WM_CAP_PAL_OPEN, 0, szName)

```

End Function

Function capPaletteSave(ByVal lwnd As Long, ByVal szName As Long) As Boolean

capPaletteSave = SendMessage(lwnd, WM\_CAP\_PAL\_SAVE, 0, szName)

End Function

Function capPalettePaste(ByVal lwnd As Long) As Boolean

capPalettePaste = SendMessage(lwnd, WM\_CAP\_PAL\_PASTE, 0, 0)

End Function

Function capPaletteAuto(ByVal lwnd As Long, ByVal iFrames As Integer, ByVal iColor As Long) As Boolean

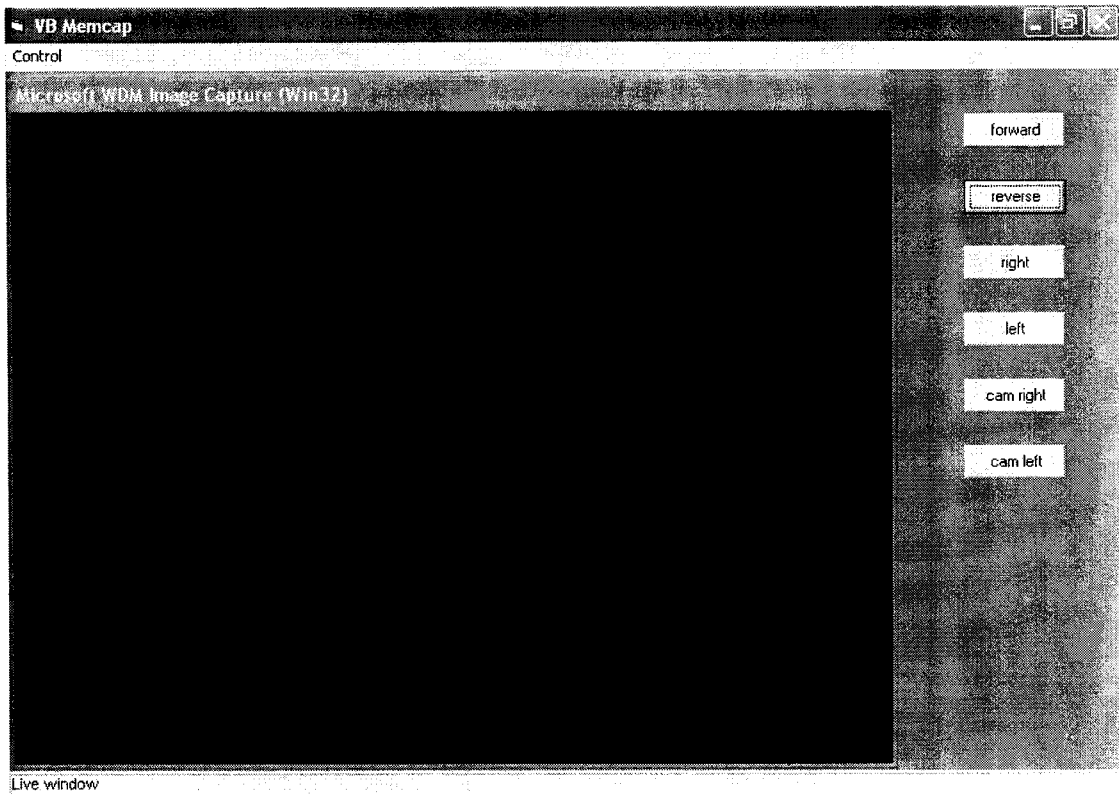
capPaletteAuto = SendMessage(lwnd, WM\_CAP\_PAL\_AUTOCREATE, iFrames, iColors)

End Function

Function capPaletteManual(ByVal lwnd As Long, ByVal fGrab As Boolean, ByVal iColors As Long) As Boolean

capPaletteManual = SendMessage(lwnd, WM\_CAP\_PAL\_MANUALCREATE, fGrab, iColors)

## 5.2 APPENDIX 2 - SNAP SHOTS



## 5.3 APPENDIX 3 – DATA SHEETS

### AT89C51

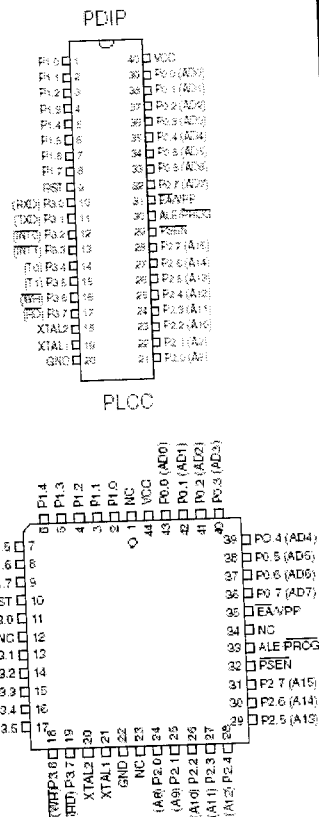
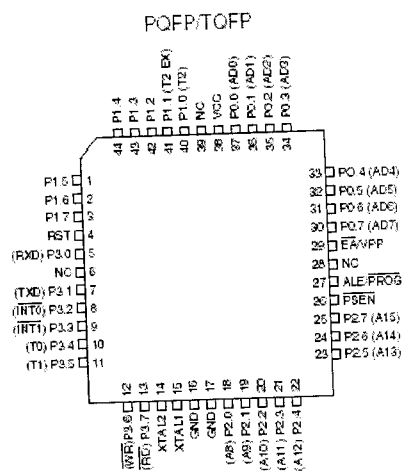
#### Features

- 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

#### 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 Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

#### Pin Configurations



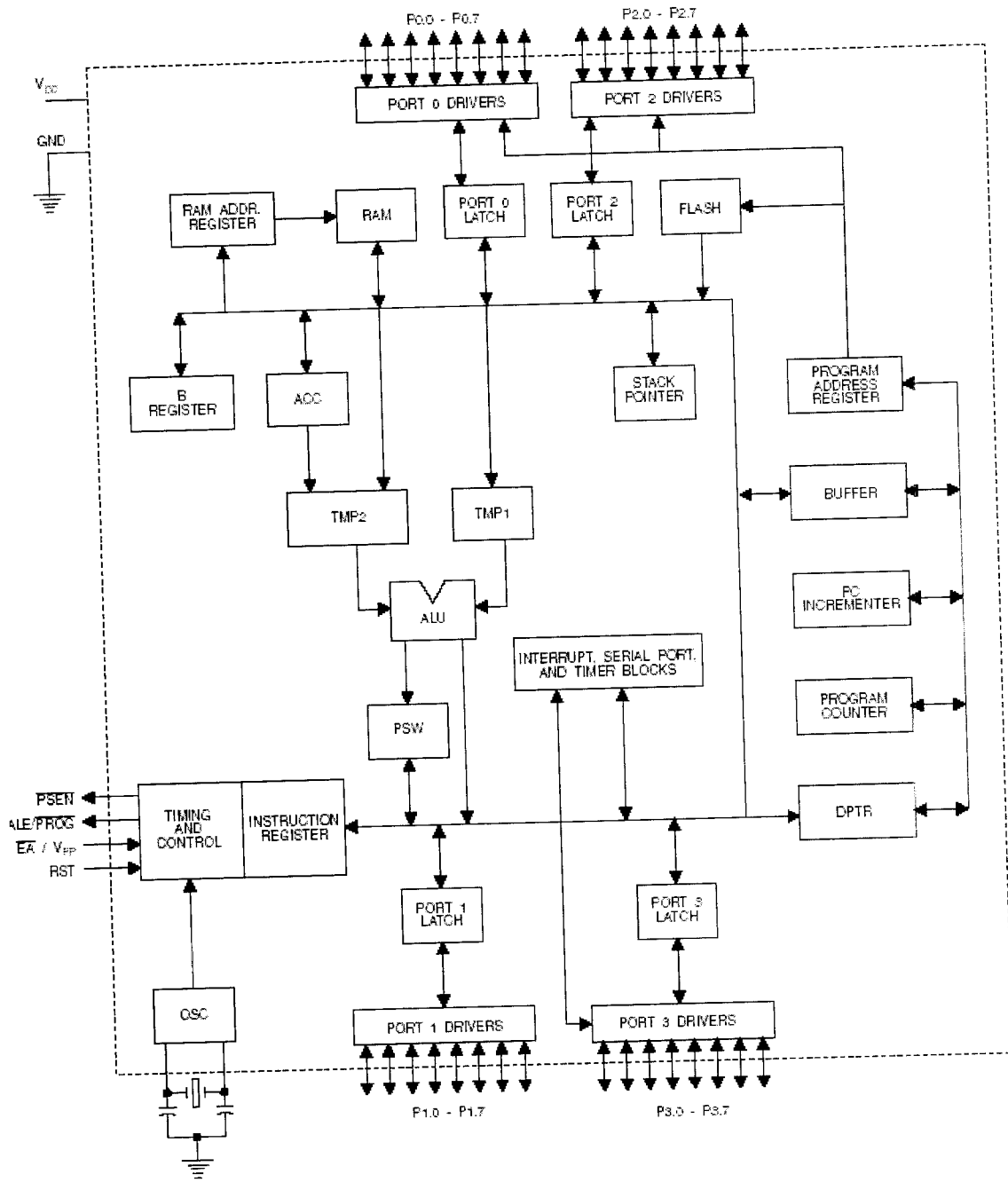
8-bit  
Microcontroller  
with 4K Bytes  
Flash

AT89C51





# Block Diagram



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.

## Pin Description

### 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 pullups.

Port 0 also receives the code bytes during Flash programming, and outputs the code bytes during program verification. External pullups 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 sink/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 ( $I_{IL}$ ) 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 sink/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 ( $I_{IL}$ ) 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 use 16-bit addresses (MOVX @ DPTR). In this application, it uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ R1), 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 sink/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 ( $I_{IL}$ ) 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	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (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/ $\overline{PROG}$

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 ( $\overline{PROG}$ ) 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 V<sub>CC</sub> for internal program executions.

This pin also receives the 12-volt programming enable voltage (V<sub>pp</sub>) during Flash programming, for parts that require 12-volt V<sub>pp</sub>.

### 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

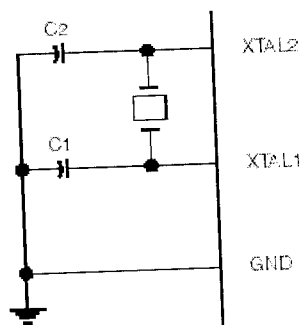
unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

## Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip 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 hardware 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.

Figure 1. Oscillator Connections

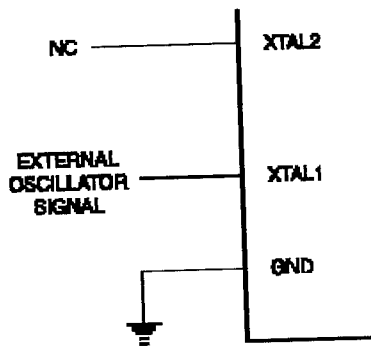


Note: C1, C2 = 30 pF ± 10 pF for Crystals  
= 40 pF ± 10 pF for Ceramic Resonators

## Status of External Pins During Idle and Power-down Modes

Mode	Program Memory	ALE	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 2. External Clock Drive Configuration



**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 Regis-

ters 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  $V_{CC}$  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 (P) to obtain the additional features listed in the table below:

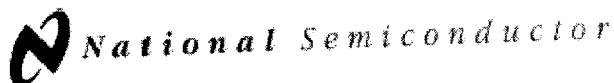
When lock bit 1 is programmed, the logic level at the  $\overline{EA}$  pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of  $\overline{EA}$  be in agreement with the current logic level at that pin in order for the device to function properly.

**Lock Bit Protection Modes**

	Program Lock Bits			Protection Type
	LB1	LB2	LB3	
1	U	U	U	No program lock features
2	P	U	U	MOV <sub>C</sub> 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

# LM7805 - REGULATOR

May 2000



## LM78XX Series Voltage Regulators

### General Description

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

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

Considerable effort was expended to make the LM78XX series of regulators easy to use and minimize the number of external components. It is not necessary to bypass the out-

put, although this does improve transient response. Input bypassing is needed only if the regulator is located far from the filter capacitor of the power supply.

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

### Features

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

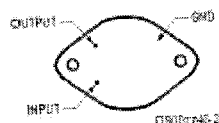
### Voltage Range

LM7805C	5V
LM7812C	12V
LM7815C	15V

LM78XX Series Voltage Regulators

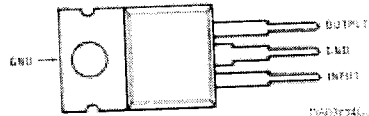
### Connection Diagrams

Metal Can Package  
TO-3 (K)  
Aluminum



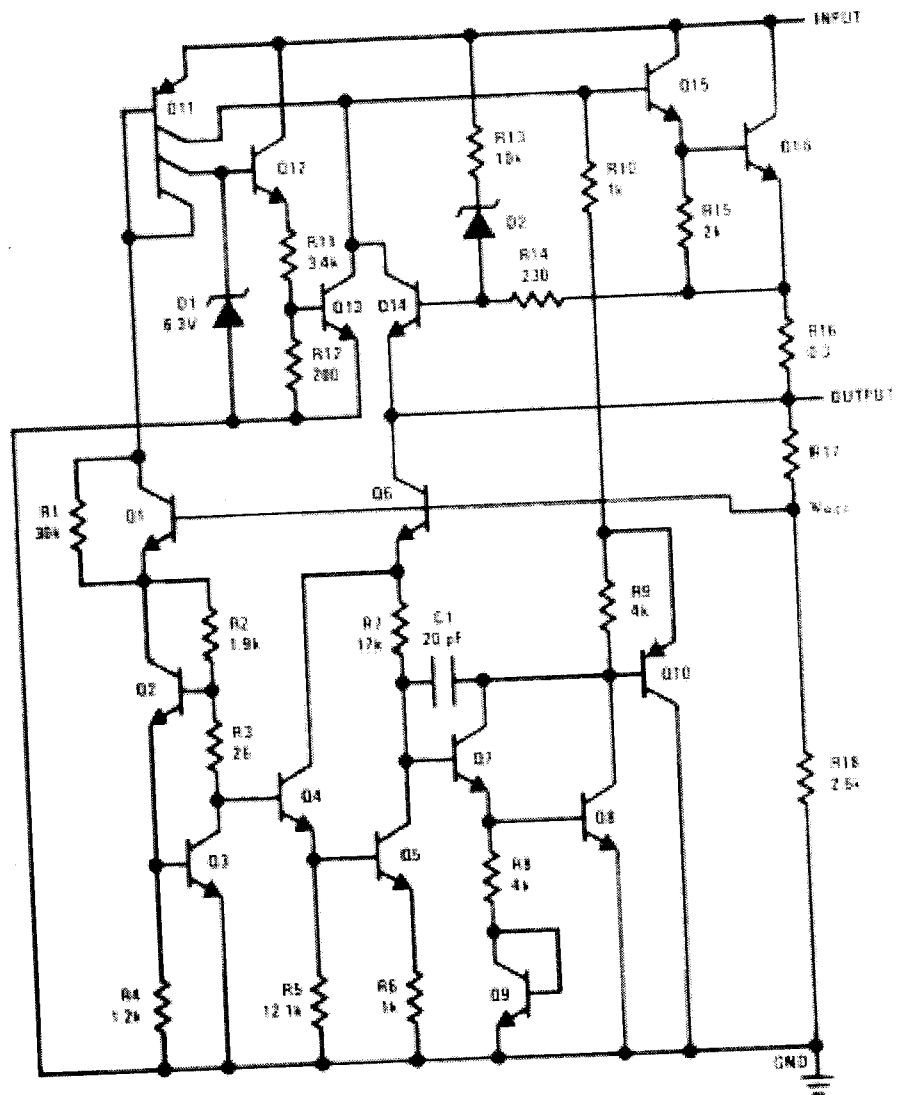
Bottom View  
Order Number LM7805CK,  
LM7812CK or LM7815CK  
See NS Package Number KC02A

Plastic Package  
TO-220 (T)



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

# Schematic



DS33740-1

## ULN2003 – RELAY DRIVER



**ULN2001A-ULN2002A**  
**ULN2003A-ULN2004A**

### SEVEN DARLINGTON ARRAYS

- SEVEN DARLINGTONS PER PACKAGE
- OUTPUT CURRENT 500mA PER DRIVER (600mA PEAK)
- OUTPUT VOLTAGE 50V
- INTEGRATED SUPPRESSION DIODES FOR INDUCTIVE LOADS
- OUTPUTS CAN BE PARALLELED FOR HIGHER CURRENT
- TTL/CMOS/PMOS/DTL COMPATIBLE INPUTS
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY LAYOUT

#### DESCRIPTION

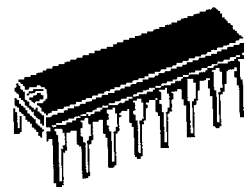
The ULN2001A, ULN2002A, ULN2003 and ULN2004A are high voltage, high current darlington arrays each containing seven open collector darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

The four versions interface to all common logic families :

ULN2001A	General Purpose, DTL, TTL, PMOS, CMOS
ULN2002A	14-25V PMOS
ULN2003A	5V TTL, CMOS
ULN2004A	6-15V CMOS, PMOS

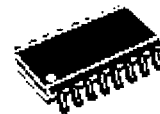
These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal print-heads and high power buffers.

The ULN2001A/2002A/2003A and 2004A are supplied in 16 pin plastic DIP packages with a copper leadframe to reduce thermal resistance. They are available also in small outline package (SO-16) as ULN2001D/2002D/2003D/2004D.



DIP 16

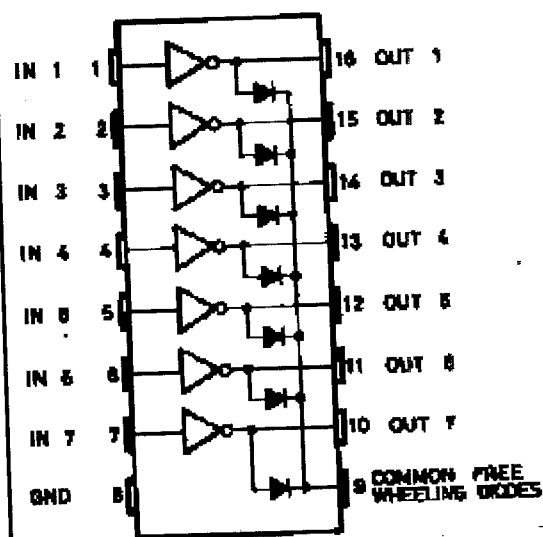
ORDERING NUMBERS: ULN2001A/2A/3A/4A



SO16

ORDERING NUMBERS: ULN2001D/2D/3D/4D

#### PIN CONNECTION



# ULN2803 – MOTOR DRIVER

## ULN2803A DARLINGTON TRANSISTOR ARRAY

SLRS049 – FEBRUARY 1997

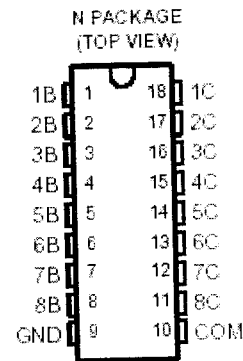
- 500 mA Rated Collector Current (Single Output)
- High-Voltage Outputs . . . 50 V
- Output Clamp Diodes
- Inputs Compatible With Various Types of Logic
- Relay Driver Applications
- Compatible with ULN2800A Series

### description

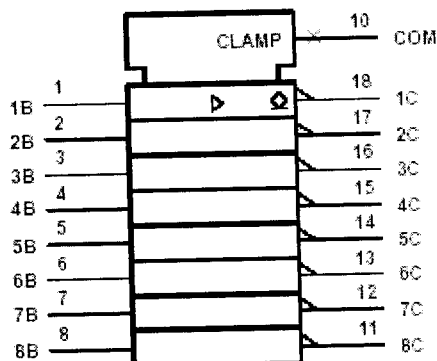
The ULN2803A is a monolithic high-voltage, high-current Darlington transistor array. The device consists of eight npn Darlington pairs that feature high-voltage outputs with common-cathode clamp diodes for switching inductive loads. The collector-current rating of each Darlington pair is 500 mA. The Darlington pairs may be paralleled for higher current capability.

Applications include relay drivers, hammer drivers, lamp drivers, display drivers (LED and gas discharge), line drivers, and logic buffers. The ULN2803A has a 2.7-k $\Omega$  series base resistor for each Darlington pair for operation directly with TTL or 5-V CMOS devices.

The ULN2803A is offered in a standard 18-pin dual in-line (N) package. The device is characterized for operation over the temperature range of -20°C to 85°C.

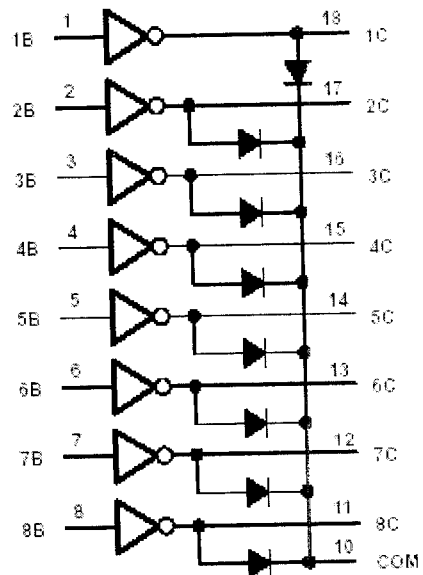


### logic symbol†



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

### logic diagram (positive logic)

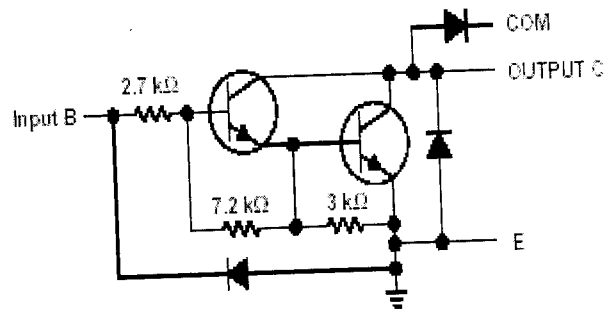




# ULN2803A DARLINGTON TRANSISTOR ARRAY

SLRS049 - FEBRUARY 1997

schematic (each Darlington pair)



# UM91215 - ENCODER

UNICORN MICROELECTRONICS 24E D ■ 9278788 0001275 T ■



# UMC

T-75-07-07

## UM91214/15 Series

### Tone/Pulse Dialer

#### Features

- One touch redial operation
- Tone/Pulse switchable
- 32-digit capacity for redialing
- Automatic mixed redialing (last number redial) of pulse to DTMF with multiple automatic access pauses.
- PABX auto-pause is 2.2 seconds
- DTMF Timing:
  - Manual dialing: minimum duration for bursts and pauses
  - Redialing: calibrated timing
- Hands-free control function

- Wide operating voltage range: 2V to 5.5V
- Key-in beep tone output
- Digits dialed manually after redialing are cascadable and stored as additional digits for the next redialing
- Uses inexpensive ceramic resonator (3.58 MHz)
- Two versions for different telephone systems
- Built-in power up reset circuit
- Four extra function keys: flash, pause, redial and DP or DTMF mixed dialing
- Four-by-four (or 2 of 8) keyboard can be used
- Low standby current

Tone/Pulse Dialer

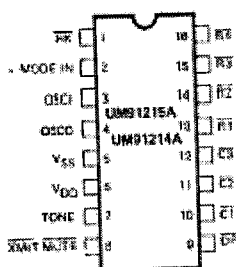
#### General Description

The UM91214/15 is a single-chip, silicon gate, CMOS integrated circuit with an on-chip oscillator for a 3.58 MHz crystal or ceramic resonator. It provides dialing pulse (DP) or dual tone multi-frequency (DTMF) dialing. A standard 4 x 4 matrix keyboard can be used to support either

DP or DTMF modes. Up to 32 digits can be saved in the on-chip RAM for redialing. In the DTMF mode, minimum tone duration and minimum intertone pause provide for rapid dialing. Maximum tone duration is dependent upon the key depression time in manual dialing.

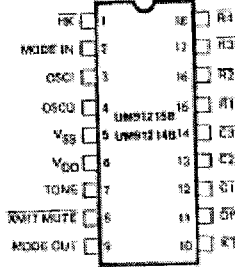
#### Pin Configurations

a. 16 Pin Package



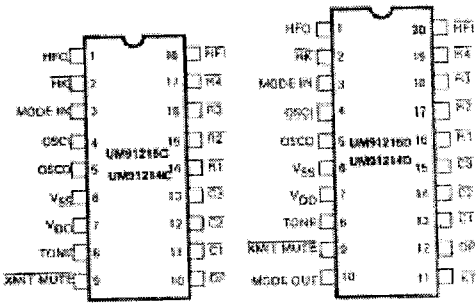
b. 18 Pin Packages

(i) Key tone output



c. 20 Pin Package

(ii) Hands free control





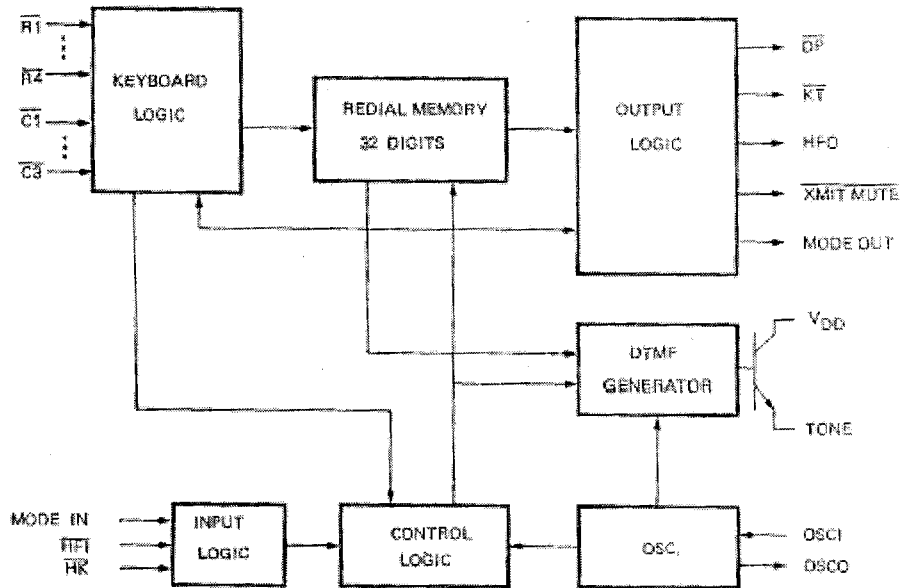
T-75-07-07

Keyboard Assignment

1	2	3	F1	R1
4	6	€	F2	R2
7	8	9	P	R3
*T	0	#	RD	R4
C1	C2	C3	GND	

1. \*T -- At Pulse mode this key works as Pulse → DTMF key (T key), at DTMF mode the key works as \*key. \*T key will occupy one memory digit in either use.
2. F1 -- Flash key. The break time is 297 ms or 96 ms (UM91214/15 respectively)
3. F2 -- Flash key for break time 640 ms
4. P -- Pause key (2.2 seconds)
5. RD -- One key radial key
6. # -- At pulse mode this key input is neglected, at DTMF mode this key works as # key.

Block Diagram

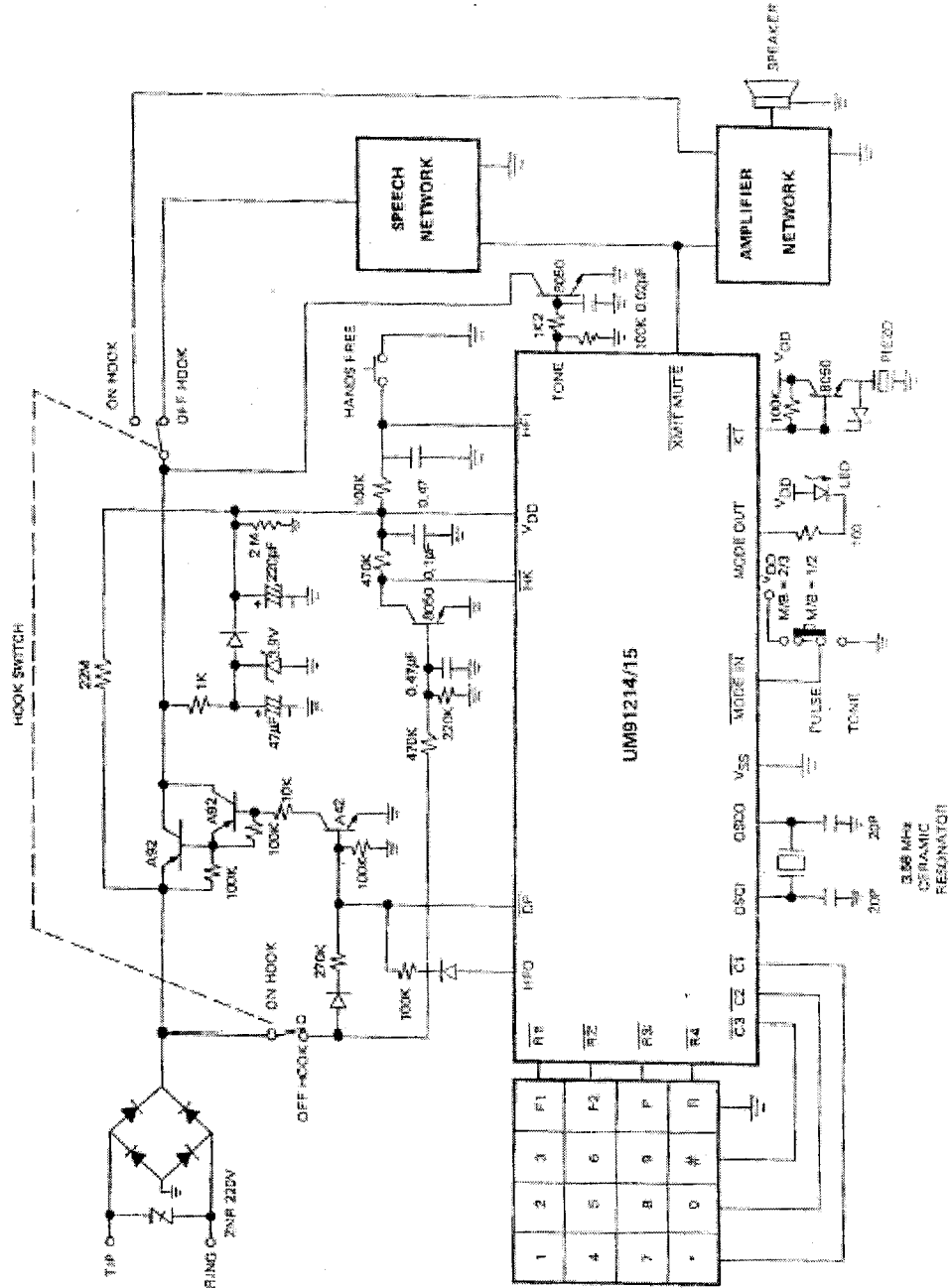




UM91214/15 Series

Application Circuit

T-75-07-07



# CM8870 - DECODER



CALIFORNIA MICRO DEVICES



CM8870/70C

## CMOS Integrated DTMF Receiver

### Features

- Full DTMF receiver
- Less than 35mW power consumption
- Industrial temperature range
- Uses quartz crystal or ceramic resonators
- Adjustable acquisition and release times
- 18-pin DIP, 18-pin DIP EIAJ, 18-pin SOIC, 20-pin PLCC
- CM8870C
  - Power down mode
  - Inhibit mode
  - Buffered OSC3 output (PLCC package only)
- CM8870C is fully compatible with CM8870 for 18-pin devices by grounding pins 5 and 6

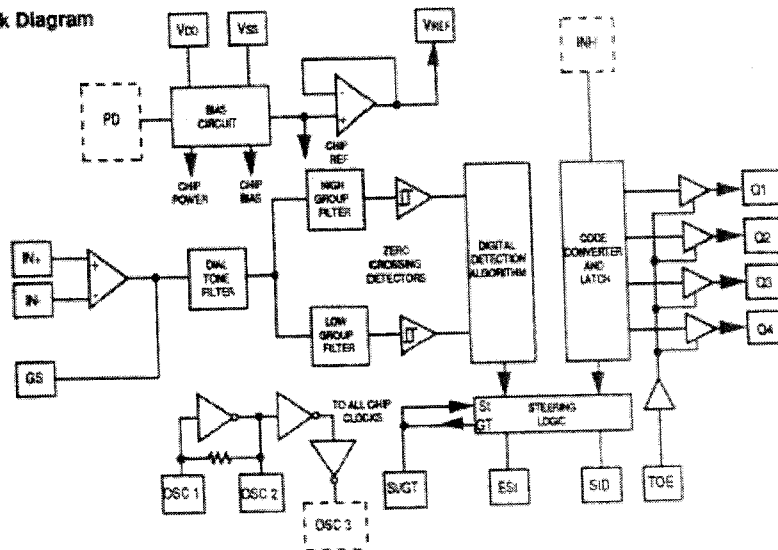
### Applications

- PABX
- Central office
- Mobile radio
- Remote control
- Remote data entry
- Call limiting
- Telephone answering systems
- Paging systems

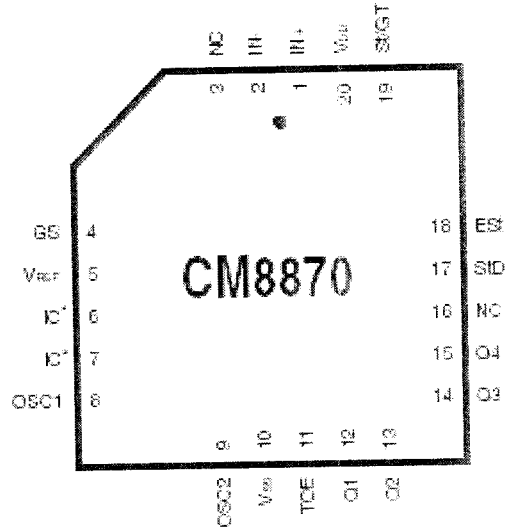
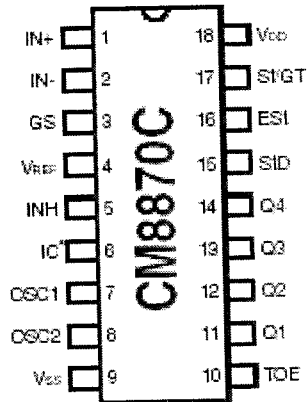
### Product Description

The CAMD CM8870/70C provides full DTMF receiver capability by integrating both the bandsplit filter and digital decoder functions into a single 18-pin DIP, SOIC, or 20-pin PLCC package. The CM8870/70C is manufactured using state-of-the-art CMOS process technology for low power consumption (35mW, max.) and precise data handling. The filter section uses a switched capacitor technique for both high and low group filters and dial tone rejection. The CM8870/70C decoder uses digital counting techniques for the detection and decoding of all 16 DTMF tone pairs into a 4-bit code. This DTMF receiver minimizes external component count by providing an on-chip differential input amplifier, clock generator, and a latched three-state interface bus. The on-chip clock generator requires only a low cost 1V crystal or ceramic resonator as an external component.

Block Diagram

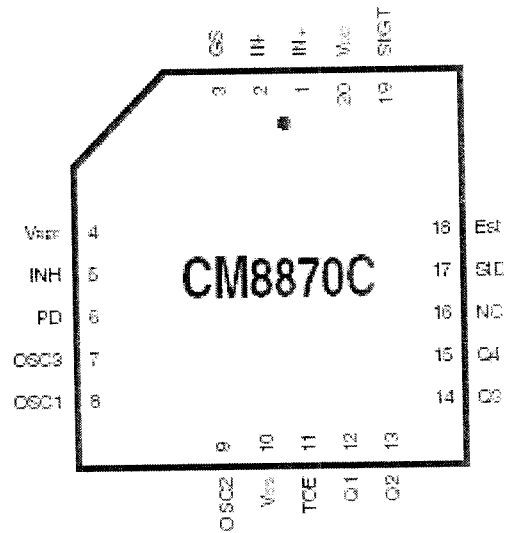
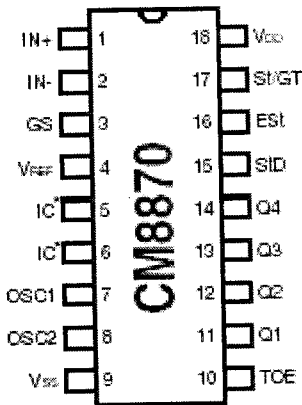


## Pin Assignments



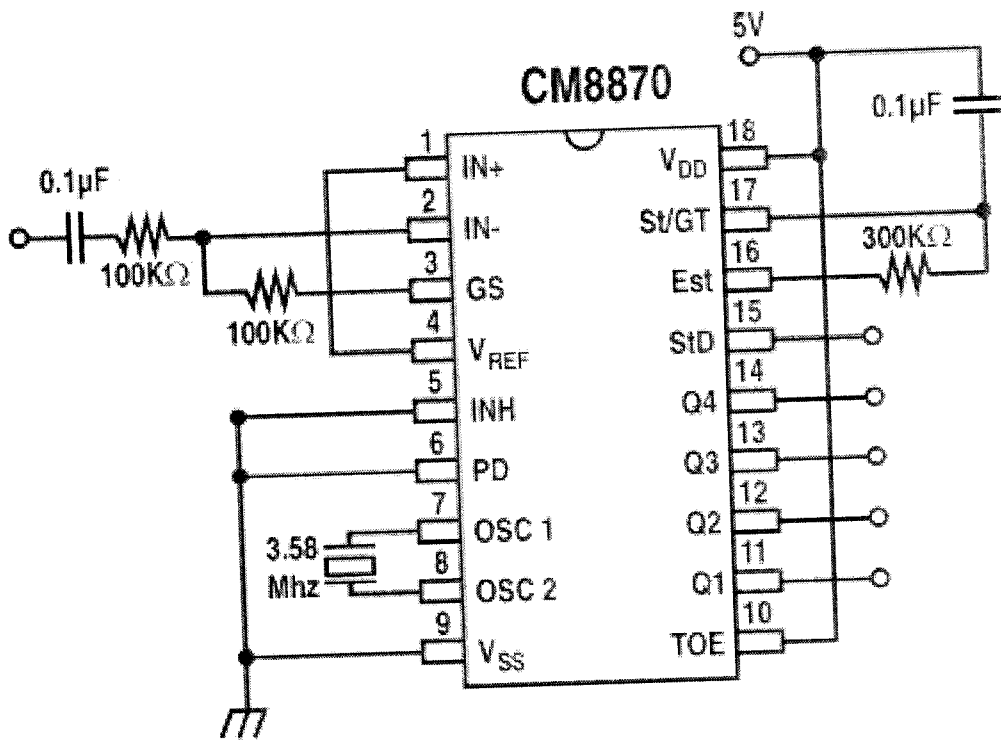
- P — Plastic DIP (18)
- F — Plastic SOP  
EIAJ (18)
- S — SOIC (18)

- PE — PLCC (20)
- \* — Connect To VSS



- P — Plastic DIP (18)
- F — Plastic SOP  
EIAJ (18)
- S — SOIC (18)

- PE — PLCC (20)



## 7. REFERENCES

1. Kenneth J. Ayala (1996) "8051 Microcontroller, Architecture, Programming and Application" Delmar Publication, Inc. II Edition
2. [www.globalsources.com](http://www.globalsources.com)
3. [www.Planet-Source-Code.com](http://www.Planet-Source-Code.com)
4. [www.atmel.com](http://www.atmel.com)
5. [www.calmicro.com](http://www.calmicro.com)
6. [www.national.com](http://www.national.com)
7. [www.st.com](http://www.st.com)
8. [www.ajoka.com](http://www.ajoka.com)
9. [www.ams2000.com](http://www.ams2000.com)
10. [www.cctvstuff.co.uk](http://www.cctvstuff.co.uk)