p-1996

L.A.S.E.R
Learning Abilities by a Speech Enabled Robot

A PROJECT REPORT

Submitted by

Arjun Raj	71203106003
M.Arun Karthick	71203106004
S.Arunachala Kani	71203106007
V.Jagadish Kumar	71203106016

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY COMBATORE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2007

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"LEARNING ABILITIES BY A SPEECH ENABLED ROBOT"** is the bonafide work of **"ARJUN RAJ, M.ARUN KARTHICK, S. ARUNACHALA KANI AND V.JAGADISH KUMAR"** who carried out the project work under my supervision.

[illegible] 20/4/07

SIGNATURE

Dr.Rajeswari Mariappan

HEAD OF THE DEPARTMENT

Department of ECE

Kumaraguru College of Technology

Coimbatore – 641 006

[illegible]

SIGNATURE 19/4/07

Prof.K.Ramprakash

SUPERVISOR

PROFESSOR

Department of ECE

Kumaraguru College of Technology

Coimbatore – 641 006

The candidates with university register number 71203106003, 71203106004, 71203106007, 71203106016 were examined by us in project viva voce examination held on 23/04/2007

[illegible]

Internal Examiner

[illegible]

External Examiner 23

ABSTRACT

ABSTRACT

Image processing, Audio signal processing and servo motor control are the critical elements that form the foundation of our project. Using the Image acquisition toolbox available as a part of MATLAB 7, images of an object are converted into digital data. The image processing toolbox then operates on this data to generate suitable results. The audio processing toolbox and servo motor controls are utilized to evoke a response from the system.

An object is placed on a table whose image is acquired as mentioned above and then the moment invariants of that object are found. Each object has its own moment invariants quite unique from all other objects. The moments are then used to identify the object. Identification takes place in the demonstration phase where as moment invariants are determined in the learning phase.

Once an object is identified the system then reacts via an audio output and a robotic arm points to the object on the table. The robotic arm is driven by servo motors controlled by the PC parallel port. After identification the arm retrieves to its original position.

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We deem it a great pleasure to place on record our deep sense of gratitude and indebt ness to **Dr. Joseph Thanikkal**, Principal, Kumaraguru College of Technology, for providing us this excellent opportunity to work on this project.

We express our sincere gratitude to **Dr Rajeswari Mariappan**, Head of the department of Electronics and Communication, for the immense concern and encouragement shown during the course of the project.

We profusely thank our project coordinator and guide, **Prof. K. Ramprakash**, Professor, for his moral support and motivation given to us regarding our project work and for guiding us throughout by his constant support, valuable comments, suggestions, and for the inspiration and guidance given to us in molding our project.

Our special thanks to our class advisor **Miss. R. Hemalatha and Prof. S. Govinda Raju** for their extraordinary help lended in completing our project.

The sense of fulfillment and triumph in completing this project would remain but incomplete without us thanking our respected teaching and non teaching staffs, our beloved parents, our intimate friends and family members for their unlimited support in crossing an important milestone in our life.

TABLE OF CONTENT

LIST OF TABLE

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES

INTRODUCTION

1. INTRODUCTION

To achieve object identification we use MATLAB. In particular the tool boxes used are image acquisition, image processing and audio support. The PC parallel port is used to control the robotic arm through the MATLAB interface.

This report discusses the different approaches we have taken to solve problems relevant to the project. In addition to successful techniques, we also present the various ideas that simply did not work.

Once this technology is implemented on a large scale basis, it will prove to be a boon to many industrial processes requiring object identification. The results and techniques of this project can be employed in the development of robots exhibiting learning abilities.

BASIC ASSEMBLY OF THE SYSTEM

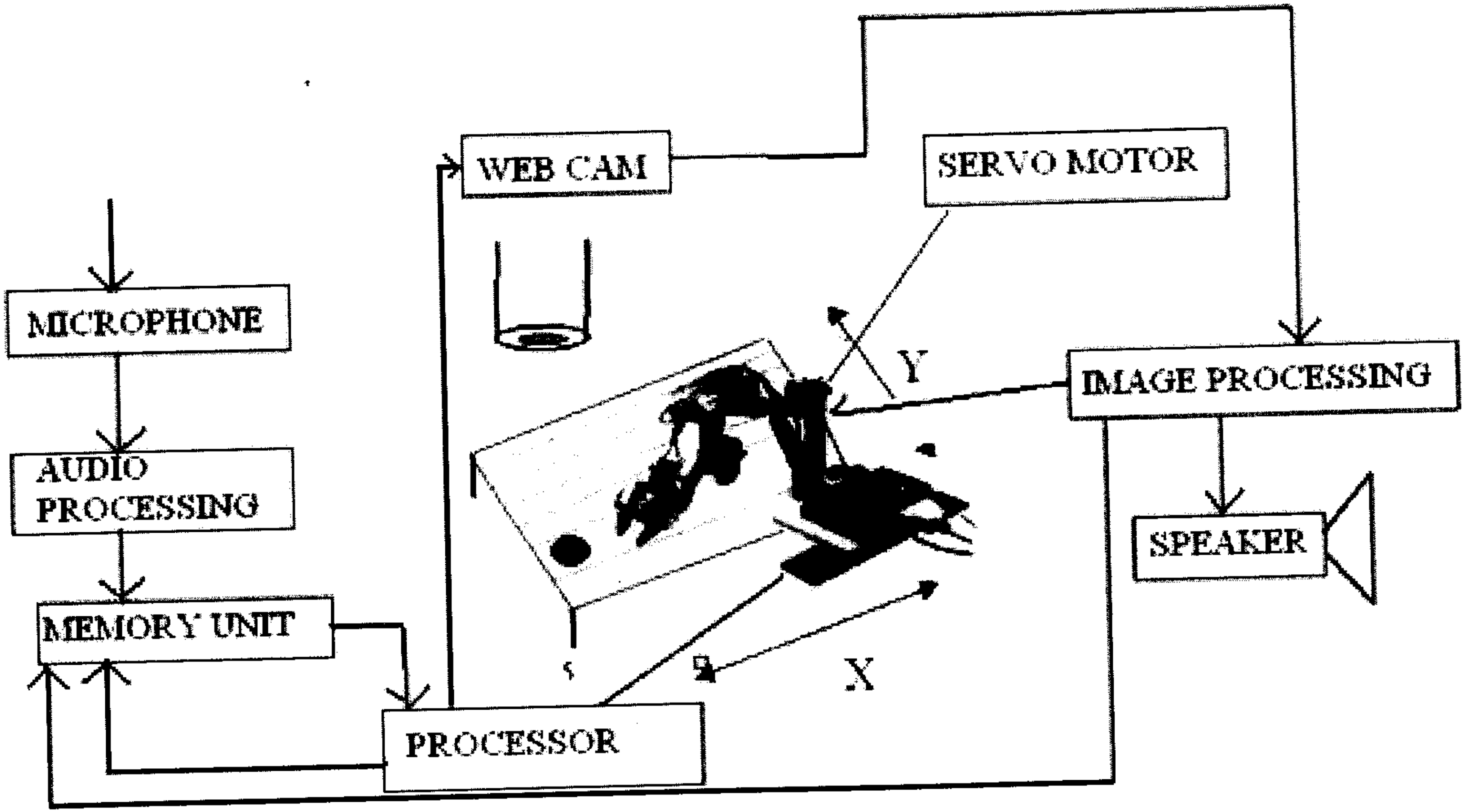


Fig1: Basic assembly of the system

The figure below is the complete black diagram of our project. It consists of a robotic arm which can move in the x and y direction. The web camera is used to convert the optical signals into electrical signals which are then converted into digital data by MATLAB. A microphone is used to bring audio input into the MATLAB environment. For audio output a loudspeaker is used.

The various components used are explained one at a time in this document along with the code required to interface them to the system

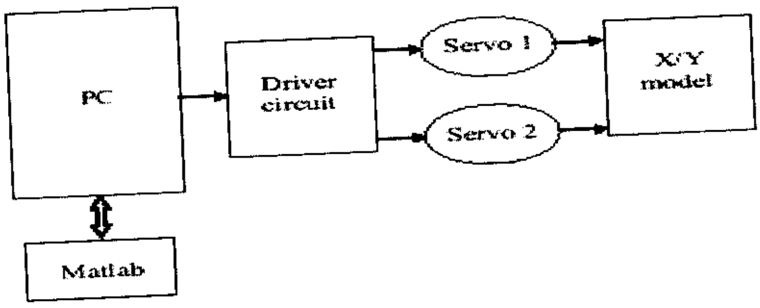


Fig 2: block diagram of the robot

DESCRIPTORS

2. DESCRIPTORS

2.1 FOURIER DESCRIPTORS

Figure shows a K-point digital boundary in the xy-plane. Starting at an arbitrary point (x_o,y_o), coordinate pairs (x_o,y_o), (x_1,y_1), (x_2,y_2),..., (x_{k-1},y_{k-1}) are encountered in traversing the boundary, say, in the counterclockwise direction. These coordinates can be expressed in the form $x(k) = x_k$ and $y(k) = y_k$. With this notation, the boundary itself can be represented as the sequence of coordinates $s(k) = [x(k), y(k)]$, for $k = 0,1.2....,K - 1$. Moreover, each coordinate pair can be treated as a complex number so that

$$s(k) = x(k) + jy(k)$$

From Section 4.1, the discrete Fourier transform (DFT) of $s(k)$ is

$$a(u) = \sum_{k=0}^{K-1} s(k) e^{-j2\pi uk/K}$$

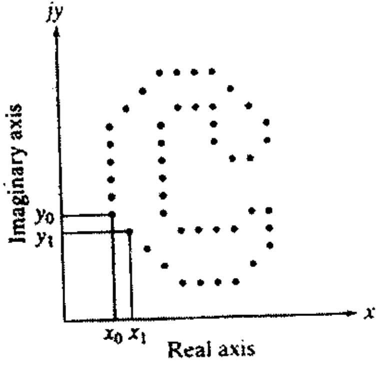
for $u = 0,1,2,..., K — 1$. The complex coefficients $a(u)$ are called the *Fourier descriptors* of the boundary. The inverse Fourier transform of these coefficients restores $s(k)$. That is,

$$s(k) = \frac{1}{K} \sum_{u=0}^{K-1} a(u) e^{-j2\pi uk/K}$$

for $k = 0,1,2,..., K - 1$. Suppose, however, that instead of all the Fourier coefficients, only the first P coefficients are used. This is equivalent to setting $a(u) = 0$ for $u > P - 1$ in the preceding equation for $a(u)$. The result is the following *approximation* to $s(k)$:

$$\hat{s}(k) = \frac{1}{P}\sum_{u=0}^{P-1} a(u)e^{-j2\pi uk/K}$$

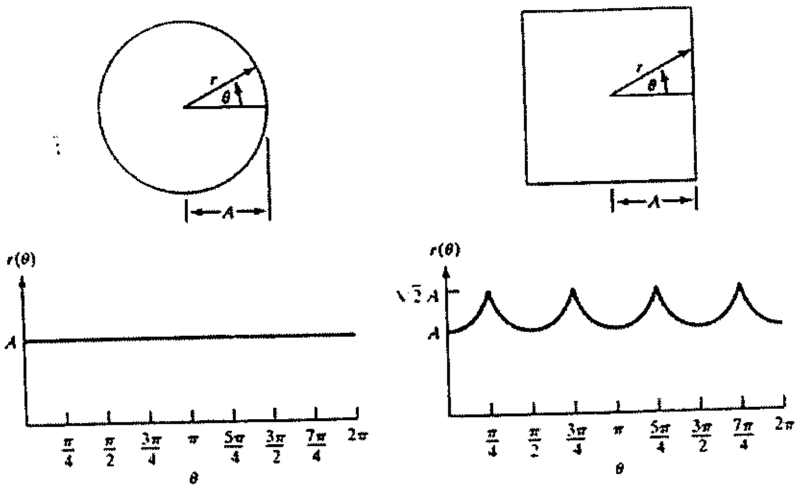
for k = 0.1,2,..., K - 1. Although only P terms are used to obtain each component of *s(k), k* still ranges from 0 to *K - 1*. That is, *the same* number of points exists in the approximate boundary, but not as many terms are used in the reconstruction of each point. Recall from Chapter 4 that high-frequency components account for fine .detail, and low-frequency components determine global shape. Thus, loss of detail in the boundary increases as *P* decreases.



The following function, `frdescp`, computes the Fourier descriptors of a boundary, `s`. Similarly, given a set of Fourier descriptors, function `ifrdescp` computes the inverse using a specified number of descriptor, to yield a closed spatial curve. The documentation section of each function explains its syntax.

2.2 SIGNATURES

A *signature* is a 1-D functional representation of a boundary and may be generated in various ways. One of the simplest is to plot the distance from an interior point (e.g.. the centroid) to the boundary as a function of angle, as illustrated in Figure. Regardless of how a signature is generated, however, the basic idea is to reduce the boundary representation to a 1-D function, which presumably is easier to describe than the original 2-D boundary. Keep in mind that it makes sense to consider using signatures only when it can be guaranteed that the vector extending from its origin to the boundary intersects the boundary only once, thus yielding a single-valued function of increasing tingle. This excludes boundaries with self-intersections, and it also typically excludes boundaries with deep, narrow concavities or thin, long protrusions.



Signatures generated by the approach just described are invariant to transla-linn. but they do depend on rotation and scaling. Normalization with respect to rotation can be achieved by finding a way to select the point to generate the signature, regardless of the shape's orientation. One way to do so is to select the starting point as the point farthest from the origin of the vector, if this point happens to be unique and independent of rotational aberrations for each shape of interest.

Another way is to select a point on the major eigen axis. This method requires more computation but is more rugged because the direction of the eigen axes is determined by using all contour points. Yet another way is to obtain the chain code of the boundary and then use the approach discussed, assuming that the rotation can be approximated by the discrete angles in the code directions defined in Figure.

Based on the assumptions of uniformity in scaling with respect to both axes, and that sampling is taken at equal intervals of 9, changes in size of a shape result in changes in the amplitude values of the corresponding signature. One way to normalize for this dependence is to scale all functions so that they always span the same range of values, say, [0,1]. The main advantage of this method is simplicity, but it has the potentially serious disadvantage that scaling of the entire function is based on only two values: the minimum and maximum. If the shapes are noisy, this can be a source of error from object to object. A more rugged approach is to divide each sample by the variance of the signature, assuming that the variance is not zero—as in the case of Figure— or so small that it creates computational difficulties. Use of the variance yields a variable scaling factor that is inversely proportional to changes in size and works much as automatic gain control does. Whatever the method used, keep in mind that the basic idea is to remove dependency on size while preserving the fundamental shape of the waveforms.

Function signature, included in Appendix C, finds the signature of a given boundary. Its syntax is

```
[st, angle, xO, yO] = signaturefb, xO, yO)
```

where b is an *np* X 2 array containing the xv-coordinatcs of a boundary ordered in a clockwise or counterclockwise direction. The amplitude of the signature as a function of increasing angle is output in st. Coordinates xO, yO) in the input are the coordinates of the origin of the vector extending to the boundary. If these coordinates are not included in the argument, the function uses the coordinates of the centroid of the boundary by default. In either case, the values of (xO, yO) used by the function are included in the output.The size of arrays st and angle is 360 X 1, indicating a resolution of one degree. The input must be a one-pixel-thick boundary obtained, for example, using function boundaries. As before, we assume that a boundary is a closed curve.

Function signature utilizes MATLAB's function cart2pol to convert Cartesian to polar coordinates. The syntax is

```
[THETA,, RHO]    = cart2p01(X,    Y)
```

where X and Y are vectors containing the coordinates of the Cartesian points. The vectors THETA and RHO contain the corresponding angle and length of the polar coordinates. If X and Y are row vectors, so are THETA and RHO, and similarly in the case of columns. The convention used by MATLAB for coordinate conversions.Note that the MATLAB coordinates (X, Y) in this situation are related to our image coordinates (x, y) as X = y and Y = -x [see Fig. 2.1 (a)]. Function pol2cart is used for converting back to Cartesian coordinates:

```
        [X,   Y]   =  pol2cart(THETA,   RHO)
```

Figures show the boundaries, bs and bt, of an irregular square and triangle, respectively, embedded in arrays of size 674 X 674 pixels. Figure shows the signature of the square, obtained using the commands

```
» [st, angle, xO, yO] = signature(bs);
» plot(angle,   st)
```

The values of xO and yO obtained in the preceding command were [342,326]. A similar pair of commands yielded the plot in Figure whose centroid islocated at [416, 335]. Note that simply counting the number of prominent peaks in tketwo signatures is sufficient to differentiate between the two boundaries.

2.3 MOMENT INVARIANTS

The 2-D moment of order (p+q) of a digital image f (x,y) is defined as

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y)$$

for p,q = 0,1,2,....., where the summations are over the values of the spatial coordinates x and y spanning the image. The corresponding central moment is defined as

$$\mu_{pq} = \sum_x \sum_y (x-\bar{y})^p (y-\bar{y})^q f(x,y)$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}} \text{ and } \bar{y} = \frac{m_{01}}{m_{00}}$$

The normalized central moment of order (p + q) is defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{y}}$$

for p, q = 0, 1, 2, 3, ...

A set of seven 2-D moment invariants that are insensitive to translation, scale change mirroring, and rotation can be derived from these equations. They are

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\phi_4 = (\eta_{30} - \eta_{12})^2 + (\eta_{21} - \eta_{03})^2$$

$$\begin{aligned}\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\big[(\eta_{30} + \eta_{12})^2 \\ - 3(\eta_{21} + \eta_{03})^2\big] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \\ \big[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\big]\end{aligned}$$

$$\begin{aligned}\phi_6 = (\eta_{20} - \eta_{02})\big[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\big] \\ + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\end{aligned}$$

$$\begin{aligned}\phi_6 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\big[(\eta_{30} + \eta_{12})^2 \\ - 3(\eta_{21} + \eta_{03})^2\big] + (3\eta_{21} - \eta_{30})(\eta_{21} + \eta_{03}) \\ \big[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\big]\end{aligned}$$

An M-function for computing the moment invariants, which we call invmoments,is a direct implementation of these seven equations.The syntax is as follows (see Appendix C for the code listing):

```
phi = invmoments(f)
```

where `f` is the input image and `phi` is a seven-element row vector containing the moment invariants just defined.

The image in Fig. (a) was obtained from an original of size 400 X 400 pixels by using the command

```
» fp = padarray(f,   [84 84],   'both');
```

Zero padding was used to make all displayed images consistent with the image occupying the largest area (568 X 568 pixels) which, as discussed below, is the image rojated by 45°. The padding is for display purposes only, and was not used in any moment computations. The half-size and corresponding padded images were obtained using the commands

```
» fhs = f(1:2:end,   1:2:end);
» fhsp = padarrayffhs,   [184 184],   'both');
```

The mirrored image was obtained using MATLAB function `fliplr`:

```
>> fm = fjiplr(f);
>> fmp = padarray(fm,   [84 84],   'both');
```

To rotate an image we use function imrotate:

```
g = imrotate(f,  angle,  method,   'crop')
```

which rotates `f` by angle degrees in the counterclockwise direction. Parameter method can be one of the following:

`'nearest'` uses nearest neighbor interpolation;

`'bilinear'` uses bilinear interpolation (typically a good choice); and

`'bicubic'` uses bicubic interpolation.

The image size is increased automatically by padding to fit the rotation. If `'crop'` is included in the argument, the central part of the rotated image is cropped to the same size as the original. The default is to specify angle only, in which case ' nearest' interpolation is used and no cropping takes place.

Invariant (|log|)	**Original**	**Half Size**	**Mirrored**	**Rotated 2°**	**Rotated 45°**
ϕ_1	6.600	6.600	6.600	6.600	6.600
ϕ_2	16.410	16.408	16.410	16.410	16.410
ϕ_3	23.972	23.958	23.972	23.978	23.973
ϕ_4	23.888	23.882	23.888	23.888	23.888
ϕ_5	49.200	49.258	49.200	49.200	49.198
ϕ_6	32.102	32.094	32.102	32.102	32.102
ϕ_7	47.953	47.933	47.850	47.953	47.954

The rotated images for our example were generated as follows:

```
» fr2 = imrotate(f,  2,   'bilinear');
» fr2p = padarray(fr2,   [76 76],   'both');
» fr45 = imrotateff,  45,   'bilinear');
```

Note that no padding was required in the last image because it is the largest image in the set. The 0s in both rotated images were generated by IPT in the process of rotation.

The seven moment invariants of the five images just discussed were generated using the commands

```
» phiorig = abs(log(invmoments(f)));
» phihalf = abs(log(invmoments(fhs)));
» phimirror = abs(log(invmoments(fm)));
» phirot2 = abs(log{invmoments(fr2)));
» phirot45 = abs(log{invmoments(fr45)));
```

Note that the absolute value of the log was used instead of the moment invariant values themselves. Use of the log reduces dynamic range, and the absolute value avoids having to deal with the complex numbers that result when computing the log of negative moment invariants. Because interest generally lies on the invariance of the moments, and not on their sign, use of the absolute value is common practice.

The seven moments of the original, half-size, mirrored, and rotated images are summarized in Table 11.4. Note how close the numbers are, indicating a high degree of invariance to the changes just mentioned. Results like these are the reason why moment invariants have been a basic staple in image description for for more than four decades.

INTERFACE OF MATLAB & ROBOTIC ARM

3. INTERFACING OF MATLAB AND ROBATIC ARM

3.1 INTRODUCTION TO PARALLEL PORTS

The Parallel Port is the most commonly used port for interfacing home made projects. This port will allow the input of up to 9 bits or the output of 12 bits at any one given time, thus requiring minimal external circuitry to implement many simple tasks. The port is composed of 4 control lines, 5 status lines and 8 data lines. It's found commonly on the back of your **PC** as a **D**-Type 25 Pin female connector. There may also be a D-Type 25 pin male connector. This will be a serial **RS-232** port and thus, is a totally incompatible port.

Newer Parallel Port's are standardized under the **IEEE 1284** standard first released in 1994. This standard defines 5 modes of operation which are as follows,

1. Compatibility Mode.
2. Nibble Mode. *(Protocol not Described in this Document)*
3. Byte Mode. *(Protocol not Described in this Document)*
4. EPP Mode *(Enhanced Parallel Port)*
5. ECP Mode *(Extended Capabilities Port).*

The aim was to design new drivers and devices which were compatible with each other and also backwards compatible with the Standard Parallel Port **(SPP).** Compatibility, Nibble & Byte modes use just the standard hardware available on the original Parallel Port cards faster speeds, while still being downloads compatible with the Standard Parallel Port.

Compatibility mode or "Centronics Mode" as it is commonly known, can only send data in the forward direction at a typical speed of 50 kbytes per second but can be as high as 150+ kbytes a second. In order to receive data, you must change the mode to either Nibble or Byte mode. Nibble mode can input a nibble *(4 bits)* in the reverse direction. E.g. from device to computer. Byte mode uses the Parallel's bi-directional feature *(found only on some cards)* to input a byte (8 bits) of data in the reverse direction.

Extended and Enhanced Parallel Ports use additional hardware to generate and manage handshaking. To output a byte to a printer (or anything in that matter) using compatibility mode, the software must.

1. Write the byte to the Data Port.
2. Check to see is the printer is busy. If the printer is busy, it will not accept any data, thus any data which is written will be lost.
3. Take the Strobe (Pin 1) low. This tell the printer that there is the correct data on the data lines. (Pin 2-9)
4. Put the strobe high again after waiting approximately 5 microseconds after putting the strobe low. (Step 3)

This limits the speed at which the port can run at. The EPP & CEP ports get by letting the hardware check to see if the printer is busy and generate a strobe and / or appropriate handshaking. This means only one

I/O instruction need to be performed, thus increasing the speed. These ports can output at around 1-2 megabytes per second. The ECP port also has the advantage of using DMA channels and FIFO buffers, thus data can be shifted around without using I/O.

Hardware Properties

On the next page is a table of the "Pin Outs" of the D-Type 25 Pin connector and the Centronics 34 Pin connector. The D-Type 25 pin connector is the most common connector found on the Parallel Port of the computer, while the Centronics Connector is commonly found on printers. The IEEE 1284 Type standard however specifies 3 different connectors for use with the Parallel Port. The first one, 1284 Type A is the D-Type 25 connector found on the back of most computers. The 2nd is the 1284 Type B which is the 36 pin Centronics Connector found on most printers.

IEEE 1284 Type C however, is a 36 conductor connector like the Centronics, but smaller. This connector is claimed to have a better clip latch, better electrical properties and is easier to assemble. It also contains two more pins for signals which can be used to see whether the other device connected, has power. 1284 Type C connector are recommended for new designs, so we can look forward on seeing these new connectors is the near future.

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/Out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out PaperEnd	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer nSelect-In	In/Out	Control	Yes
18	19-30	Ground	Gnd		

Table 1: pin details of parallel port

The above table uses "n" in found of the signal name to denote that the signal is active low. E.g. Error. If the printer has occurred an error then this line is low. This line normally is high, should the printer be functioning correctly. The "Hardware Inverted" means the signal is inverted by the Parallel card's hardware. Such an example is the Busy line. If +5v (Logic 1) was applied to this pin and the status register read, it would return back a 0 in Bit 7 of the Status Register.

The output of the Parallel Port is normally TTL logic levels. The voltage levels are the easy part. The current you can sink and source around 12mA. However these are just some of the figures taken from Data sheets, Sink/Source 6mA, Source 12mA/Sink 20mA, Sink 16mA/Source 4mA, Sink/Source 12mA. As you can see they very quit a bit. The best bet is to use a buffer, so the least current is drawn from the Parallel Port.

3.2 PORT ADDRESSES

The Parallel Port has three commonly used base addresses. These are listed in table 2, below. The 3BCh base address was originally introduced used for Parallel Port on early Video Cards. This address then disappeared for a while, when Parallel Ports were later removed from Video Cards. They has now reappered as an option for Parallel Port integrated onto motherboards, upon which their configuration can be changed using BIOS.

LPTI is normally assigned base address 378h, while LPT2 is assigned 278h. However this may not always be the case as explained later. 378h & 278h have always been commonly used for Parallel Port. The lower case h denotes that it is in hexadecimal. These addresses may change from machine to machine.

Address	Notes:
3BCh – 3BFh	Used for Parallel Ports which were incorporated in to Video Cards and now, commonly an option for Ports controlled by BIOS. – Doesn't support ECP addresses.
378h – 37Fhz	Usual Address For LPT 1
278h – 27Fh	Usual Address For LPT 2

Table 2: port address

When the computer is first turned on, BIOS (Basic Input/Output System) will determine the number of ports you have and assign device labels LPT1, LPT2 & LPT3 to them BIOS first looks at address 3BCh. If a Parallel Port is found here, it is assigned as LPT1, them it searches at location 378h. If a Parallel card is found there, it is assigned the next free device label. This would be LPT1 if a card wasn't found at 3BCh or LPT2 if a card was found at 3BCh. The last port of call, is 278h and follows the same procedure than the other two ports. Therefore it is possible to have a LPT2 which is at 378h and not at the expected address 278h.

What can make this even confusing, is that some manufactures of Parallel Port Cards, have jumpers which allow you to set your Port to LPT1, LPT2 & LPT3. Now what address is LPT1? – On the majority of cards LPT1 is 378h, and LPT2, 278h, but some will use 3BCh as LPT1, 378h as LPt1 and 278h as LPT2. Life wasn't meant to be easy.

The assigned devices LPT1, LPT2 & LPT3 should not be a worry to people wishing to interface devices to their PC's. Most of the time the base address is used to interface the port rather than LPT1 etc. However should you want to find the address of LPT1 or any of the Line Printer Devices, you can use a lookup table provided by BIOS. When BIOS assigns addresses to your printer devices, it stores the address at specific locations in memory, so we can find them.

Start Address	**Function**
0000:0408	LPT1's Base Address
0000:040A	LPT2's Base Address
0000:040C	LPT3's Base Address
0000:040E	LPT4's Base Address (Note 1)

Table 3: base address

Note 1: Address 0000:040E in the BIOS Data Area may be used as the Extended Bios Data Area in PS/2 and newer Bioses, and thus this field may be invalid.

3.3 INTERFACE BOARD

Introduction

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

A block diagram containing the parts of a typical power supply and the voltage at various points in the unit is shown in fig 19.1. The ac voltage, typically 120 V rms, is connected to a transformer, which steps that ac voltage down to the level for the desired dc output. A diode rectifier then provides a full-wave rectified voltage that is initially filtered by a simple capacitor filter to produce a dc voltage. This resulting dc voltage usually has some ripple or ac voltage variation. A regulator circuit can use this dc input to provide a dc voltage that not only has much less ripple voltage but also remains the same dc value even if the input dc voltage varies somewhat, or the load connected to the output dc voltage changes. This voltage regulation is usually obtained using one of a number of popular voltage regulator IC units.

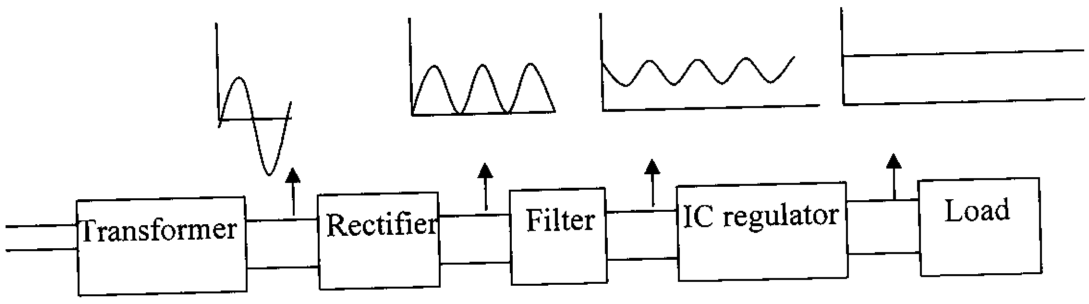


Fig3: Block diagram of power supply

IC voltage regulators

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

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

Three-terminal voltage regulators

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

Fixed Positive Voltage Regulators:

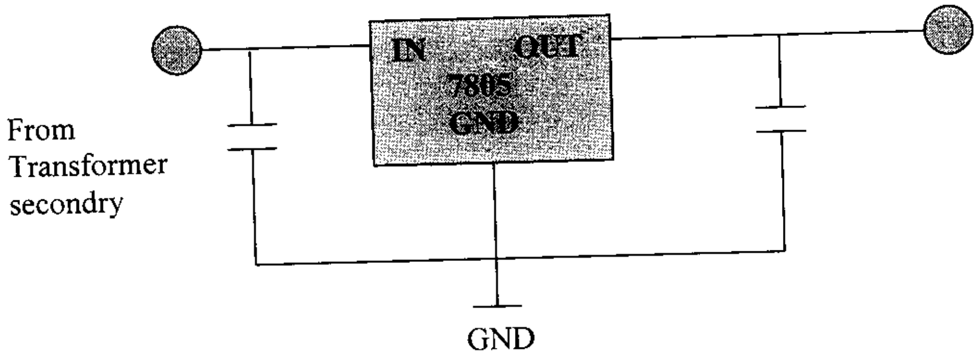


Fig4: Fixed Positive Voltage Regulators

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

IC Part	**Output Voltage (V)**	**Minimum Vi (V)**
7805	+5	7.3
7806	+6	8.3
7808	+8	10.5

7810	+10	12.5
7812	+12	14.6
7815	+15	17.7
7818	+18	21.0
7824	+24	27.1

Table 4: Positive voltage regulators in 7800 series

3.4 POWER SUPPLY

Block diagram

The ac voltage, typically 220V rms, is connected to a transformer, which steps that ac voltage down to the level of the desired dc output. A diode rectifier then provides a full-wave rectified voltage that is initially filtered by a simple capacitor filter to produce a dc voltage. This resulting dc voltage usually has some ripple or ac voltage variation.

A regulator circuit removes the ripples and also remains the same dc value even if the input dc voltage varies, or the load connected to the output dc voltage changes. This voltage regulation is usually obtained using one of the popular voltage regulator IC units.

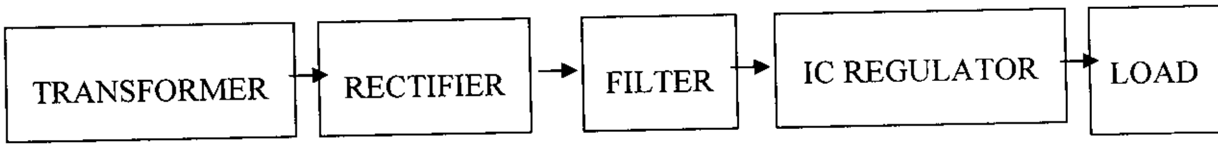


Fig 5: Block diagram (Power supply)

Working principle

Transformer

The potential transformer will step down the power supply voltage (0-230V) to (0-6V) level. Then the secondary of the potential transformer will be connected to the precision rectifier, which is constructed with the help of op–amp. The advantages of using precision rectifier are it will give peak voltage output as DC, rest of the circuits will give only RMS output.

Bridge rectifier

When four diodes are connected as shown in figure, the circuit is called as bridge rectifier. The input to the circuit is applied to the diagonally opposite corners of the network, and the output is taken from the remaining two corners.

Let us assume that the transformer is working properly and there is a positive potential, at point A and a negative potential at point B. the positive potential at point A will forward bias D3 and reverse bias D4.

The negative potential at point B will forward bias D1 and reverse D2. At this time D3 and D1 are forward biased and will allow current flow to pass through them; D4 and D2 are reverse biased and will block current flow.

The path for current flow is from point B through D1, up through RL, through D3, through the secondary of the transformer back to point

B. this path is indicated by the solid arrows. Waveforms (1) and (2) can be observed across D1 and D3.

One-half cycle later the polarity across the secondary of the transformer reverse, forward biasing D2 and D4 and reverse biasing D1 and D3. Current flow will now be from point A through D4, up through RL, through D2, through the secondary of T1, and back to point A. This path is indicated by the broken arrows. Waveforms (3) and (4) can be observed across D2 and D4. The current flow through RL is always in the same direction. In flowing through RL this current develops a voltage corresponding to that shown waveform (5). Since current flows through the load (RL) during both half cycles of the applied voltage, this bridge rectifier is a full-wave rectifier.

One advantage of a bridge rectifier over a conventional full-wave rectifier is that with a given transformer the bridge rectifier produces a voltage output that is nearly twice that of the conventional full-wave circuit.

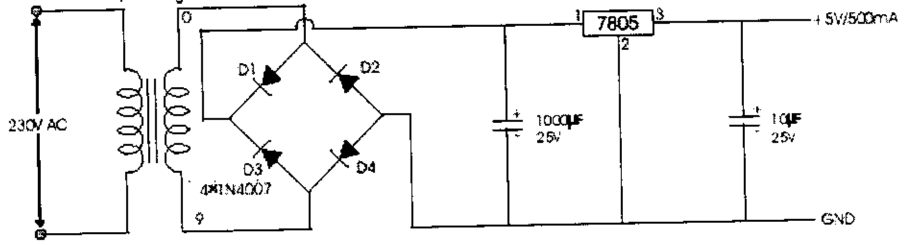
This may be shown by assigning values to some of the components shown in views A and B. assume that the same transformer is used in both circuits. The peak voltage developed between points X and y is 1000 volts in both circuits. In the conventional full-wave circuit shown—in view A, the peak voltage from the center tap to either X or Y is 500 volts. Since only one diode can conduct at any instant, the maximum voltage that can be rectified at any instant is 500 volts.

The maximum voltage that appears across the load resistor is nearly-but never exceeds-500 v0lts, as result of the small voltage drop across the diode. In the bridge rectifier shown in view B, the maximum voltage that can be rectified is the full secondary voltage, which is 1000 volts. Therefore, the peak output voltage across the load resistor is nearly 1000 volts. With both circuits using the same transformer, the bridge rectifier circuit produces a higher output voltage than the conventional full-wave rectifier circuit.

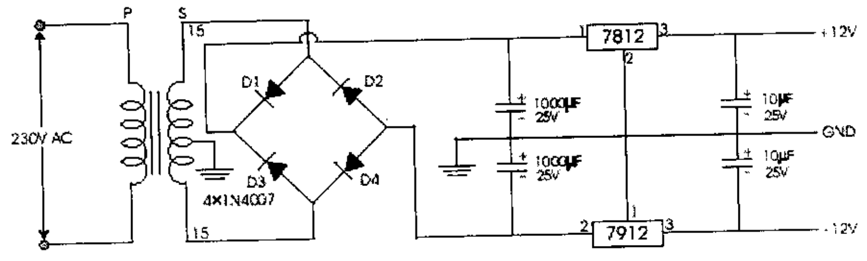
IC voltage regulators

Voltage regulators comprise a class of widely used ICs. Regulator IC units contain the circuitry for reference source, comparator amplifier, control device, and overload protection all in a single IC. IC units provide regulation of either a fixed positive voltage, a fixed negative voltage, or an adjustably set voltage. The regulators can be selected for operation with load currents from hundreds of milli amperes to tens of amperes, corresponding to power ratings from milli watts to tens of watts.

+5V POWER SUPPLY



+12V AND -12V POWER SUPPLY



A fixed three-terminal voltage regulator has an unregulated dc input voltage, Vi, applied to one input terminal, a regulated dc output voltage, Vo, from a second terminal, with the third terminal connected to ground.

The series 78 regulators provide fixed positive regulated voltages from 5 to 24 volts. Similarly, the series 79 regulators provide fixed negative regulated voltages from 5 to 24 volts.

- For ICs, microcontroller, LCD --------- 5 volts
- For alarm circuit, op-amp, relay circuits ---------- 12 volts

3.5 PCB DESIGN

Design and Fabrication of Printed circuit boards

Introduction

Printed circuit boards, or PCBs, form the core of electronic equipment domestic and industrial. Some of the areas where PCBs are intensively used are computers, process control, telecommunications and instrumentation.

Manufatcuring

The manufacturing process consists of two methods; print and etch, and print, plate and etch.

The single sided PCBs are usually made using the print and etch method. The double sided plate through – hole (PTH) boards are made by the print plate and etch method.

The production of multi layer boards uses both the methods. The inner layers are printed and etch while the outer layers are produced by print, plate and etch after pressing the inner layers.

Software

The software used in our project to obtain the schematic layout is MICROSIM.

Panelisation

Here the schematic transformed in to the working positive/negative films. The circuit is repeated conveniently to accommodate economically as many circuits as possible in a panel, which can be operated in every sequence of subsequent steps in the PCB process. This is called penalization. For the PTH boards, the next operation is drilling.

Drilling

PCB drilling is a state of the art operation. Very small holes are drilled with high speed CNC drilling machines, giving a wall finish with less or no smear or epoxy, required for void free through hole plating.

Plating

The heart of the PCB manufacturing process. The holes drilled in the board are treated both mechanically and chemically before depositing the copper by the electro less copper platting process.

Etching

Once a multiplayer board is drilled and electro less copper deposited, the image available in the form of a film is transferred on to the out side by photo printing using a dry film printing process. The boards are then electrolyticaly plated on to the circuit pattern with copper and tin. The tin-plated deposit serves an etch resist when copper in the unwanted area is removed by the conveyorised spray etching machines with chemical etchants. The etching machines are attached to an automatic dosing equipment, which analyses and controls etchants concentrations.

Soldermask

Since a PCB design may call for very close spacing between conductors, a solder mask has to be applied on the both sides of the circuitry to avoid the bridging of conductors. The solder mask ink is applied by screening. The ink is dried, exposed to UV, developed in a mild alkaline solution and finally cured by both UV and thermal energy.

Hot Air Levelling

After applying the solder mask, the circuit pads are soldered using the hot air leveling process. The bare bodies fluxed and dipped in to a molten solder bath. While removing the board from the solder bath, hot air is blown on both sides of the board through air knives in the machines, leaving the board soldered and leveled. This is one of the common finishes given to the boards. Thus the double sided plated through whole printed circuit board is manufactured and is now ready for the components to be soldered.

SERVO MOTOR CONTROL

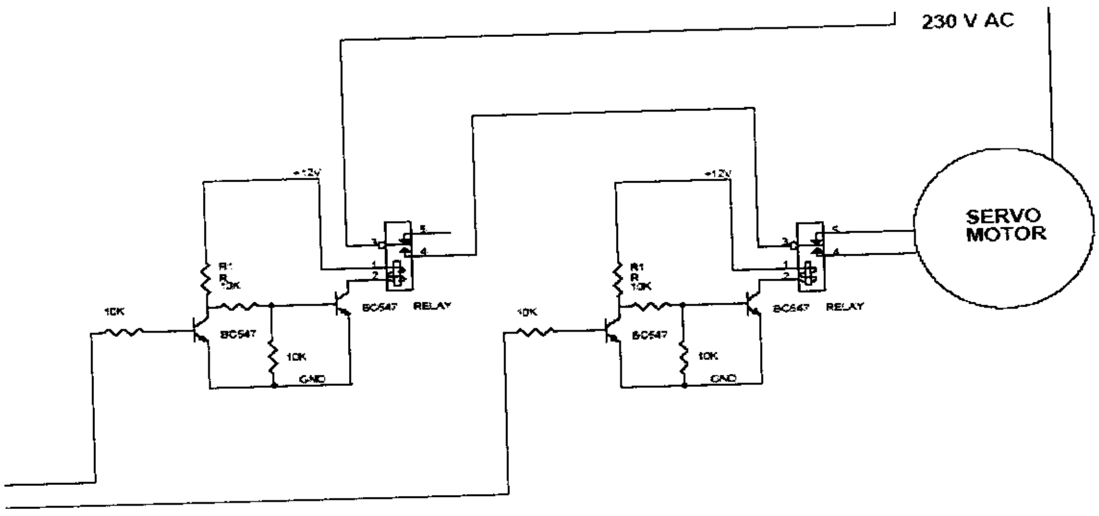


fig 7 :servo motor control

Servo Motor

The servo motor some times referred as servo mechanism. A servomechanism, usually shortened to servo, is a device used to provide mechanical control at a distance. For example, a servo can be used at a remote location to proportionally follow the angular position of a control knob. The connection between the two devices is not mechanical, but electrical or wireless.

The most common type of servo is the one mentioned above, which gives *positional control*. Servos are commonly electrical or partially electronic in nature, using an electric motor as the primary means of creating mechanical force. Other types of servos use hydraulics, pneumatics, or magnetic principles. Usually, servos operate on the principle of negative feedback, where the control input is compared to the actual position of the mechanical system as measured by some sort of transducer at the output. Any difference between the actual and wanted values (an "error signal") is amplified and used to drive the system in the direction necessary to reduce or eliminate the error. An entire science known as control theory has been developed on this type of system.

Servos are found in many applications. They operate the throttle of engines that use a cruise control. CNC machines use servos to make the motion axes of a machine tool follow the desired tool path. Fly-by-wire systems in aircraft use servos to actuate the control surfaces that control the aircraft. Radio-controlled airplanes use servos for the same purpose. Many autofocus cameras also use a servomechanism to accurately move the lens, and thus adjust the focus.

Typical servos give a rotary (angular) output. Linear types are common as well, using a screw thread or a linear motor to give linear motion. Another device commonly referred to as a servo is used in automobiles to amplify the steering or braking force applied by the driver. In this form this device is not a true servo, but rather a mechanical amplifier.

Circuit Working Description

In this circuit the servo motor is controlled through two relays in which the both the relays are controlled by the microcontroller with the help of transistor network. The transistor network is consists of two set of

BC547. The relay is connected in the T2 transistor collector terminal. A Relay is nothing but electromagnetic switching device which consists of three pins. They are Common, Normally close (NC) and normally open (NO).

The relay common pin is connected to supply voltage. The normally open (NO) pin connected to common terminal of second relay. The servo motor is connected in the NO and NC pin of the second relay. When high pulse signal is given to base of the T1 transistors, the transistor is conducting and shorts the collector and emitter terminal and zero signals is given to base of the T2 transistor. So the relay is turned OFF state.

When low pulse is given to base of transistor T1 transistor, the transistor is turned OFF. Now 12v is given to base of T2 transistor so the transistor is conducting and relay is turned ON. Hence the common terminal and NO terminal of relay are shorted. Now servo motor gets the supply voltage through relay.

3.6 X-Y TRACKING ROBOT

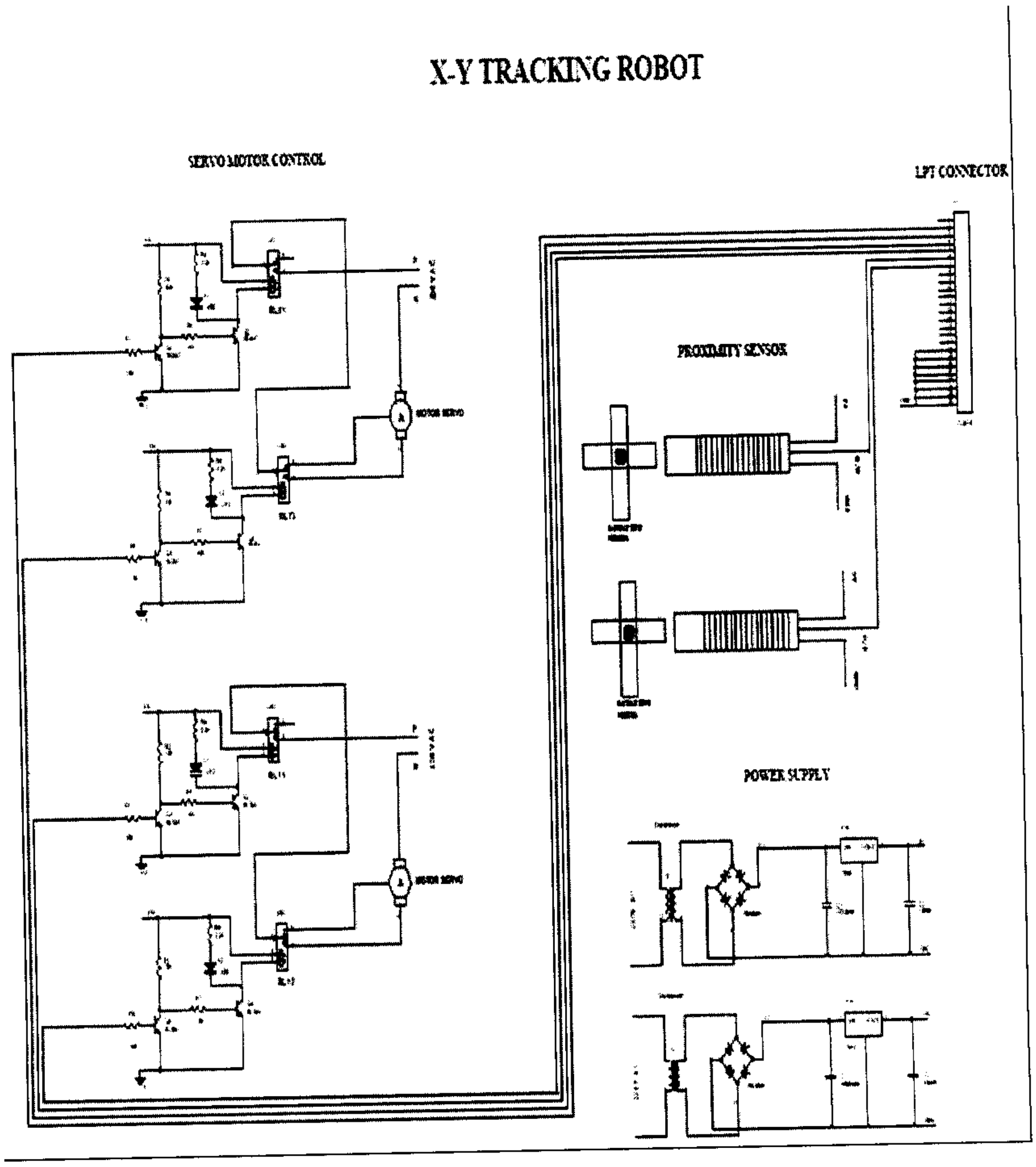


Fig 8: X-Y tracking robot

IMAGE PROCESSING

4. IMAGE PROCESSING

The toolboxes that we have used for image processing are the image acquisition toolbox, the image processing toolbox and Dipum toolbox

➢ **Image Processing Tool Box**

The image processing toolbox in MATLAB allows us to convert images into digital data, operate on the digital data and convert the data again into an image.

The commands that we have used in our project pertaining to the image processing toolbox are

4.1 IMREAD

A = Imread (FILENAME, FMT) reads the image in filename into A. If the

IMREAD Read image from graphics file.

A = IMREAD(FILENAME,FMT) reads a grayscale or color image from the file specified by the string FILENAME, where the string FMT specifies the format of the file. See the reference page, or the output of the function IMFORMATS, for a list of supported formats. If the file is not in the current directory or in a directory in the MATLAB path, specify the full pathname of the location on your system. If IMREAD cannot find a file named FILENAME, it looks for a file named FILENAME.FMT.

IMREAD returns the image data in the array A. If the file contains a grayscale image, A is a two-dimensional (M-by-N) array. If the file contains a color image, A is a three-dimensional (M-by-N-by-3) array. The class of the returned array depends on the data type used by the file format.

4.2 IMSHOW

IMSHOW(I,N) displays the intensity image I with N discrete levels of gray. If you omit N, IMSHOW uses 256 gray levels on 24-bit displays, or 64 gray levels on other systems.

IMSHOW(I,[LOW HIGH]) displays I as a grayscale intensity image, specifying the data range for I. The value LOW (and any value less than LOW) displays as black, the value HIGH (and any value greater than HIGH) displays as white, and values in between display as intermediate shades of gray. IMSHOW uses the default number of gray levels. If you use an empty matrix ([]) for [LOW HIGH], IMSHOW uses [min(I(:)) max(I(:))]; the minimum value in I displays as black, and the maximum value displays as white.

4.3 IMADJUST

IMADJUST Adjusts the image intensity values or colormap.

J = IMADJUST(I) maps the values in intensity image I to new values in J such that 1% of data is saturated at low and high intensities of I. This increases the contrast of the output image J.

J = IMADJUST(I,[LOW_IN; HIGH_IN],[LOW_OUT; HIGH_OUT]) maps the values in intensity image I to new values in J such that values between LOW_IN and HIGH_IN map to values between LOW_OUT and HIGH_OUT.

Examples

```
I = imread('pout.tif');
J = imadjust(I);
imview(I), imview(J)

K = imadjust(I,[0.3 0.7],[]);
imview(K)

RGB1 = imread('football.jpg');
RGB2 = imadjust(RGB1,[.2 .3 0; .6 .7 1],[]);
imview(RGB1), imview(RGB2)
```

4.4 RGB2GRAY

RGB2GRAY Convert RGB image or colormap to grayscale.

RGB2GRAY converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

I = RGB2GRAY(RGB) converts the truecolor image RGB to the grayscale intensity image I.

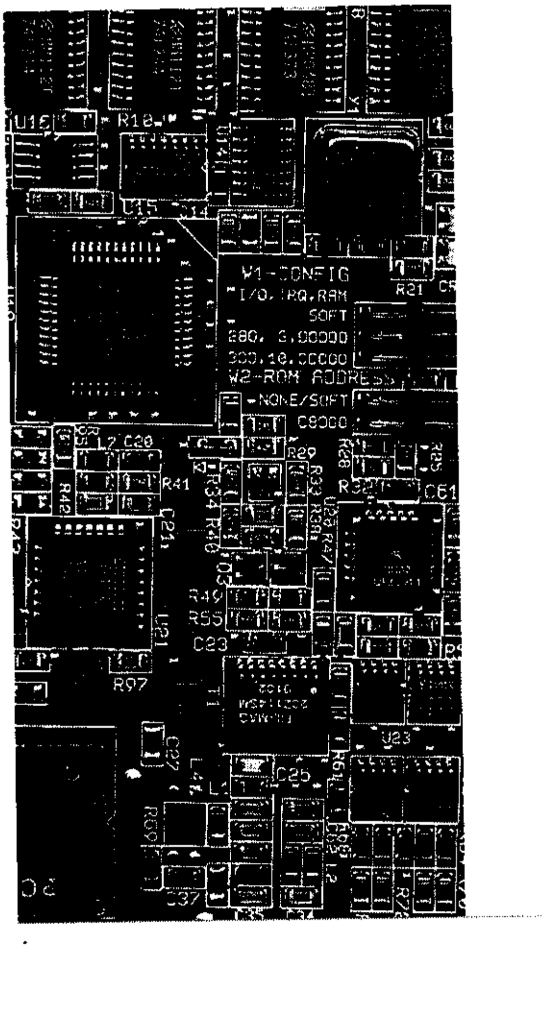
NEWMAP = RGB2GRAY(MAP) returns a grayscale colormap equivalent to MAP.

Example

```
I = imread('board.tif');
J = rgb2gray(I);
imview(I), imview(J);

[X,map] = imread('trees.tif');
gmap = rgb2gray(map);
imview(X,map), imview(X,gmap);
```

SAMPLE IMAGE



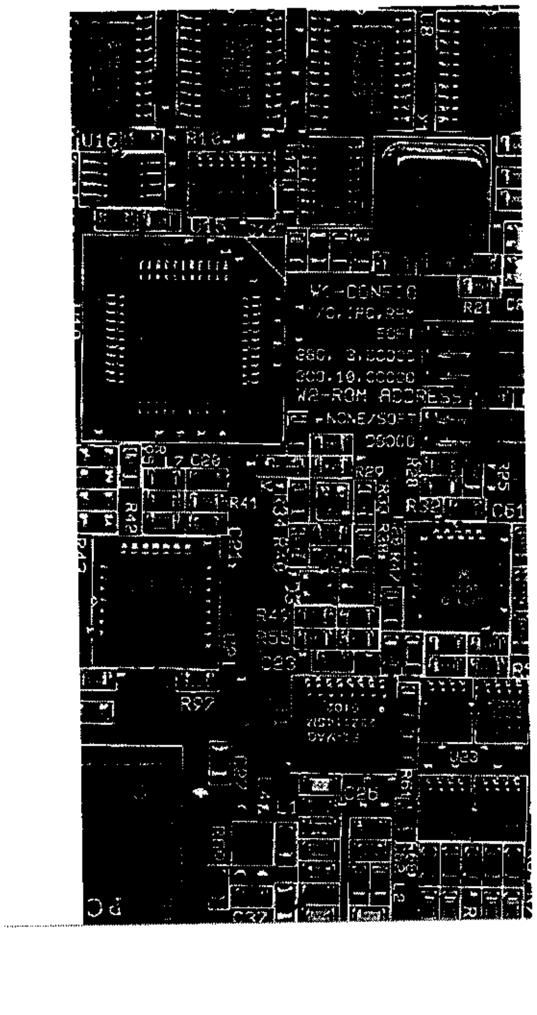


IMAGE ACQUISITION
TOOL BOX

5. IMAGE ACQUISITION TOOLBOX

The image acquisition toolbox is used to bring images from the external environment into MATLAB. The commands used in our project are

Image Processing Toolbox 4.2 Release Notes

New Features

Enhanced DICOM support

The dicomwrite function can now write data in any modality that can be read using dicomread. Note, however, that when writing data in these modalities, dicomwrite does not verify the data or check to see if the correct amount of data is written. The toolbox can now read and write private metadata, even if they are not defined in the DICOM data dictionary. When private metadata is not in the data dictionary, dicomread uses generic names for the metadata fields, rather than the descriptive names available to attributes defined in the data dictionary. You can create a custom data dictionary that contains definitions of your private metadata, using the dicomdict function

5.1 VIDEOINPUT

Create a video input object.

```
OBJ = VIDEOINPUT(ADAPTORNAME)
OBJ = VIDEOINPUT(ADAPTORNAME, DEVICEID)
OBJ = VIDEOINPUT(ADAPTORNAME, DEVICEID, FORMAT)
```

constructs a video input, object, OBJ, using:

ADAPTORNAME - a string specifying the device adaptor OBJ is associated with. Valid values can be determined by using the

IMAQHWINFO function.

DEVICEID - a numerical device identifier. If DEVICEID is not specified, the first available device ID is used.

FORMAT - a string specifying the video format for OBJ. If FORMAT is not specified, the device's default format is used.

Valid DEVICEID and FORMAT values, as well as the default format, can be determined by using the IMAQHWINFO (ADAPTORNAME) syntax.

Upon creation, OBJ's VideoFormat property will reflect the specified FORMAT. The first available video source will also be selected and indicated by OBJ's Selected Source Name property. Use GETSELECTEDSOURCE(OBJ) to access the video source object that will be used for acquisition.

Example:

```
% Construct a video input object associated
% with a Matrox device at ID 1:
obj = videoinput('matrox', 1);

% Select the source to use for acquisition.
```

```
set(obj, 'SelectedSourceName', 'input1')

% View the properties for the selected video source object.
src_obj = getselectedsource(obj);
get(src_obj)

% Preview a stream of image frames:
preview(obj);

% Acquire and display a single image frame:
frame = getsnapshot(obj);
image(frame);
 % Remove video input object from memory:
delete(obj);
```

5.2 SET

Set object properties.

SET(H,'PropertyName',PropertyValue) sets the value of the specified property for the graphics object with handle H. H can be a vector of handles, in which case SET sets the properties' values for all the objects.

SET(H,a) where a is a structure whose field names are object property names, sets the properties named in each field name with the values contained in the structure.

SET(H, pn, pv) sets the named properties specified in the cell array of strings pn to the corresponding values in the cell array pv for all objects specified in H. The cell array pn must be 1-by-N, but the cell array pv can be M-by-N where M is equal to length(H) so that each object will be updated with a different set of values for the list of property names contained in pn.

5.3 PREVIEW

Activate a live video preview window.

PREVIEW(OBJ) immediately activates a live video preview window for video input object OBJ. The size of the previewed images reflects the ROIPosition property configuration. The preview window remains active until it is closed with CLOSEPREVIEW.

The behavior of the preview window depends on the object's current state and trigger configuration.

While an object is stopped (Running=off), the preview window shows a live view of the image being acquired from the device, for all trigger types. The image is updated to reflect changes made to configurations of object properties. (The FrameGrabInterval property is ignored until a trigger occurs.)

When an object is started (Running=on), the behavior of the preview preview window depends on the trigger type. If the trigger type is set to immediate or manual, the preview window continues to update the image displayed. If the trigger type is set to hardware, the preview window stops updating the image displayed until a trigger occurs.

Upon deleting OBJ with DELETE, any associated preview windows will be closed.

5.4 STOP

Stop timer(s).

STOP(OBJ) stops the timer, represented by the timer object, OBJ. If OBJ is an array of timer objects, STOP stops all of the timers. Use the TIMER function to create a timer object.

STOP sets the Running property of the timer object, OBJ, to 'Off', halts further TimerFcn callbacks, and executes the StopFcn callback.

5.5 CLOSEPREVIEW

Close image preview window.

CLOSEPREVIEW(OBJ) closes the image preview window associated with the image acquisition object OBJ.

CLOSEPREVIEW closes all image preview windows for all image acquisition objects.

IMAGE PROCESSING CODE

6. IMAGE PROCESSING CODE

Complete Image Processing Code

```
disp('DEMONSTRATI2ON OF LEARNING ABILITIES');
disp('');
b=1;
while b==1;
disp('Enter a choice');
disp('enter 1 for learning phase');
disp('enter 2 for demonstration phase');
disp('enter 3 to terminate');
k=input('enter :');
switch k

    case 1
        disp('welcome to the learning phase');
        no=input('enter the number of objects :');
        for m= 1:no
        l= getsnapshot(OBJ);
        l=imadjust(l,[.43 1],[0 1]);

          %f=imread(l);
          a=rgb2gray(l);
          phi=invmoments(a);
          phi=log(phi);
          phi=abs(phi);
          s=input('enter object name :','s');
```

```
     N{m}=s;
     DESCP{m}=phi;
     %imshow(f);

   end

case 2
    disp('welcome to the demonstration phase');
    mark=0;
    %l=input('enter path for image :','s');
      %f=imread(l);
      l= getsnapshot(OBJ);
      l=imadjust(l,[.43 1],[0 1]);
      a=rgb2gray(l);
      phi=invmoments(a);
      phi=log(phi);
      phi=abs(phi);
      for m= 1:no
         if norm(phi-DESCP{m})>0 & norm(phi-DESCP{m})<7
            disp('the object found is :');
            disp(N{m})
            mark=1;

         end

       end
       if mark==0
```

```
        disp('unknown object found');


      end
  case 3
     b=0;;
     otherwise
     disp('ok');

end
end
```

OUTPUT

DEMONSTRATION OF LEARNING ABILITIES

```
Enter a choice
enter 1 for learning phase
enter 2 for demonstration phase
enter 3 to terminate
enter :1
welcome to the learning phase
enter the number of objects :3
enter object name : mobile
enter object name: strip
enter  object name: box
Enter a choice
enter 1 for learning phase
enter 2 for demonstration phase
enter 3 to terminate
enter :2
welcome to the demonstration  phase
the object found is :box
unknown object found
the object found is :mobile
the object found is :strip
unknown object found
Enter a choice
enter 1  for learning phase
enter 2  for demonstration phase
enter 3 to terminate
enter:3
```

AUDIO PROCESSING

7. AUDIO PROCESSING

➢ Wavrecord

WAVRECORD Record sound using Windows audio input device. WAVRECORD(N,FS,CH) records N audio samples at FS Hertz from CH number of input channels from the Windows WAVE audio device. Standard audio rates are 8000, 11025, 22050, and 44100 Hz. CH can be 1 or 2 (mono or stereo). Samples are returned in a matrix of size N x CH. If not specified, FS=11025 Hz, and CH=1.

WAVRECORD(..., DTYPE) records and returns data using the data type specified by DTYPE. Supported data types and the corresponding number of bits per sample recorded in each format are as follows:

DTYPE	bits/sample
'double'	16
'single'	16
'int16'	16
'uint8'	8

This function is only for use with 32-bit Windows machines.

Example: Record and play back 5 seconds of 16-bit audio sampled at 11.025 kHz.

```
Fs = 11025;
y  = wavrecord(5*Fs, Fs, 'int16');
wavplay(y, Fs);
```

➢ **Wavplay**

WAVPLAY Play sound using Windows audio output device. WAVPLAY(Y,FS) sends the signal in vector Y with sample frequency of FS Hertz to the Windows WAVE audio device. Standard audio rates are 8000, 11025, 22050, and 44100 Hz. WAVPLAY(Y) automatically sets the sample rate to 11025 Hz. For stereo playback, Y should be an N-by-2 matrix.

WAVPLAY(...,'async') begins sound playback and returns immediately from the function call (i.e., a nonblocking call). WAVPLAY(...,'sync') does not return from the function call until the sound has finished playing (i.e., a blocking call). This is the default playback mode.

Y must contain audio samples stored in double, int16, or uint8 matrices. Double precision data samples must be in the range -1.0 <= y <= 1.0; values outside that range are clipped.

Supported data types for Y and the corresponding number of bits per sample used during playback in each format are as follows:

Data Type	bits/sample
'double'	16
'single'	16
'int16'	16
'uint8'	8

This function is only for use with 32-bit Windows machines.

GRAPHICAL USER INTERFACE

8. GRAPHICAL USER INTERFACE

A graphical user interface enhances usability and improves user interaction. We have designed a GUI for LASER in the following manner.

Type GUIDE in the command window

>>GUIDE

This launches MATLAB figure window. The required command buttons are placed on the figure window. By double clicking the component its code can be edited.

CODE WITH AUDIO OUTPUT AND GUI:

```
function varargout = laser(varargin)
% LASER M-file for laser.fig
%      LASER, by itself, creates a new LASER or raises the existing
%      singleton*.
%
%      H = LASER returns the handle to a new LASER or the handle to
%      the existing singleton*.
%
%      LASER('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in LASER.M with the given input
arguments.
%
```

```
%    LASER('Property','Value',...) creates a new LASER or raises the
%    existing singleton*.  Starting from the left, property value pairs are
%    applied to the GUI before laser_OpeningFunction gets called.  An
%    unrecognized property name or invalid value makes property
application
%    stop.  All inputs are passed to laser_OpeningFcn via varargin.
%
%    *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%    instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Copyright 2002-2003 The MathWorks, Inc.
% Edit the above text to modify the response to help laser
% Last Modified by GUIDE v2.5 19-Jan-2007 15:32:22
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
             'gui_Singleton',  gui_Singleton, ...
             'gui_OpeningFcn', @laser_OpeningFcn, ...
             'gui_OutputFcn',  @laser_OutputFcn, ...
             'gui_LayoutFcn',  [] , ...
             'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
```

```
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code – DO NOT EDIT
% --- Executes just before laser is made visible.
Function laser_OpeningFcn(hObject, eventdata, handles, varargin)
 global OBJ;
 OBJ = VIDEOINPUT('winvideo')
  set(OBJ, 'SelectedSourceName', 'input1')
  preview(OBJ);
 a=imread('blank.jpg');
 imshow(a);
 global m;
 %init;
 m=0;
 % This function has no output args, see OutputFcn.
 % hObject    handle to figure
 % eventdata  reserved – to be defined in a future version of MATLAB
 % handles    structure with handles and user data (see GUIDATA)
 % varargin   command line arguments to laser (see VARARGIN)
 % Choose default command line output for laser
 handles.output = hObject;
 % Update handles structure
 guidata(hObject, handles);
 % UIWAIT makes laser wait for user response (see UIRESUME)
 % uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
Function varargout = laser_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double
% --- Executes during object creation, after setting all properties.
Function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    empty – handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
If ispc
    set(hObject,'BackgroundColor','white');
else
```

```
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor
'));
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved – to be defined in a future version of  MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double

% --- Executes during object creation, after setting all properties.
Function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    empty – handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
If ispc
   set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor
'));
```

```
end

% --- Executes on button press in pushbutton2.
Function pushbutton2_Callback(hObject, eventdata, handles)
global OBJ;
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 global m;
 global N;
  global DESCP;
 m=m+1;
 l= getsnapshot(OBJ);
 imshow(l)
 l=imadjust(l,[.2 1],[0 1]);
         a=rgb2gray(l);
                 phi=invmoments(a);
         phi=log(phi);
         phi=abs(phi);
         nob=wavread('name.wav');
         nob=nob*10;
         wavplay(nob,28000);
         pause(1);
         beep
         Fs = 11025;
        y  = wavrecord(2*Fs, Fs, 'int16');
        y=y*10;
```

```
    N{m}=y;
      DESCP{m}=phi;
% --- Executes on button press in pushbutton3.
Function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
mark=0;
global m;global OBJ;
global N;
global DESCP;
l= getsnapshot(OBJ);
imshow(l);
        l=imadjust(l,[.2 1],[0 1]);
        a=rgb2gray(l);


        [y,x]=find(a~=0);
        y=y(10);
        x=x(10);
        phi=invmoments(a);
        phi=log(phi);
        phi=abs(phi);
        for x= 1:m
           if norm(phi-DESCP{x})>0 & norm(phi-DESCP{x})<7
              %disp('the object found is :');
              wavplay(N{x}, 8000);
              mark=1;
```

```
         end
       end
if mark==0
   nob=wavread('unknown.wav');
        nob=nob*10;
        wavplay(nob,28000);


       end


% --- Executes on button press in pushbutton4.
Function pushbutton4_Callback(hObject, eventdata, handles)
global OBJ;
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
stop(OBJ);
closepreview;
close ;
```

CONCLUSION

9. CONCLUSION

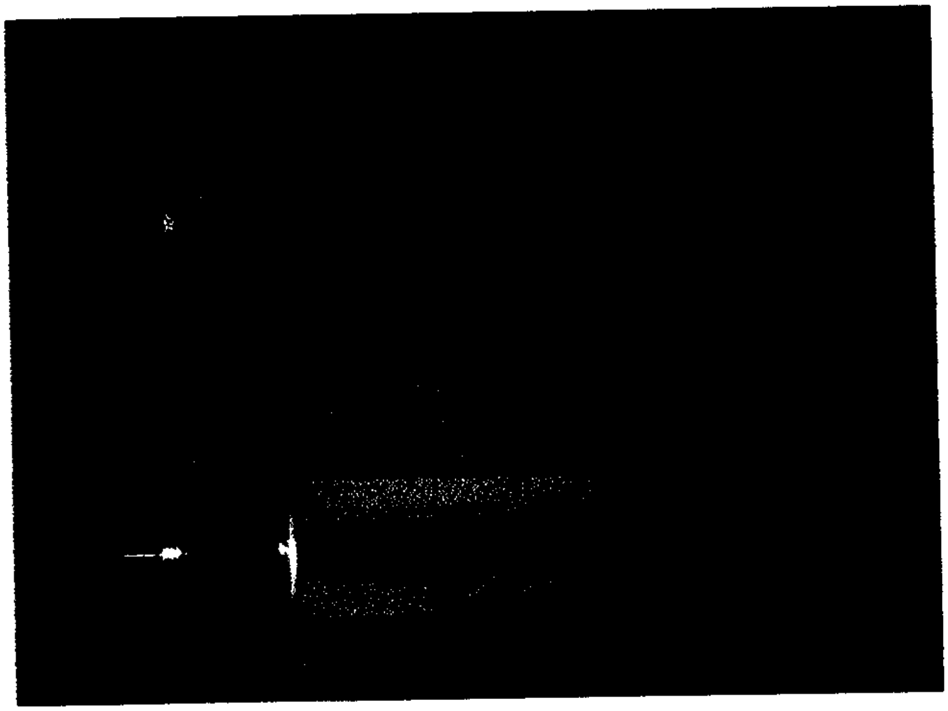
RESULT

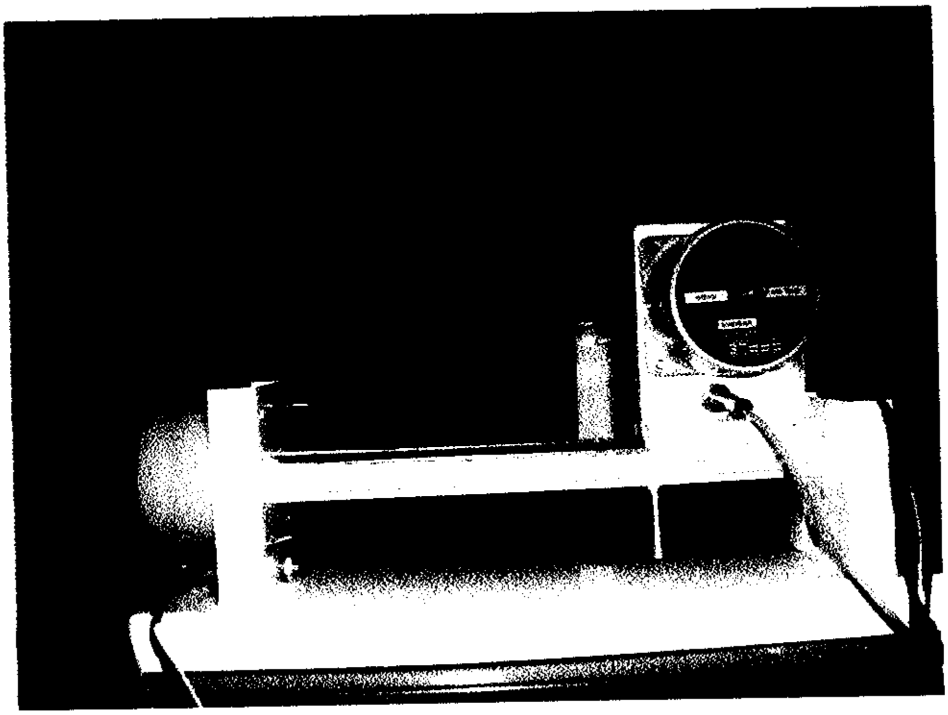
All the objectives of our project successfully achieved and the satisfaction we derived from our work was over whelming.

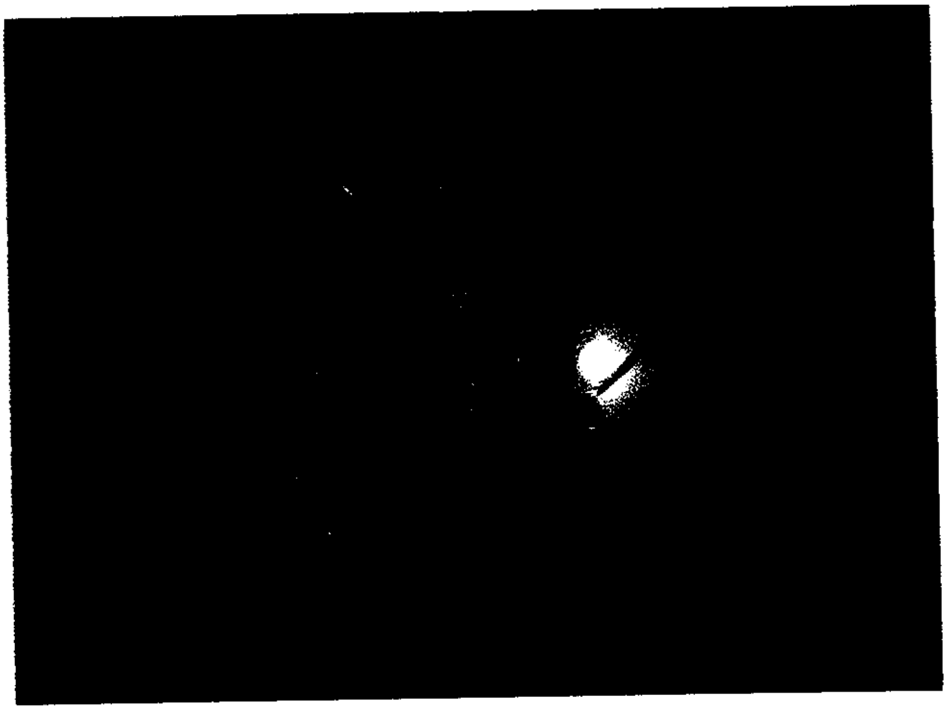
After many attempts that went in vain we were finally rewarded for all our efforts. The X-Y Robotic manipulated that we constructed though a little slow, satisfies all constrains originally proposed.

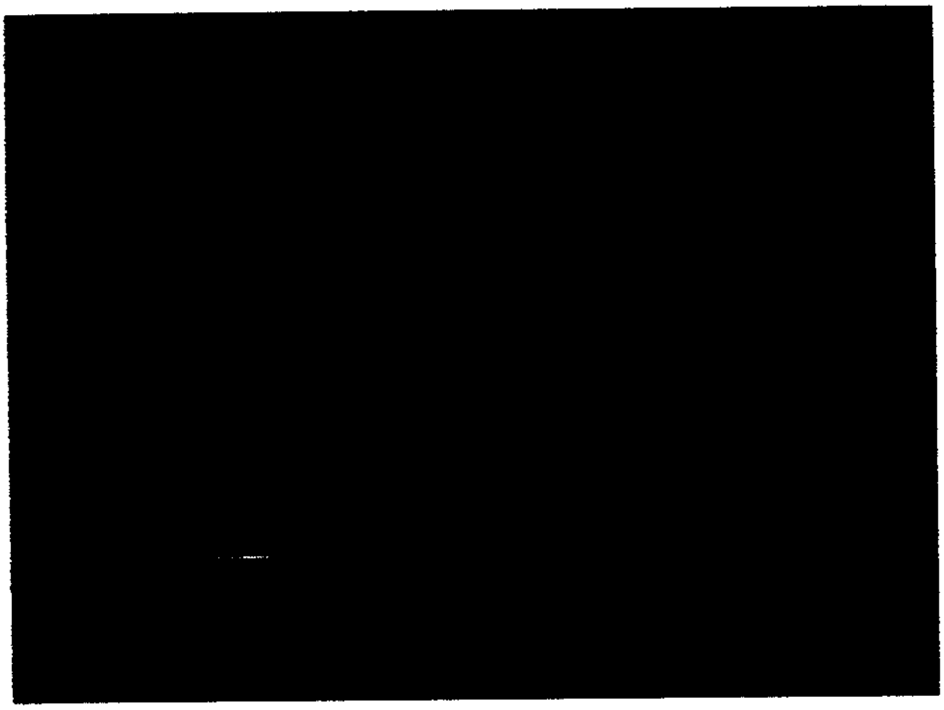
We are sure that our project will be helpful to future attempts for designing similar system.

PHOTOS









FUTURE ENHANCEMENTS

- The speed of the arm could be increased to enhance the performance
- One object is recognize at a time the code can be rectified to identify multiple objects.
- Instead of using PC independent processor can be used.

APPENDIX

FAIRCHILD
SEMICONDUCTOR™

August 1986
Revised March 2000

DM74LS14
Hex Inverter with Schmitt Trigger Inputs

General Description

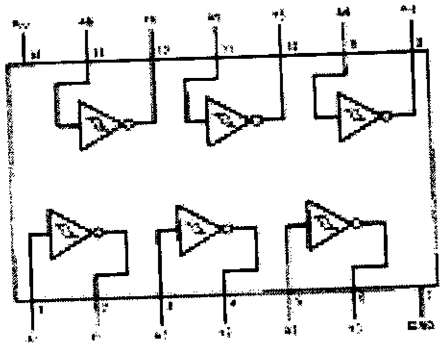
This device contains six independent gates each of which performs the logic INVERT function. Each input has hysteresis which increases the noise immunity and transforms a slowly changing input signal to a fast changing, jitter free output.

Ordering Code:

Order Number	Package Number	Package Description
DM74LS14M	M14A	14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-120, 0.150 Narrow
DM74LS14SJ	M14D	14-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS14N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

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

Connection Diagram



Function Table

$Y = \overline{A}$

Input	Output
A	Y
L	H
H	L

H = HIGH Logic Level
L = LOW Logic Level

Absolute Maximum Ratings (Note 1)

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

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{T+}	Positive-Going Input Threshold Voltage (Note 2)	1.4	1.6	1.9	V
V_{T-}	Negative-Going Input Threshold Voltage (Note 2)	0.5	0.8	1	V
HYS	Input Hysteresis (Note 2)	0.4	0.8		V
I_{OH}	HIGH Level Output Current			−0.4	mA
I_{OL}	LOW Level Output Current			8	mA
T_A	Free Air Operating Temperature	0		70	°C

Note 2: $V_{CC} = 5V$.

Electrical Characteristics

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

Symbol	Parameter	Conditions	Min	Typ (Note 3)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, $I_I = -18$ mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max	2.7	3.4		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IH} = Min		0.35	0.5	V
		V_{CC} = Min, I_{OL} = 4 mA		0.25	0.4	
I_{T+}	Input Current at Positive-Going Threshold	$V_{CC} = 5V$, $V_I = V_{T+}$		−0.14		mA
I_{T-}	Input Current at Negative-Going Threshold	$V_{CC} = 5V$, $V_I = V_{T-}$		−0.18		mA
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, $V_I = 7V$			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, $V_I = 2.7V$			20	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, $V_I = 0.4V$			−0.4	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 4)	−20		−100	mA
I_{CCH}	Supply Current with Outputs HIGH	V_{CC} = Max		8.6	16	mA
I_{CCL}	Supply Current with Outputs LOW	V_{CC} = Max		12	21	mA

Note 3: All typicals are at $V_{CC} = 5V$, $T_A = 25°C$.

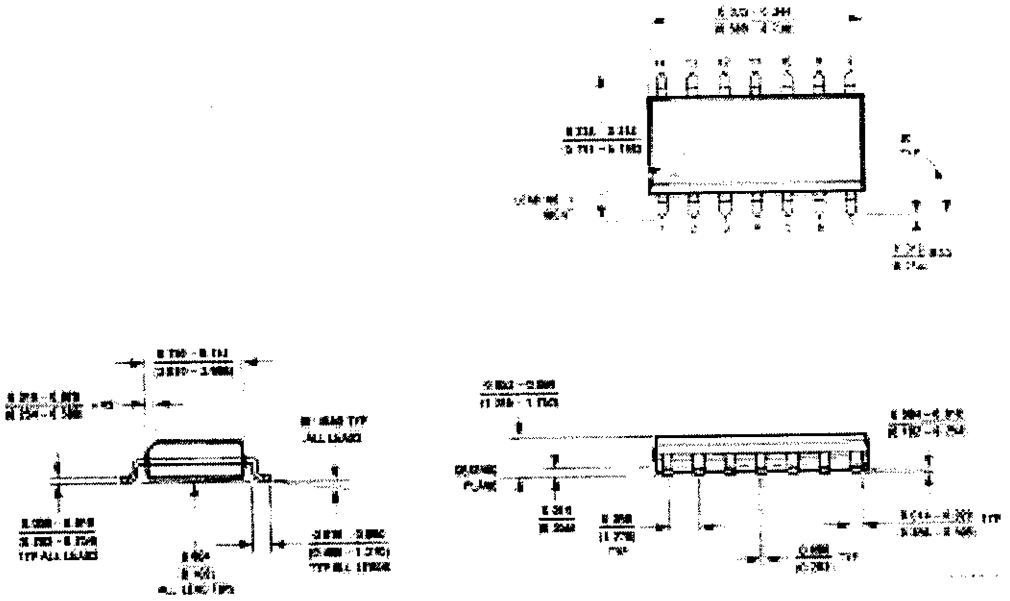
Note 4: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Switching Characteristics

at $V_{CC} = 5V$ and $T_A = 25°C$

Symbol	Parameter	$R_L = 2\ k\Omega$				Units
		$C_L = 15$ pF		$C_L = 50$ pF		
		Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	5	22	8	25	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	5	22	10	33	ns

Physical Dimensions inches (millimeters) unless otherwise noted



14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-120, 0.150 Narrow
Package Number M14A

NPN general purpose transistors

BC546; BC547

FEATURES

- Low current (max. 100 mA)
- Low voltage (max. 65 V).

APPLICATIONS

- General purpose switching and amplification.

DESCRIPTION

NPN transistor in a TO-92; SOT54 plastic package.
PNP complements: BC556 and BC557.

PINNING

PIN	DESCRIPTION
1	emitter
2	base
3	collector

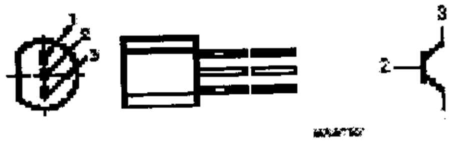


Fig.1 Simplified outline (TO-92; SOT54) and symbol.

LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter			
	BC546		–	80	V
	BC547		–	50	V
V_{CEO}	collector-emitter voltage	open base			
	BC546		–	65	V
	BC547		–	45	V
V_{EBO}	emitter-base voltage	open collector			
	BC546		–	6	V
	BC547		–	6	V
I_C	collector current (DC)		–	100	mA
I_{CM}	peak collector current		–	200	mA
I_{BM}	peak base current		–	200	mA
P_{tot}	total power dissipation	$T_{amb} \leq 25$ °C; note 1	–	500	mW
T_{stg}	storage temperature		–65	+150	°C
T_j	junction temperature		–	150	°C
T_{amb}	operating ambient temperature		–65	+150	°C

Note

1. Transistor mounted on an FR4 printed-circuit board.

THERMAL CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	VALUE	UNIT
$R_{th\,j-a}$	thermal resistance from junction to ambient	note 1	0.25	K/mW

Note

1. Transistor mounted on an FR4 printed-circuit board.

CHARACTERISTICS

$T_j = 25$ °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
I_{CBO}	collector cut-off current	$I_E = 0$; $V_{CB} = 30$ V	–	–	15	nA
		$I_E = 0$; $V_{CB} = 30$ V; $T_j = 150$ °C	–	–	5	µA
I_{EBO}	emitter cut-off current	$I_C = 0$; $V_{EB} = 5$ V	–	–	100	nA
h_{FE}	DC current gain	$I_C = 10$ µA; $V_{CE} = 5$ V; see Figs 2, 3 and 4				
	BC546A		–	90	–	
	BC546B; BC547B		–	150	–	
	BC547C		–	270	–	
	DC current gain	$I_C = 2$ mA; $V_{CE} = 5$ V; see Figs 2, 3 and 4				
	BC546A		110	180	220	
	BC546B; BC547B		200	290	450	
	BC547C		420	520	800	
	BC547		110	–	800	
	BC546		110	–	450	
V_{CEsat}	collector-emitter saturation voltage	$I_C = 10$ mA; $I_B = 0.5$ mA	–	90	250	mV
		$I_C = 100$ mA; $I_B = 5$ mA	–	200	600	mV
V_{BEsat}	base-emitter saturation voltage	$I_C = 10$ mA; $I_B = 0.5$ mA; note 1	–	700	–	mV
		$I_C = 100$ mA; $I_B = 5$ mA; note 1	–	900	–	mV
V_{BE}	base-emitter voltage	$I_C = 2$ mA; $V_{CE} = 5$ V; note 2	580	660	700	mV
		$I_C = 10$ mA; $V_{CE} = 5$ V	–	–	770	mV
C_c	collector capacitance	$I_E = i_e = 0$; $V_{CB} = 10$ V; f = 1 MHz	–	1.5	–	pF
C_e	emitter capacitance	$I_C = i_c = 0$; $V_{EB} = 0.5$ V; f = 1 MHz	–	11	–	pF
f_T	transition frequency	$I_C = 10$mA; $V_{CE} = 5$ V; f = 100 MHz	100	–	–	MHz
F	noise figure	$I_C = 200$ µA; $V_{CE} = 5$ V; $R_S = 2$ kΩ; f = 1 kHz; B = 200 Hz	–	2	10	dB

Notes

1. V_{BEsat} decreases by about 1.7 mV/K with increasing temperature.
2. V_{BE} decreases by about 2 mV/K with increasing temperature.

National Semiconductor

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

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

Features

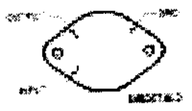
- Output current in excess of 1A
- Internal thermal overload protection
- No external components required
- Output transistor safe area protection
- Internal short circuit current limit
- Available in the aluminum TO-3 package

Voltage Range

LM7805C	5V
LM7812C	12V
LM7815C	15V

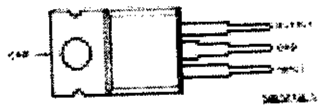
Connection Diagrams

Metal Can Package
TO-3 (K)
Aluminum



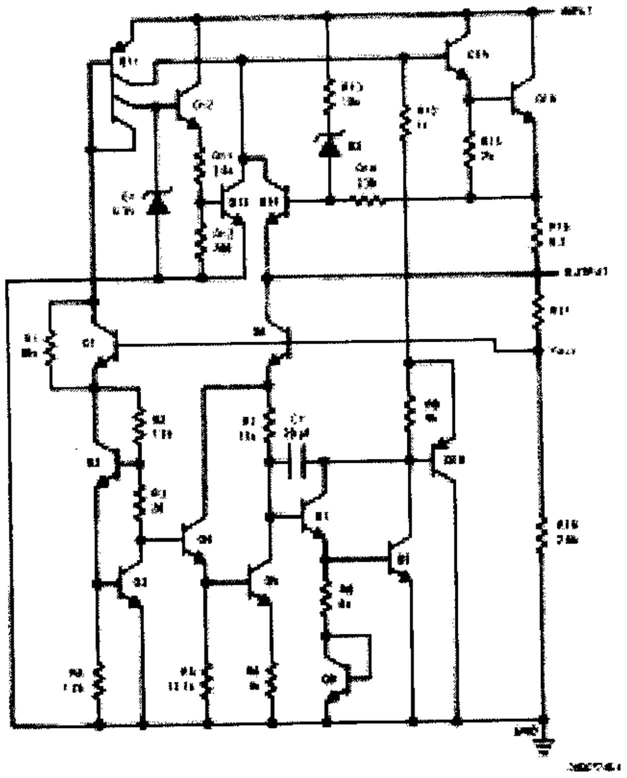
Bottom View
Order Number LM7805CK,
LM7812CK or LM7815CK
See NS Package Number KC02A

Plastic Package
TO-220 (T)



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

Schematic



Absolute Maximum Ratings (Note 3)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Input Voltage	
(V_O = 5V, 12V and 15V)	35V
Internal Power Dissipation (Note 1)	Internally Limited
Operating Temperature Range (T_A)	0°C to +70°C
Maximum Junction Temperature	
(K Package)	150°C
(T Package)	150°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	
TO-3 Package K	300°C
TO-220 Package T	230°C

Electrical Characteristics LM78XXC (Note 2)

0°C ≤ T_J ≤ 125°C unless otherwise noted

				5V			12V			15V			
Output Voltage				5V			12V			15V			Units
Input Voltage (unless otherwise noted)				10V			19V			23V			
Symbol	Parameter	Conditions		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
V_O	Output Voltage	Tj = 25°C, 5 mA ≤ I_O ≤ 1A		4.8	5	5.2	11.5	12	12.5	14.4	15	15.6	V
		P_D ≤ 15W, 5 mA ≤ I_O ≤ 1A		4.75		5.25	11.4		12.6	14.25		15.75	V
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(7.5 ≤ V_{IN} ≤ 20)			(14.5 ≤ V_{IN} ≤ 27)			(17.5 ≤ V_{IN} ≤ 30)			V
ΔV_O	Line Regulation	I_O = 500 mA	Tj = 25°C		3	50		4	120		4	150	mV
			ΔV_{IN}	(7 ≤ V_{IN} ≤ 25)			(14.5 ≤ V_{IN} ≤ 30)			(17.5 ≤ V_{IN} ≤ 30)			V
			0°C ≤ Tj ≤ +125°C			50			120			150	mV
			ΔV_{IN}	(8 ≤ V_{IN} ≤ 20)			(15 ≤ V_{IN} ≤ 27)			(18.5 ≤ V_{IN} ≤ 30)			V
		I_O ≤ 1A	Tj = 25°C			50			120			150	mV
			ΔV_{IN}	(7.5 ≤ V_{IN} ≤ 20)			(14.6 ≤ V_{IN} ≤ 27)			(17.7 ≤ V_{IN} ≤ 30)			V
			0°C ≤ Tj ≤ +125°C			25			60			75	mV
			ΔV_{IN}	(8 ≤ V_{IN} ≤ 12)			(16 ≤ V_{IN} ≤ 22)			(20 ≤ V_{IN} ≤ 26)			V
ΔV_O	Load Regulation	Tj = 25°C	5 mA ≤ I_O ≤ 1.5A		10	50		12	120		12	150	mV
			250 mA ≤ I_O ≤ 750 mA			25			60			75	mV
		5 mA ≤ I_O ≤ 1A, 0°C ≤ Tj ≤ +125°C				50			120			150	mV
I_Q	Quiescent Current	I_O ≤ 1A	Tj = 25°C			8			8			8	mA
			0°C ≤ Tj ≤ +125°C			8.5			8.5			8.5	mA
ΔI_Q	Quiescent Current Change	5 mA ≤ I_O ≤ 1A				0.5			0.5			0.5	mA
		Tj = 25°C, I_O ≤ 1A				1.0			1.0			1.0	mA
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(7.5 ≤ V_{IN} ≤ 20)			(14.8 ≤ V_{IN} ≤ 27)			(17.9 ≤ V_{IN} ≤ 30)			V
		I_O ≤ 500 mA, 0°C ≤ Tj ≤ +125°C				1.0			1.0			1.0	mA
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(7 ≤ V_{IN} ≤ 25)			(14.5 ≤ V_{IN} ≤ 30)			(17.5 ≤ V_{IN} ≤ 30)			V
V_N	Output Noise Voltage	T_A = 25°C, 10 Hz ≤ f ≤ 100 kHz			40			75			90		µV
$\frac{\Delta V_{IN}}{\Delta V_{OUT}}$	Ripple Rejection	f = 120 Hz	I_O ≤ 1A, Tj = 25°C or	62	80		55	72		54	70		dB
			I_O ≤ 500 mA 0°C ≤ Tj ≤ +125°C	62			55			54			dB
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(8 ≤ V_{IN} ≤ 18)			(15 ≤ V_{IN} ≤ 25)			(18.5 ≤ V_{IN} ≤ 28.5)			V
R_O	Dropout Voltage	Tj = 25°C, I_{OUT} = 1A			2.0			2.0			2.0		V
	Output Resistance	f = 1 kHz			8			18			19		mΩ

Electrical Characteristics LM78XXC (Note 2) (Continued)

0°C ≤ T_J ≤ 125°C unless otherwise noted.

Output Voltage			5V			12V			15V			Units
Input Voltage (unless otherwise noted)			10V			19V			23V			
Symbol	Parameter	Conditions	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
	Short-Circuit Current	Tj = 25°C		2.1			1.5			1.2		A
	Peak Output Current	Tj = 25°C		2.4			2.4			2.4		A
	Average TC of V_{OUT}	0°C ≤ Tj ≤ +125°C, I_O = 5 mA		0.6			1.5			1.8		mV/°C
V_{IN}	Input Voltage Required to Maintain Line Regulation	Tj = 25°C, I_O ≤ 1A		7.3		14.6			17.7			V

Note 1: Thermal resistance of the TO-3 package (K, KC) is typically 4°C/W junction to case and 35°C/W case to ambient. Thermal resistance of the TO-220 package (T) is typically 4°C/W junction to case and 50°C/W case to ambient.

Note 2: All characteristics are measured with capacitor across the input of 0.22 μF, and a capacitor across the output of 0.1μF. All characteristics except noise voltage and ripple rejection ratio are measured using pulse techniques (t_w ≤ 10 ms, duty cycle ≤ 5%). Output voltage changes due to changes in internal temperature must be taken into account separately.

Note 3: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. For guaranteed specifications and the test conditions, see Electrical Characteristics.

REFERENCE

10. REFERENCE

1. Digital Image Processing Using MATLAB by Rafael C. Gonzalaz, Richard E. Wooods, Steven L. Eddins.

2. Digital Image Processing by William. K. Pratt.

3. www.intel.com

4. www.data-sheets.com

5. www.howstuffitworks.com