

**SP04014**



# **HYBRID LINE FOLLOWING AUTOMATED VEHICLE**

**Anshul Rajput**  
**Enrollment no. 041289**

**Aseem Lodha**  
**Enrollment no. 041287**

**Dhruva Vishwanathan**  
**Enrollment no. 041286**

**Yash Gupta**  
**Enrollment no. 041405**



**JAYPEE UNIVERSITY OF  
INFORMATION TECHNOLOGY**

**May 2008**  
**Submitted in partial fulfillment of the Degree of Bachelor of  
Technology**

**DEPARTMENT OF COMPUTER SCIENCE**  
**JAYPEE UNIVERSITY OF INFORMATION**  
**TECHNOLOGY-WAKNAGHAT**

## CERTIFICATE

This is to certify that the work entitled, "*Hybrid Line Following Automated Vehicle*" submitted by Anshul Rajput, Aseem Lodha, Dhruva Vishwanathan, Yash Gupta in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science , Information Technology of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Supervisor:



20.05.2008.

Mr. Ajay Kumar Singh

Senior Lecturer

Department of Computer Science Engineering and Information Technology,

Jaypee University of Information Technology,

Waknaghat, Solan – 173215, Himachal Pradesh,

INDIA.



## ACKNOWLEDGEMENT

We wish to express our earnest gratitude to **Mr. Ajay Kumar Singh**, for providing us invaluable guidance and suggestions, which inspired us to submit this project report on time.

We would also like to thank all the staff members of **Computer Science and Engineering Department of Jaypee University of Information Technology, Wagnaghat**, for providing us all the facilities required for the completion of this project report.

Anshul Rajput  
Enrollment no. 041289  
CSE



Aseem Lodha  
Enrollment no. 041287  
CSE



Dhruva Vishwanathan  
Enrollment no. 041286  
CSE



Yash Gupta  
Enrollment no. 041405  
IT





## **CONTENTS**

## **Page No.**

1. Abstract	12
2. Introduction	14
2.1. What is a line follower?	14
2.2. Why build a line follower?	14
2.3. Prerequisites	14
3. Practical Implementation and Approach	15
3.1. Power Supply	16
3.2. External Clock Setup	20
3.3. Programming AT89c51	26
3.3.a. Language Choice	27
3.3.b. Compiler SDCC	28
3.4. PHASE I I/O port control	30
3.4.a. Time delay	30
3.4.b. Desired Port Configurations	31
3.4.c. C code for LED blink	33
3.4.d. HEX code	34

3.5. PHASE II Motor control	36
3.5.a. Motor Driver theory	36
3.5.b. C code for motor drive	39
3.5.c. HEX code	40
3.6 PHASE III IR Implementation	43
3.6.a. IR Implementation Without Modulation	43
3.6.b. Modulating the sensors	49
3.6.c. Designing the algorithm for signal generation	57
4. Datasheets and details of the ICs used	62
4.1. AT89c51 Microcontroller	63
4.2. L293D Motor Driver	80
4.3. LM324N Comparator IC	92
4.4. 8051	106
4.5. Installation guide for the programmer kit	126
5. Algorithm and C code for Basic Line Follower	138
5.1. Algorithm	139
5.1.a Hybridization through human insight	141
5.1.b. Hybridization through algorithm design	142



5.2. C code	143
6. Assembly of the all modules	148
7. Project Design	156
8. Conclusion and Future improvements to be implemented	161
9. Bibliography	180

## LIST OF FIGURES

Figure no.	Description	Page no.
3.1.1	Power supply	16
3.1.2	Pin out 7805	17
3.2.1	External clock setup	20
3.2.2	Assembly with microcontroller	23
3.5.1	L293D Pin layout	37
3.6.1	LM24N Pin layout	44
3.6.2	IR Assembly	46
4.4.1	8051 Memory	106
4.4.2	Register Banks	109
4.4.3	SFRs	114
5.1	Basic Block Diagrams	138
6.1	DFD Level 1	156
6.2	DFD Level 2	157
6.3	ER Diagram	158
6.4	Pert Chart part I	159
6.5	Pert Chart part II	160
7.1	DTMF Receiver	164
7.2	GPS setup	177



## LIST OF ABBREVIATIONS

mA-	MILLI AMPERE
μF-	MICRO FARAD
I/O-	INPUT OUTPUT
EPROM-	ERASABLE PROGRAMABLE READ ONLY MEMORY
SDCC-	SMALL DEVICE C COMPILER
MCU-	MICROCONTROLLER UNIT
EN-	ENABLE
IN-	INPUT
IR-	INFRA RED
OP-AMPS-	OPERATIONAL AMPILFIER
KHZ-	KILO HERTZ
CMOS-	COMPLEMENTARY METALOXIDE SEMICONDUCTOR
TMOD-	TIME MODE
DTMF-	DUAL TONE MULTI FREQUENCY
GSM-	GLOBAL SYSTEM FOR MOBILE
GPS-	GLOBAL POSITIONING SYSTEM
SMS-	SHORT MESSAGE SERVICE
CLIP-	CALLER LINE IDENTIFICATION PRESENTATION
UMS-	UNIFIED MESSAGING SERVICES
CLIR-	CALLER LINE IDENTIFICATION RESTRICTION
CUG-	CLOSED USER GROUP
BCD-	BINARY CODEDE DECIMAL
DOD-	DEPARTMENT OF DEFENCE

SFR- SPECIAL FUNCTION REGISTERS

PCON- POWER CONTROL

TCON- TIMER CONTROL

DPL- DATA POINT LOW

DPH- DATA POINT HIGH

IE- INTERRUPT ENABLE

IP- INTERRUPT PRIORITY

PSW- PROGRAM STATUS WORD

PC- PROGRAM COUNTER

SP- STACK POINTER



## 1. ABSTRACT

The line follower is a concept which increasingly being used in today's world where software control of the line type (dark or light) is required to make automatic detection possible. They might follow a visual line painted or embedded in the floor or ceiling or an electrical wire in the floor. Most of these robots operated a simple "keep the line in the center sensor" algorithm. They could not circumnavigate obstacles; they just stopped and waited when something blocked their path. Many examples of such vehicles are still sold, by Transbotics, FMC, Egemin, HK Systems and many other companies.

We have innovated with a hybrid design of the basic line follower which would generally perform all the functions of the line follower, and would be able to overcome the short comings of this basic model in case the vehicle is unable to locate the line, by making use of the human controller's knowledge of the grid in which the arbitrary paths are located.

We have also designed the algorithm in such a way that the vehicle would automatically be able to judge the degree of curvature of the line being followed and then accordingly decide the mechanism of turning.

Microcontroller based line followers are applied in the fields like auto drive mode in cars, auto pilot in aircrafts, automatic breaking system in trains, land mine detection, guidance system for industrial robots

moving on shop floor etc. These robots can be used to explore places too dangerous or otherwise impossible for humans to go.

The robots typically sense the line by measuring light reflected off the ground, where a black line reflects little/no light and the white floor reflects a lot of light back.

Starting with an overview of the system the document would cover implementation details like circuit and algorithms, followed by some suggestions on improving the design.

The 'Bibliography' page has a list of relevant books, websites and commonly used parts.



## **2. INTRODUCTION**

### **2.1. *What is a line follower?***

Line follower is a machine that can follow a path. The path can be visible like a black line on a white surface (or vice-versa) or it can be invisible like a magnetic field.

### **2.2. *Why build a line follower?***

Sensing a line and maneuvering the robot to stay on course, while constantly correcting wrong moves using feedback mechanism forms a simple yet effective closed loop system. As a programmer you get an opportunity to 'teach' the robot how to follow the line thus giving it human like property of responding to stimuli.

Practical applications of a line follower : Automated cars running on roads with embedded magnets, guidance system for industrial robots moving on the shop floor etc.

### **2.3. *Prerequisites***

-Knowledge of basic digital and analog electronics.

(A course on Digital Design and Electronic Devices & circuits would be helpful).

-C Programming

-Sheer interest, an innovative brain and perseverance.

### 3. PRACTICAL IMPLEMENTATION AND APPROACH

In our approach the first goal was to program the microcontroller to control the output at the Input/Output ports.

For achieving the above we took up a sub-project for *making a group of 8 LEDs Blink alternatively, with a desired delay.*

This part was implemented on a breadboard.

The basic requirements for this sub-project are:

- A 8 bit microcontroller AT89c51
- 5 volt regulated power supply
- A crystal oscillator 11.0592 Mhz
- 8 LEDs(red)
- Programmer Kit that supports AT89c51
- A compiler for generating the hexadecimal code, from either assembly or C code
- 7805 voltage regulator chip

### 3.1. POWER SUPPLY (5V REGULATED):

We started with implementing a circuit to get a regulated 5 V supply from a standard 9 V battery which we intend to use as the primary power source in our final portable model.

The circuit for the power supply is:

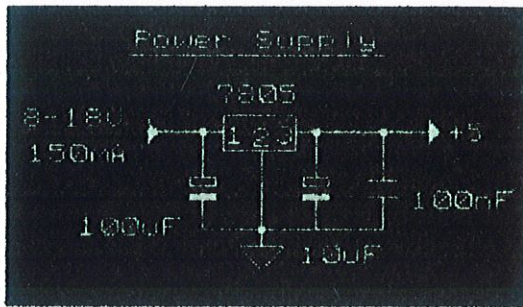


Figure 3.1.1

#### Need for this power supply circuit :

Now we have used the small inexpensive wall transformers with variable output voltage which are available from any electronics shop. Those transformers are easily available, but usually their voltage regulation is very poor, which makes them not very usable for digital circuit unless a better regulation can be achieved in some way. The above circuit is the answer to the problem.

This circuit as shown in figure 3.1.1 can give +5V output at about 150 mA current, but it can be increased to 1 A when good cooling is added to 7805 regulator chip. The circuit has over overload and thermal protection.





*Pin out of the 7805 regulator IC*

Figure 3.1.2

Pin 1: Unregulated voltage input (8-18 V)

Pin 2: Grounded

Pin 3: Regulated voltage output (5 V)

**Components used:**

7805 regulator IC

100  $\mu$ F electrolytic capacitor, at least 25V voltage rating

10  $\mu$ F electrolytic capacitor, at least 6V voltage rating

100 nF ceramic or polyester capacitor

**Modification ideas to get more output current**

If you need more than 150 mA of output current, you can update the output current up to 1A doing the following modifications:

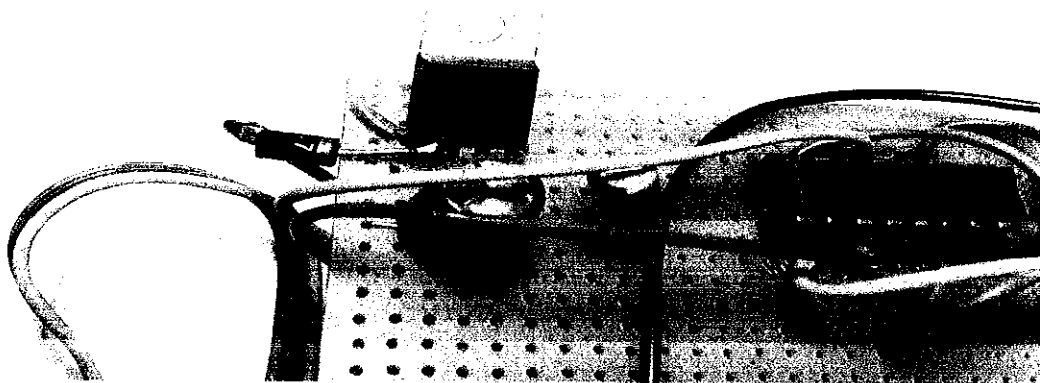
-Change the transformer from where you take the power to the circuit to a model which can give as much current as you need from output

-Put a bigger heat sink to the 7805 regulator (so big that it does not overheat because of the extra losses in the regulator)

If you need other voltages than +5V, you can modify the circuit by replacing the 7805 chips with another regulator with different output voltage from regulator 78xx chip family. The last numbers in the the chip code tells the output voltage. Remember that the input voltage must be at least 3V greater than regulator output voltage or otherwise the regulator does not work well.

Now in the whole purported setup, the PIN 3 is going to be used as a power source

- to voltage supply of microcontroller at PIN 40 (Vcc)
- to PIN 31( EA/Vpp), in order to enable the microcontroller for internal program executions
- to the positive PIN of all the LEDs for voltage supply.
- and to PIN 9, to provide a high on this pin for two machine cycles while the oscillator is running in order to reset the device



*Power supply circuit implemented on the General purpose board*



### 3.2. EXTERNAL CLOCK SETUP

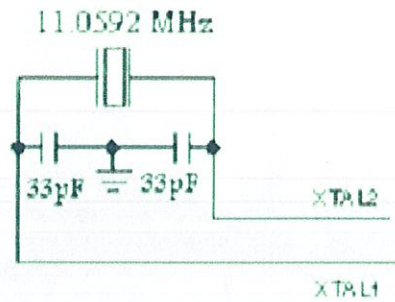


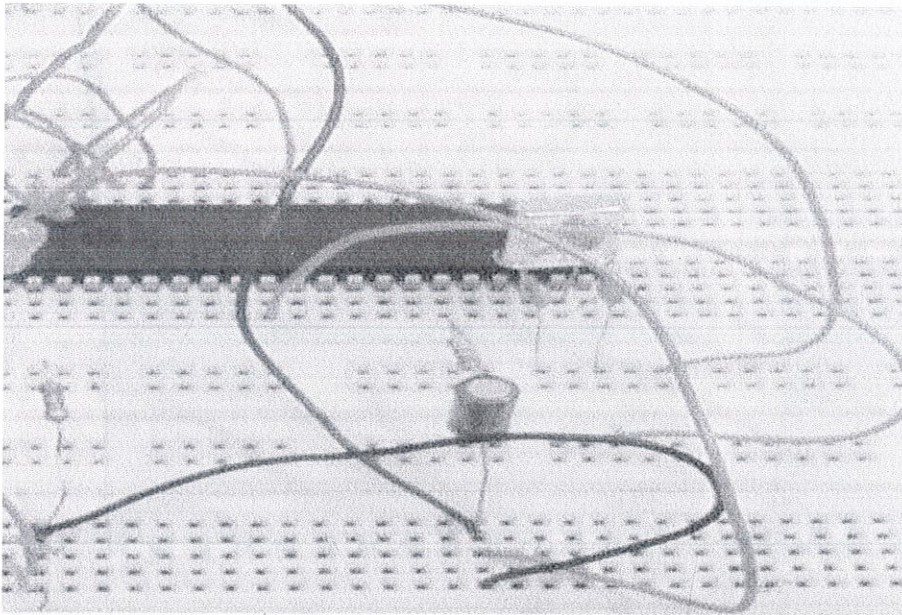
Figure 3.2.1

XTAL1 (PIN 18) and XTAL2 (PIN 19) (as shown in figure 3.2.1) are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator.

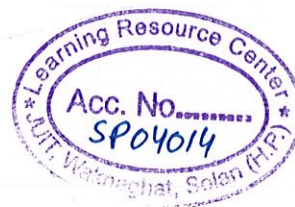
Either a quartz crystal or ceramic resonator may be used.

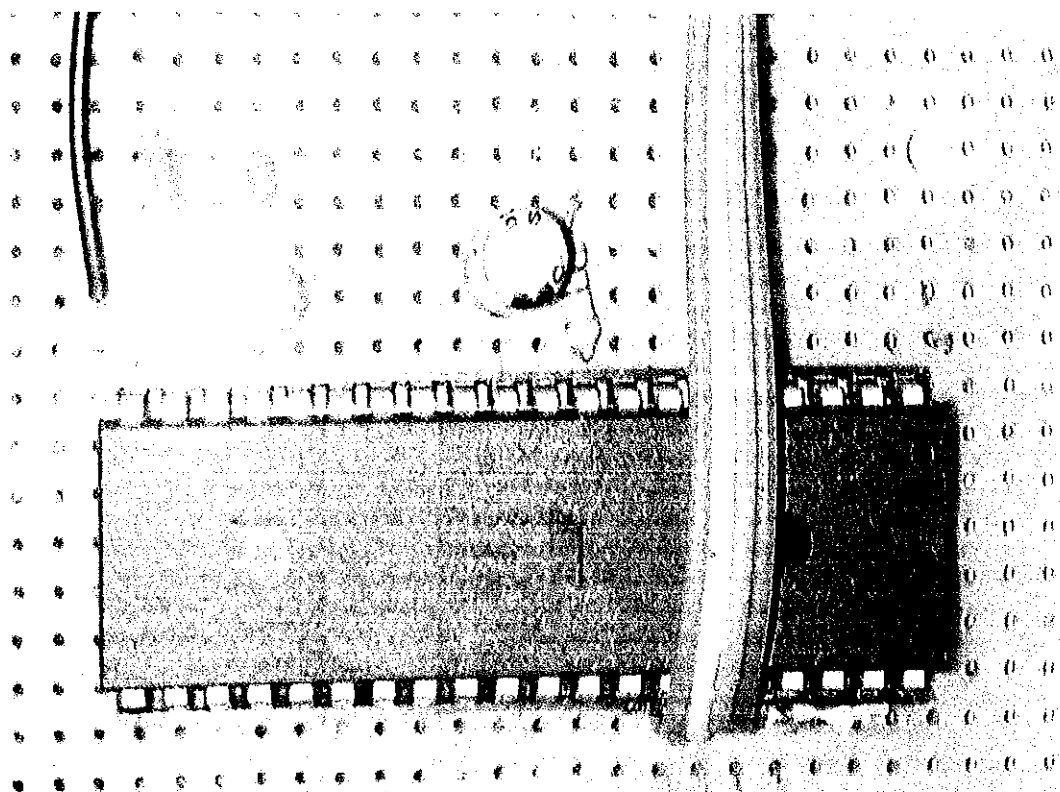
In our setup we used the Crystal oscillator which is more easily available, and more reliable.

The next step was to assemble these power supply and clock circuits with the microcontroller chip as shown in the figure.



**Clock setup on the breadboard using the 11.0592 Mhz crystal oscillator in silver**





*Clock circuit with 11.0592 Mhz crystal oscillator implemented on GP board*



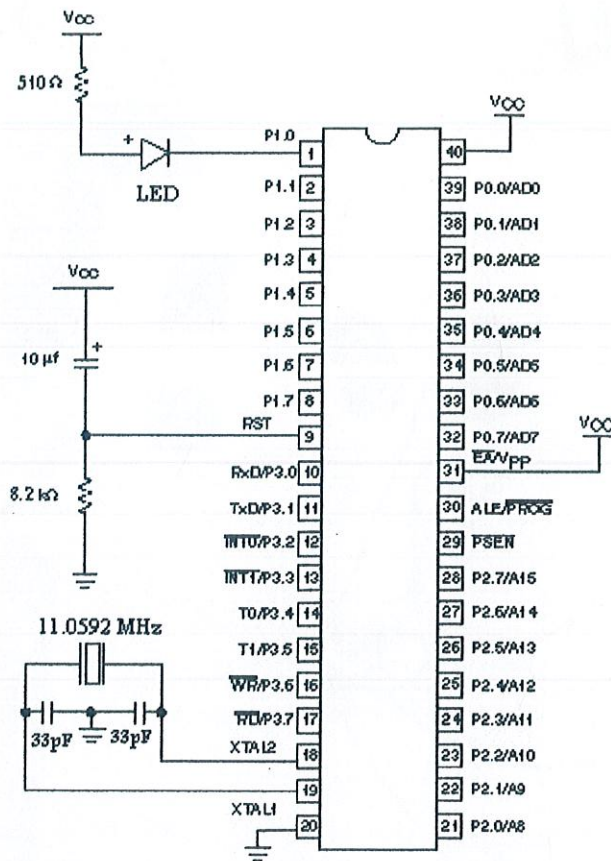
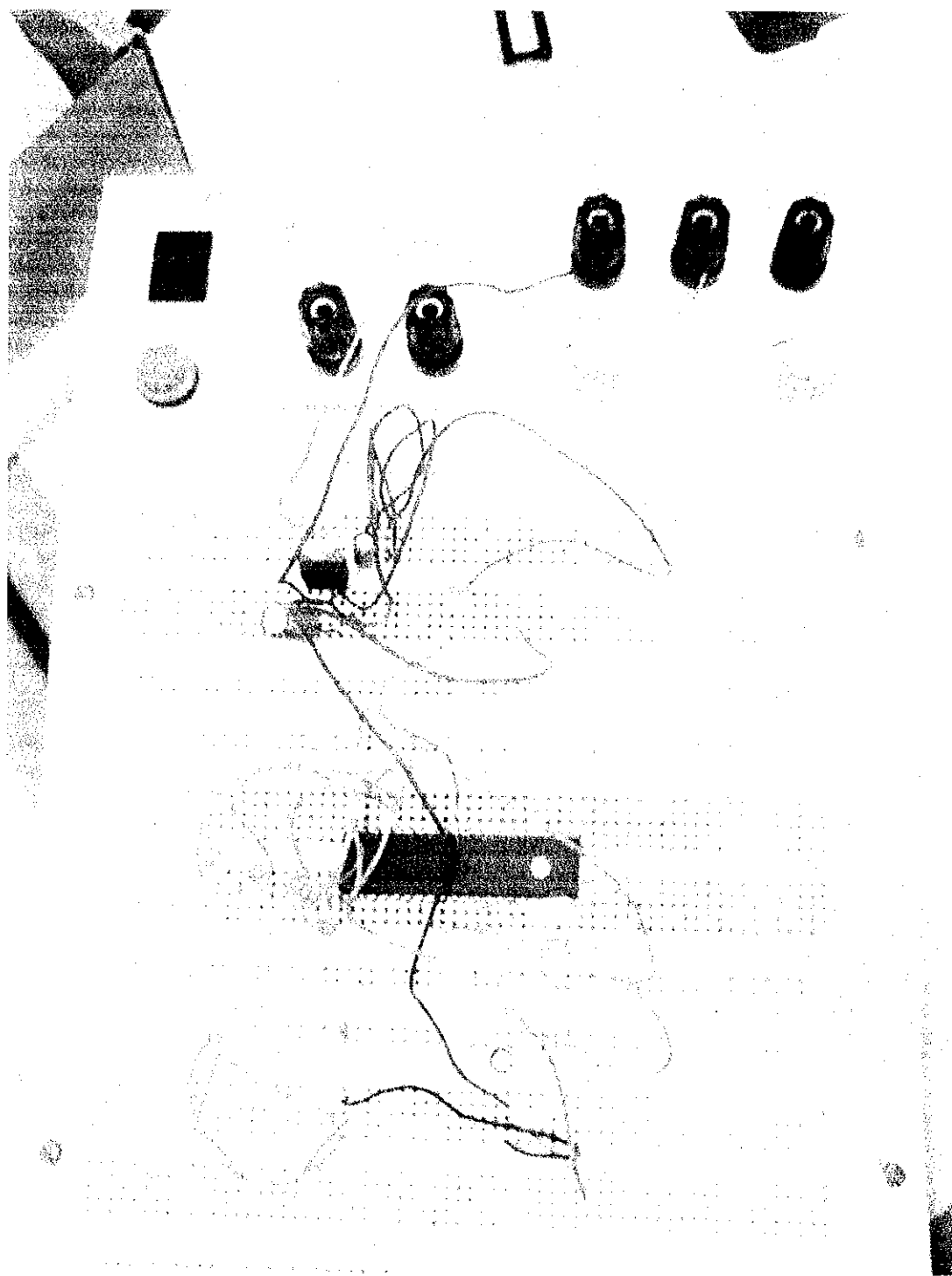


Figure 3.2.2

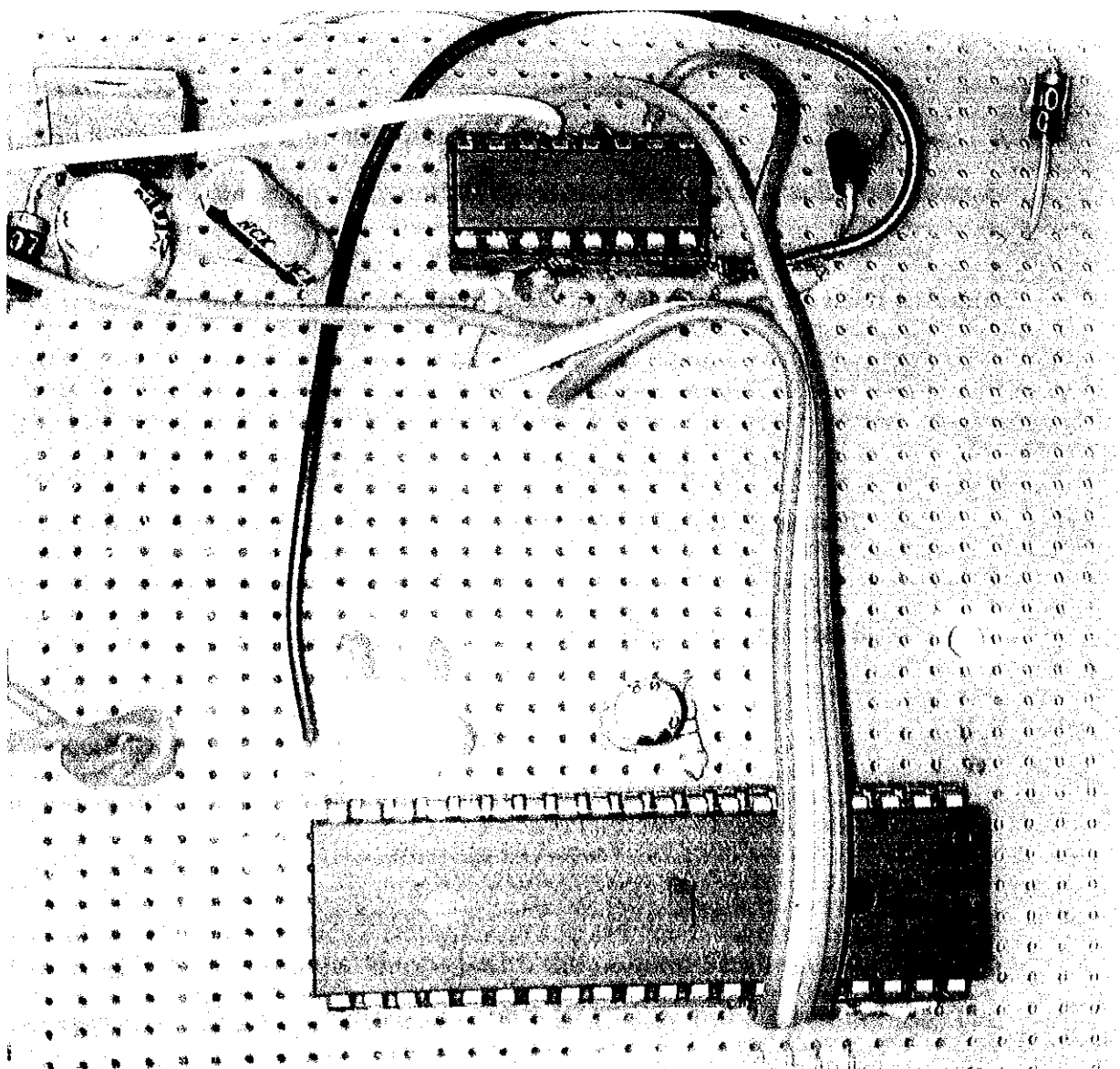
In this setup we have shown only one LED connected to pin 0 of port 1, assume similar connections to all other pins of port 1, making it a total of 8 LEDs connected in succession to port 1 from P1.0 to P1.7.

After making this whole setup on the bread board, the output from port 1 would be illogical since we have not yet programmed the microcontroller with an appropriate code.

So, the next and the most important step is to program the AT89C51 chip with a code that produce high-low voltage pattern in such a way that the 8 LEDs Blink alternatively, with a desired delay.



***Circuit assembly done on breadboard***



*Actual circuit assembly on GP board*



### 3.3. PROGRAMMING AT89C51

#### Port details:

In the 8051 there are total of 4 ports for I/O operations.

32 pins are set aside for four ports P0, P1, P2, P3, where each port takes 8 pins

Upon reset all the ports are configured as inputs, ready to be used as input ports

When first 0 is written it becomes an output port

To configure it as an input port, a 1 must be sent to the port

And to use any of the port as an input port it must be programmed

P0 occupies pins 32-39, used for both I/O

P0 can be used for both address and data, while connecting 8051 to an external memory

P1 occupies pins 1-8, used for both I/O

P2 occupies pins 21-28, used for both I/O

P0 occupies pins 10-17, used for both I/O

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

The AT89C51 code memory array is normally programmed byte-byte with a parallel interface.

### **3.3.A. LANGUAGE CHOICE:**

We are required to feed the microcontroller programmer kit with an appropriate code in HEX to be loaded in the EPROM of AT89c51.

Now, we can't manually write this low level Hex code, so we need a compiler which can convert the algorithm Code in high level language like C or JAVA or assembly language to HEX file.

The size of this HEX file produced by the compiler is one of the main concerns because:

Microcontrollers have limited in system reprogrammable flash memory, in case of AT89c51 its 4K.

We had the choice of using either assembly language or C to write the code.

Assembly language produces a HEX file which is much smaller than C, but the major reason for writing programs in C instead of assembly are:

- It is easier and less time consuming
- It is easier to modify and update the code written in C
- We can make use of the predefined codes in the function libraries
- C code is portable to other microcontrollers with little or no modifications
- Non structural style of assembly language

Since, speed and code size is not our major concern so we chose C for the initial phase of our project.

Since 8051 is an 8-bit microcontroller, the most commonly used data type is unsigned char.

Unsigned char is an 8-bit data type that takes a value in the range of 0-255(00-FFH)

Using INT in place of unsigned char could lead to larger size HEX files.

### **3.3.B. COMPILER**

Our next requirement was a compiler that could efficiently produce the HEX code corresponding to a given C code.

We chose **SDCC** (*Small Device C Compiler*)

SDCC (Small Device C Compiler) is claimed to be an open source, optimizing ANSI-C compiler that designed for 8-bit Microprocessors.

#### **The current target platform includes:**

- MCS51 based Microprocessors (8031, 8032, 8051, 8052 and etc)
- Dallas DS80C390 variants
- Free scale (formerly Motorola) HC08
- Zilog Z80 based MCUs
- Microchip PIC
- Atmel AVR

#### **Some of the features include:**

- ASXXXX and ASLINK, a Freeware, retargettable assembler and linker.

-extensive MCU specific language extensions, allowing effective use of the underlying hardware.

-a host of standard optimizations such as *global sub expression elimination*, *loop optimizations (loop invariant, strength reduction of induction variables and loop reversing )*, *constant folding and propagation*, *copy propagation*, *dead code elimination and jump tables for 'switch' statements*.

-MCU specific optimizations, including a global register allocator.

adaptable MCU specific backend that should be well suited for other 8 bit MCUs independent rule based peep hole optimizer.

-a full range of data types: char (8 bits, 1 byte), short (16 bits, 2 bytes), int (16 bits, 2 bytes), long (32 bit, 4 bytes) and float (4 byte IEEE).

-the ability to add inline assembler code anywhere in a function.

-the ability to report on the complexity of a function to help decide what should be re-written in assembler.

-a good selection of automated regression tests.

SDCC also comes with the *source level debugger SDCDB*, using the current version of Daniel's s51 simulator.

So, it supports AT89c51, i.e. produces a HEX code compatible with it.

*The other option was KEIL compiler which is not open source and we were required to purchase it before using.*



### 3.4. PHASE I

#### Writing the C code for LED blink functionality

First of all, include the appropriate header file for the microcontroller used that includes the library functions.

In case of AT89c51 its "at89x51.h"

Then our approach was to identify the *desired port configuration*.

And after that we have to choose the way to create time delay between the different port configurations.

#### 3.4.A. TIME DELAY

There are two ways to create time delays in 8051 in C:

- Using a simple for loop
- Using the 8051 timers

In both cases, when we write a time delay we must use the oscilloscope to measure the duration of our time delay.

In our microprocessor laboratory we tested with various values to get the standard program for 1ms time delay using a simple for loop.

So, the time delay function below is tested for the crystal oscillator that we used with clock frequency of 11.0592 MHz.

```

void MSdelay(unsigned int time)
{
    unsigned int i, j;
    for(i=0; i<time;i++)
        for(j=0;j<1275;j++);
}

```

here time is in 'ms'

Initially we had a rough idea that to get a 1 ms delay the value of inter for loop counter is between 1000 and 1500.

And we stumbled upon 1275, after calculating the delay value at 1000 and 1500 using oscilloscope and then interpolating the value for 1 ms to be 1275, which we later verified to be correct.

Now using this standard indigenously developed function we can add any time delay multiple of 1 ms.

### **3.4.B. DESIRED PORT CONFIGURATIONS**

In C programming of 8051 unsigned char values (0-255) (indirectly) or hex data values(0x00-0xFF)(directly) are assigned to any port designated by P0, P1, P2, P3

After the device resets the first desired state for port 0 should be such that only first pin is high and all other pins are low.

So initially,

**P0=0x01;**

which indicates the following pin output,

**Pin :    7 6 5 4 3 2 1 0**

**Value: 0 0 0 0 0 0 0 1**

This binary value converted to hex comes out to b 0x01.

Then after a required delay the second pin becomes high and all other pins become low,

**P0=0x02**

Proceeding in similar way, P0 would take the following values between delays:

**P0=0x04**

**P0=0x08**

**P0=0x10**

**P0=0x20**

**P0=0x40**

**P0=0x80**

And then the whole cycle would be repeated indefinitely.

**3.4.c. The final C code for our sub-project with 250 ms delay is as follows:**

```
#include <at89x51.h>

void MSdelay(unsigned int);

void main()
{
    while(1) //repeat forever
    {
        P0=0x01;
        MSdelay(250);
        P0=0x02;
        MSdelay(250);
        P0=0x04;
        MSdelay(250);
        P0=0x08;
        MSdelay(250);
        P0=0x10;
        MSdelay(250);
        P0=0x20;
        MSdelay(250);
        P0=0x40;
        MSdelay(250);
        P0=0x80;
        MSdelay(250);
    }
}
```



```

void MSdelay(unsigned int time)
{
    unsigned int i, j;
    for(i=0; i<time;i++)
        for(j=0;j<1275;j++);
}

```

Now this code would be saved as a file with .c extension, and compiled with SDCC

### **.3.4.D THE FOLLOWING HEX CODE WOULD BE GENERATED**

```

:03000000020008F3
:0300610002000397
:0500030012006480FE04
:0D0064007580019000FA1200AE75800290C8
:0E00710000FA1200AE7580049000FA1200AE84
:0D007F007580089000FA1200AE7580109098
:0E008C0000FA1200AE7580209000FA1200AE4D
:0D009A007580409000FA1200AE75808090D5
:0700A70000FA1200AE80B662
:0800AE00AA82AB837C007D00F7
:0B00B600C3EC9AED9B50147EFB7F040E
:0500C1001EBEFF011F3F
:0B00C600EE4F70F70CBC00E80D80E569
:0100D100220C
:06003700E478FFF6D8FD9D
:080015007900E94400601B7A48

```

:05001D00009000D67800  
:030022000075A0C6  
:0A00250000E493F2A308B8000205FE  
:08002F00A0D9F4DAF275A0FF7C  
:04003D0075A0007832  
:0700410000E84400600779AC  
:0600480000E4F309D8FCFE  
:08004E007800E84400600C7921  
:0B00560000900000E4F0A3D8FCD9FAF1  
:03000800758107F8  
:0A000B001200D2E582600302000338  
:0400D2007582002211  
:00000001FF

The size of the HEX code is 804 bytes.

Now the next step was to load this code on the microcontroller's EPROM using the programmer kit and check the output by making the initial assembly on the bread board with power supply and clock provided.

On the final setup, we got the appropriate output for what we programmed the AT89c51 IC.

*This completed our first sub-project.*

### 3.5. PHASE II

*Our second goal was to control the DC motor with AT89c51.*

#### 3.5.A. MOTOR DRIVER THEORY

For this purpose we took up another sub project, of *programming the microcontroller with an algorithm that would alternatively drive the motor clockwise and anticlockwise.*

**NOTE:** with appropriate combinations of clockwise and anticlockwise motor motion, we would be able to cover all the dimensions on a 2-D plane.

i.e. move our vehicle in left, right, forward or backward direction.

For interfacing the motor with AT89c51 we used L293D IC, which is a quadruple high-current half-H driver.

The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages

from 4.5 V to 36 V.

The datasheet and other details of L293D is given in section 4.

We chose a standard 9V DC motor.

Circuit connections made:

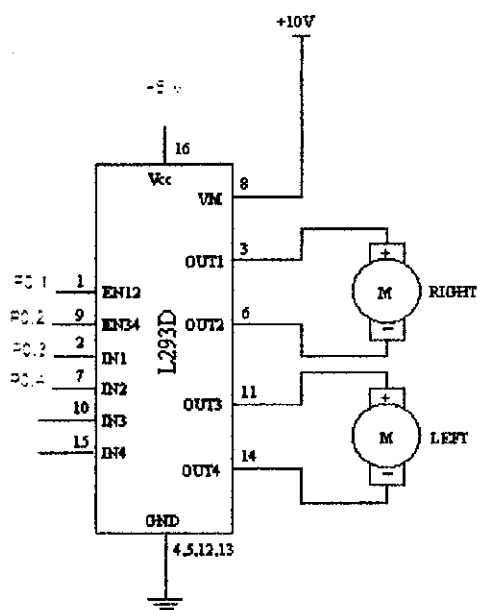


Figure 3.5.1

Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. (as given in the pin layout in figure 3.5.1) When an enable input is high, the associated drivers are enabled and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable motor applications.

We chose driver 1 to control a single motor in our experiment and made the following connection with the AT89c51:



**P0.0 to EN1,2 (pin1)**

**P0.1 to EN3,4 (pin9)**

**P0.2to IN1 (pin2)**

**P0.3 to IN2 (pin7)**

We are required to keep the EN1,2 high in order to keep the output for motor 1 active and in phase with the input, and EN3,4 always low to keep the output for motor 2 off, since we are using only motor 1.

Therefore the state of **P0.0=1** and **P0.1=0** in all cases.

Now the P0.2 and P0.3 connected to IN1 and IN2 respectively will decide the direction of current at OUT1 and OUT2 according to following logic table

EN1,2	IN1	IN2	OUTPUT
<i>H</i>	<i>L</i>	<i>H</i>	<i>clockwise</i>
<i>H</i>	<i>H</i>	<i>L</i>	<i>anticlockwise</i>
<i>H</i>	<i>L</i>	<i>L</i>	<i>Stop</i>
<i>H</i>	<i>H</i>	<i>H</i>	<i>Stop</i>
<i>L</i>	<i>X</i>	<i>X</i>	<i>Stop</i>

Now identifying the desired port configurations:

For clockwise motion:

**Pin:        7 6 5 4 3 2 1 0**

**Value:     0 0 0 0 1 0 0 1**

Therefore for clockwise motion on the motor,

**P0=0x09**

For Anticlockwise motion:

**Pin:        7 6 5 4 3 2 1 0**

**Value:     0 0 0 0 0 1 0 1**

Therefore for Anticlockwise motion on the motor,

**P0=0x05**

So we have to program the microcontroller to alternate between these 2 port configurations with a given delay.

### **3.5.B. C CODE FOR THIS IMPLEMENTATION IS:**

```
#include <at89x51.h>
```

```
void MSdelay(unsigned int);
```

```
void main()
```

```
{  
    while(1) //repeat forever  
    {  
        P0=0x05;  
        MSdelay(3000);  
        P0=0x09;  
        MSdelay(3000);  
    }  
}
```

```

void MSdelay(unsigned int time)
{
    unsigned int i, j;
    for(i=0; i<time;i++)
        for(j=0;j<1275;j++);
}

```

### 3.5.C. AND THE HEX CODE GENERATED AFTER COMPILEATION IS:

```

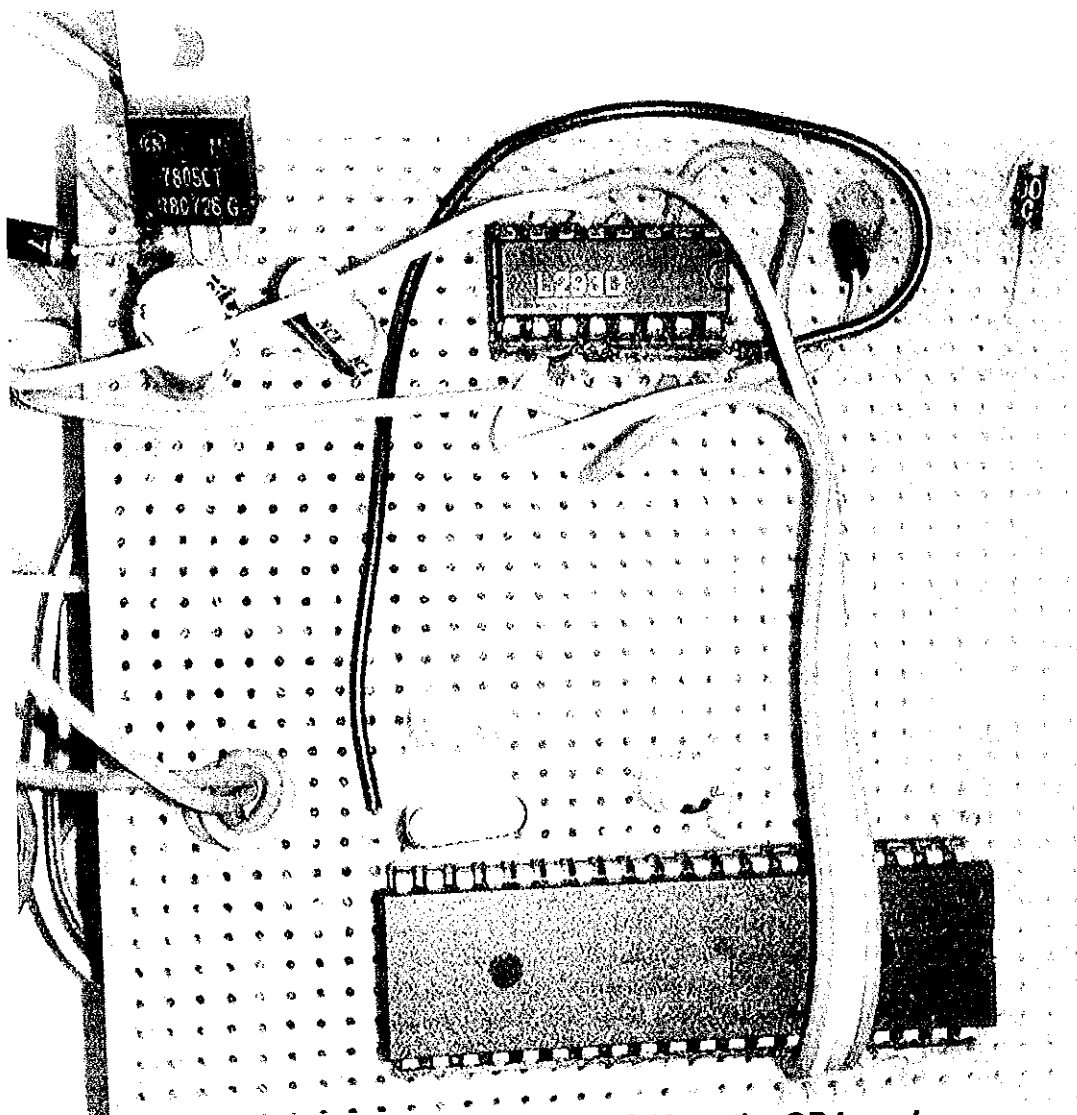
:03000000020008F3
:0300610002000397
:0500030012006480FE04
:0D006400758005900BB8120078758009902A
:070071000BB812007880ECCF
:08007800AA82AB837C007D002D
:0B008000C3EC9AED9B50147EFB7F0444
:05008B001EBEFF011F75
:0B009000EE4F70F70CBC00E80D80E59F
:01009B002242
:06003700E478FFF6D8FD9D
:080015007900E94400601B7A48
:05001D00009000A07836
:030022000075A0C6
:0A00250000E493F2A308B8000205FE
:08002F00A0D9F4DAF275A0FF7C

```

:04003D0075A0007832  
:0700410000E84400600779AC  
:0600480000E4F309D8FCFE  
:08004E007800E84400600C7921  
:0B00560000900000E4F0A3D8FCD9FAF1  
:03000800758107F8  
:0A000B0012009CE58260030200036E  
:04009C007582002247  
:00000001FF

This HEX code was loaded in the flash memory of AT9c51 and the circuit assembly was made as shown in the figure to get the postulated output.

*This completed the phase two of our project.*



*L293D motor driver IC connected to AT89C51 on the GP board*



### **3.6 PHASE III**

In Phase three of the project we were required to implement a simple infra red sensor circuit.

Infra red sensors are one of the most widely used sensors for object detection for small mobile robots. They are easy to implement and low cost making them one of the most perfect sensors.

Infra red as we all know is the wavelength above VIBGYOR's Red which the humans can't see.

#### **3.6.A IR IMPLEMENTATION WITHOUT MODULATION**

Now, we are going to explain the functionality of a single pair of IR transmitter and receiver and in our proposed final model we have made use of eight similar IR sensor modules as described in this phase.

The core of this module is the LM324N IC.

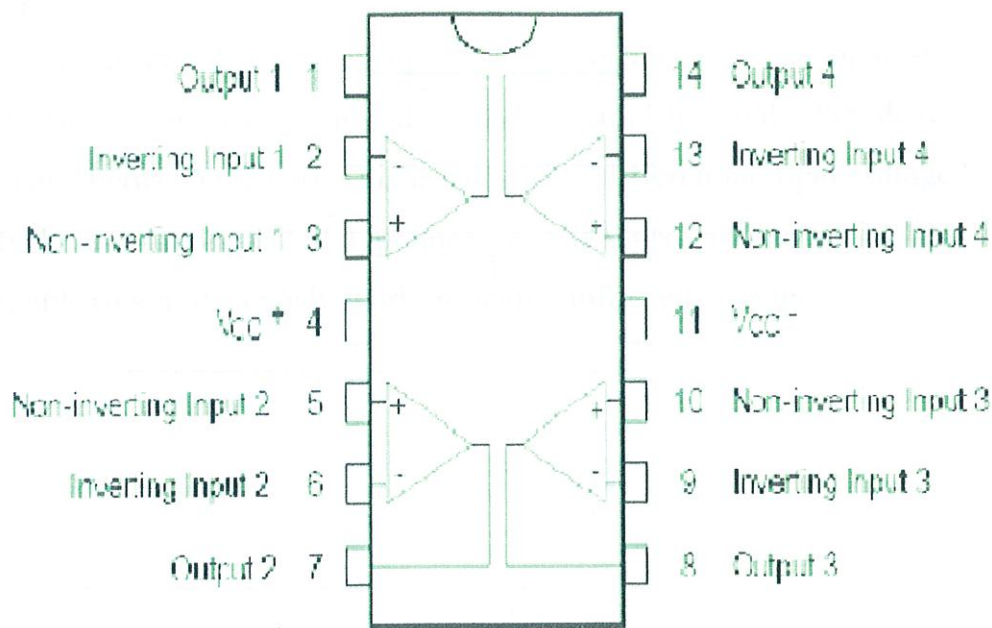


Figure 3.6.1

Pin configuration of LM324N

Here the output pin 1 gives the output in the form of a high or low voltage by comparing the voltage across the 2 corresponding inverting and non inverting input pins and the output is triggered when the difference in the voltage across these input pins crosses a threshold value.

The LM324N consists of four independent, high gain, internally frequency compensated operational amplifiers.

An operational amplifier, usually referred to as an op-amp for brevity, is a DC-coupled high-gain electronic voltage amplifier with differential inputs<sup>[1]</sup> and, usually, a single output. In its typical usage, the output of the op-amp is controlled by negative feedback which largely determines the magnitude of

its output voltage gain, input impedance at one of its input terminals and output impedance.

The amplifier's differential inputs consist of an inverting input and a non-inverting input and ideally the op-amp amplifies only the difference in voltage between the two. This is called the "differential input voltage."

There are 4 independent Op-amps in LM324N, and therefore this IC is capable of simultaneously working with 4 infra-red sensors.

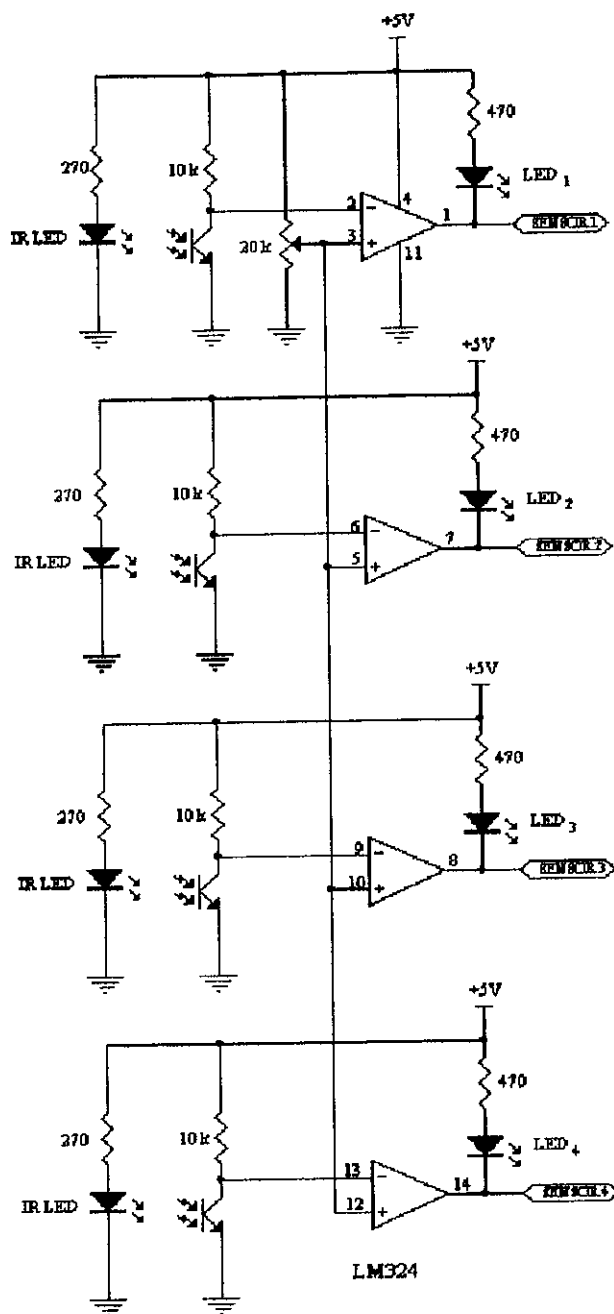
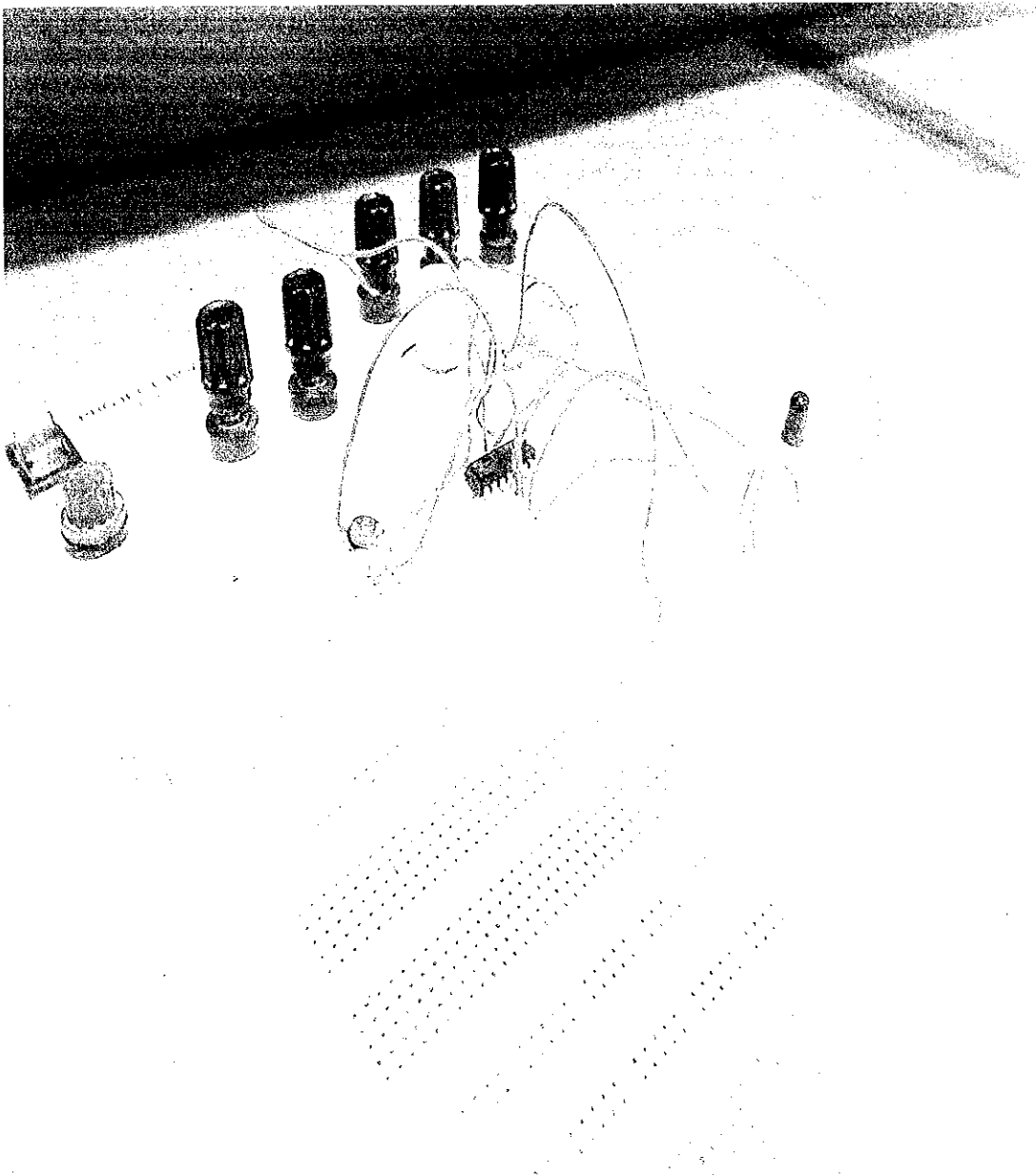


Figure 3.6.2  
Circuit diagram of IR assembly



**IR circuit implementation on circuit board without obstruction**





***IR sensor circuit with obstruction, you can see the indicator red LED in low state***

Here the 20K variable resistance is used to define the threshold value of the voltage at which the output is triggered.

And the LEDs here are used to indicate the status of the output pin, whether the sensor is receiving the IR signal or not.

The binary output we are getting here would be fed to the microcontroller. And after reading this output from all the 6 sensors the microcontroller will get the data indicating the sensor status ( which sensors are detecting the line and which are not)

Based on this sensor data the microcontroller will send appropriate signals to the L293D motor drivers IC to control the motors which in turn would control the direction of the vehicle to get it back o the track.

### **3.6.B MODULATION OF IR SENSORS**

The sensors are not modulated, so their range is restricted to about 5 cm (which is sufficient for following a line).

And without modulation the sensors would also be sensitive to external noise.

Now what I mean by modulation here is that the IR emitter LED will be powered by a 38 KHz square wave.

Or the signal powering the IR emitter will switch on and off 38,000 times in a second.

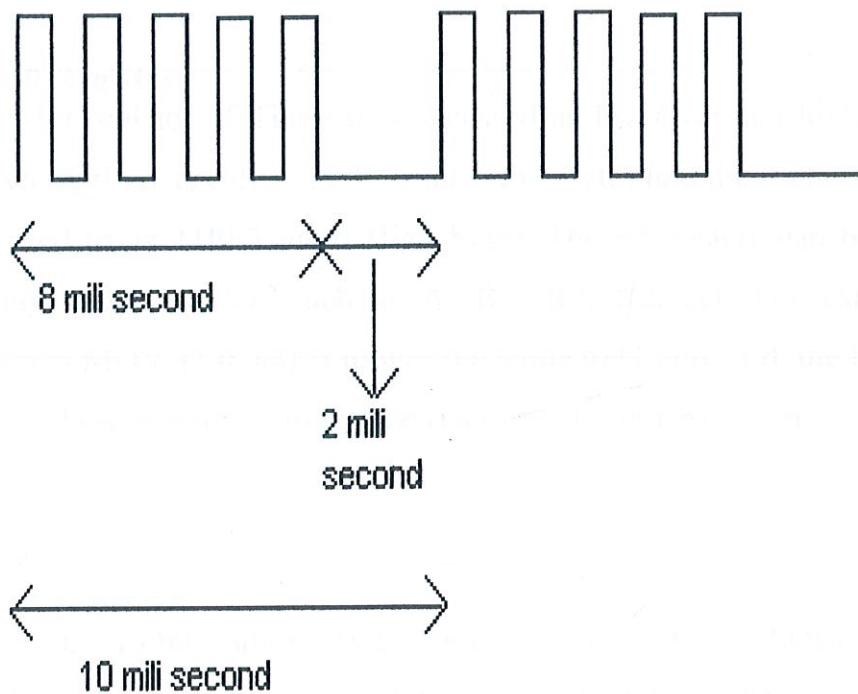
The purpose of this modulation is to increase the range and noise tolerance of the sensors because the sensors that we are using are tuned(while manufacturing) to work most efficiently at 38 Khz with duty cycle of 80%.

Therefore our requirement here is to generate a square wave of 38 KHz with a duty cycle of 80%

Means,

38kHz wave is generated for  $[0.8 \times (1/100)] = 8\text{ms}$  and for 2ms there is no wave i.e. only zero

**here 38 khz wave is suppressed on 100hz wave of 80% duty cycle**



Now a very simple method of generating this signal is by using a separate hardware module, CMOS IC's like 555 and 7404

But this will make the circuit more congested.

Therefore we decided to make use of the timer registers of AT89C51 to generate a time delay corresponding to the time period of this 38 KHz wave.

### **Basic Registers of the timer**

Both Timer 0 and Timer 1 are 16 bits wide. Since the 8051 has an 8-bit architecture, each 16 bit timer is accessed as two separate registers of low bit and high byte. Each timer is discussed separately.

#### **Timer 0 Registers**

The 16-bit register of Timer 0 is accessed as low byte and high byte. The low byte register is called TL0(Timer 0 low byte) and the high byte register is referred to as TH0(Timer0 High byte). These registers can be accessed like any other register, such as A, B, R0, R2, ect. For example, the instruction MOV TL0, #4FH moves the value 4FH into TL0, the low byte of timer 0. These registers can also be read like any other register.

#### **Timer 1 registers**

Timer 1 also 16 bits, and its 16-bit register is split into two bytes, referred to as TL1(Timer 1 low byte) and TH1 (Timer 1 high byte). These registers are accessible in the same way as the registers of timer 0.



### TMOD (timer mode) register

Both timer 0 and 1 use the same register, called TMOD, to set the various timer mode operations. TMOD is an 8-bit register in which the lower 4 bits are set aside for timer 0 and the upper 4 bits are set aside for timer 1.

### M1, M0

M1 and M0 select the timer mode. As shown in the figure there are three modes 0, 1, 2. Mode 0 is a 13-bit timer, mode 1 is a 16-bit timer, and mode 2 is an 8-bit timer.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
-----	-----	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----

D15 TO D8 ARE TH0

D7 TO D0 ARE TL0

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
-----	-----	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----

D15 TO D8 ARE TH1

D7 TO D0 ARE TL1

MSB

LSB

GATE	C/T	M1	M0	GATE	C/T	M1	M0
TIMER1				TIMER0			



**GATE**      Getting control when set. The timer/counter is enabled only while the INTx pin is high and the TRx control pin is set.

**C/T**        Timer of the counter selected is cleared for the timer operation(Input from the internal system clock).

**M1**        Mode bit 1

**M0**        Mode bit 0

**C/T(clock/timer)**

This bit in the TMOD register is used to decide whether the timer is used as an delay generator or an event counter. If C/T=0, it is used as a timer for delay generation.

**Example:**

Find the values of the TMOD to operate ion the following modes.

- Model Timer1
- Mode2 Timer2
- Mode0 Timer1

**Solution:**

1. TMOD is 00010000=01H

The gate control bit and C/T bit are made 0, and the unused timer(Timer0 bit is also 0).

2. TMOD is 01010010=52H
3. TMOD is 00000000H=0H

### **Clock Source for Timer**

As you know every timer needs a clock pulse to tick. If C/T=0, the crystal frequency is attached to the 8051 is the source for the timer of the clock. This means that the size of the crystal frequency attached to the 8051 is the source for the clock timer. The frequency of the timer is always  $\frac{1}{2}$  of the frequency of the clock attached to the 8051.

Although various 8051-based systems have XTAL frequency of 10 MHz, we will concentrate on the XTAL frequency of 11.0592 MHz.

### **GATE**

The other bit of the TMOD register is the GATE bit. The timers of the 8051 have both. The start and stop of the timers are controlled by way of software by the TR(Timer Start) bit TR0 and TR1. This is achieved by the instructions SETB TR1 and CLR TR1 for timer1 and SETB TR0 and CLR TR0 for timer0. These instructions start and stop the the timers as long as GATE=0 in the TMOD register. The hardware of start and stop is achieved by making GATE=1 in the TMOD register.

### **Example:**

How are the Timers0 and 1 atrted and stopped by instructions?

### **Solution:**

The timers are started by using the instructions to set the timer start bits TR0 and TR1

Which are called timer run control bits. They can be cleared by clearing those bits. When a timer counts to its maximum value, it sets a TF0 or TF1. At this point it is necessary to know more about the bits TF and TR for timers 0 and 1. While TMOD controls the timer modes, another register controls the timer control operations called TCON.

MSB				LSB			
TF1	TR1	TF0	TR0	IE	IT1	IE0	IT0

BIT	SYMBOL	FUNCTION
TCON.7	TF1	Timer1 overflow flag
TCON.6	TR1	Timer1 run control bit
TCON.5	TF0	Timer0 overflow flag
TCON.4	TR0	Timer0 run control bit

### Mode 1 programming

The following are the characteristics and operations of mode1

1. It is a 16-bit timer; therefore, it allows values of 0000 to FFFFH to be loaded in the timer's registers TL and TH.
2. After TH and TL are loaded with a 16 bit initial value, the timer must be started. This is done by SETB TR0 for Timer0 and SETB TR1 for Timer1.

3. After timer is started, it starts to count up. It counts up till it reaches its levels FFFFH. When it rolls over from FFFFH to 0000, it sets high a flag bit called TF(timer flag). This timer flag can be monitored.
4. After the timer reaches its limit and rolls over, in order to repeat the process the registers TH and TL must be reloaded with the original value, and TF must be reset to 0.

### **Steps to program in mode 1**

To generate a time delay, using the timer's mode 1, the following steps are taken.

1. Load the TMOD value register indicating which timer(timer0 or1) is to be used and which timer mode is to be selected (0or1) is selected.
2. Load registers TL and TH with initial count values.
3. Start the timer.
4. Keep monitoring the timer flag(TF) with the JNB TFx target instruction to see if it is raised. Get out of the loop, when TF becomes high.
5. Stop the timer.
6. Clear the TF flag for the next round.
7. Go back to step 2 to load TH and TL again.

### 3.6.C. DESIGNING ALGORITHM FOR GENERATING THIS REQUIRED SIGNAL

The frequency of the microcontroller timer,

$F_t = 1/12 \times \text{frequency of crystal oscillator}$

$XTAL = 11.0592 \text{ Mhz}$

Therefore  $F_t = 1/12 \times 11.0592$

$F_t = 921.6 \text{ KHz}$

Corresponding time period,

$T_t = 1.085 \text{ us}$

**This means that the clock ticks every 1.085 us**

Required frequency,

**$f = 38 \text{ KHz}$**

Corresponding time period,

$T = 1/f$

$T = 1/38K$

**$T = 26.3 \text{ us}$**

This is the time period for one complete cycle i.e. one high period and one low period.

Therefore the desired time delay is  $T/2$

**$T/2 = 13.15 \text{ us}$**

If we divide this desired delay by 1.085 us, we will get the no. of clocks required to attain this time delay

Therefore,  $13.15/1.085 = 12.119815$

**$= 12 \text{ clocks}$**

now, calculating the value of low byte of the timer to generate the desired number of clocks

FFFFH=65536 in decimal

$65536 - 12 = 65524$

$65524 = \text{FFF4H}$

**Therefore the value of TL to generate 12 clocks is F4H**

## WRITING THE FUNCTION FOR GENERATING THIS TIME DELAY BY 12 CLOCKS

```
void delay38k()
{
    TMOD=0x01; //using timer 1 in mode 1
    TL0=0xF4;
    TH0=0xFF;
    TR0=1; //to start the timer
    while(TF0==0); //check for flag which will be set when TH reached
    TR0=0; //stop the timer
    TF0=0; // reset the flag
}
```

## USING THIS TIME DELAY TO GENERATE THE DESIRED 38 KHZ SIGNAL

```
#include<reg51.h>
void delay38k(void);
```

```
void main(void)
{
    while(1)
    {
        P0.0=1;
        delay38k();
    }
}
```



```

        P0.0=0;
        delay38k();
    }
}

```

```

void delay38k()
{
    TMOD=0x01;
    TL0=0xF4;
    TH0=0xFF;
    TR0=1;
    while(TF0==0);
    TR0=0;
    TF0=0;
}

```

### **ADDING A DUTY CYCLE OF 80%**

```

#include<reg51.h>
void delay38k(void);
void delay(unsigned int i)
{
    while(i!=0)
    {
        i--;
    }
}

```

```
void delay38k()
```

```
{  
    TMOD=0x01;  
    TL0=0xF4;  
    TH0=0xFF;  
    TR0=1;  
    while(TF0==0);  
    TR0=0;  
    TF0=0;  
}
```

```
void main(void)
```

```
{  
    for (q=52,q>=0,q--) //52 is used for maintaining 8 mili sec time  
    {  
        P0.0=0;  
        timer38khz();  
        P0.0=1;  
        timer38khz();  
    }
```

```
    delay(0x0D); //0D hex no. is used for maintaining 2 mili sec time  
}
```

Now by using the configured pin to power the IR emitter, we managed to increase the range and the sensitivity of the sensors without using any additional resources.

# AT89C51

## Features

- Compatible with 8051
- 4K Bytes of On-Chip Flash
- 128 x 8-bit internal RAM
- Fully Static Operation
- Three-level Program Memory Lock
- 128 x 8-bit internal RAM
- 32 Programmable I/O Lines
- Two 16-bit Timers/Counters
- Six Interrupt Sources and Five Priority Levels
- Programmable Parallel Slave Port
- Low-power Idle and Sleep Modes

## Description

The AT89C51 is a high-performance, low-power, 8-bit CMOS microcontroller. It is manufactured using a 1.5µm CMOS technology, which provides high performance and low power consumption. The device is compatible with the 8051 microcontroller architecture. It features 4K Bytes of On-Chip Flash memory, 128 x 8-bit internal RAM, and 32 Programmable I/O Lines. The AT89C51 also includes two 16-bit Timers/Counters, six Interrupt Sources and five Priority Levels, a Programmable Parallel Slave Port, and Low-power Idle and Sleep Modes.

## Pin Configuration



## 4. Datasheets and details of the ICs used

8-bit  
Microcontroller  
with 4K Bytes

For New Design  
Use AT89C51



## 4.1 AT89C51 MICROCONTROLLER

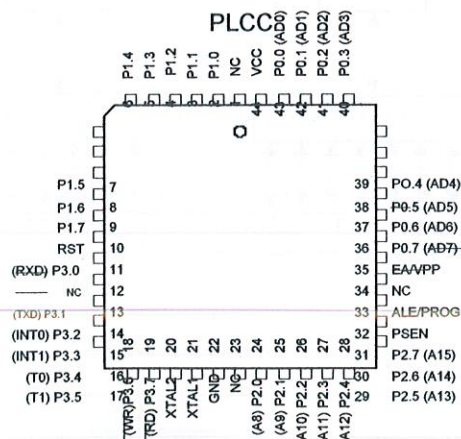
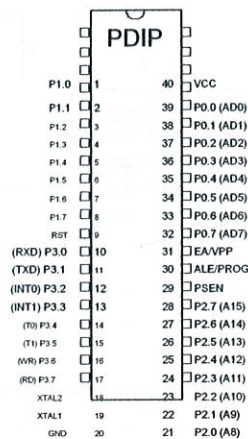
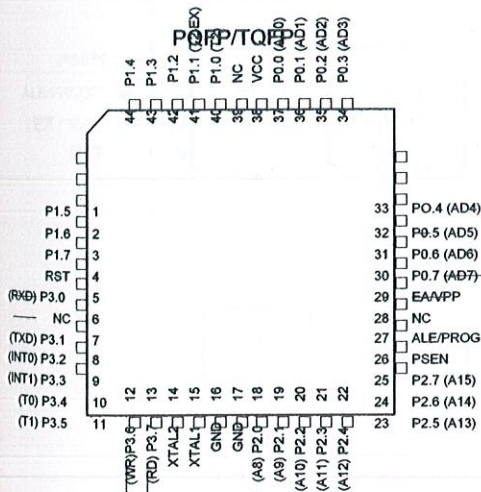
### Features

- 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

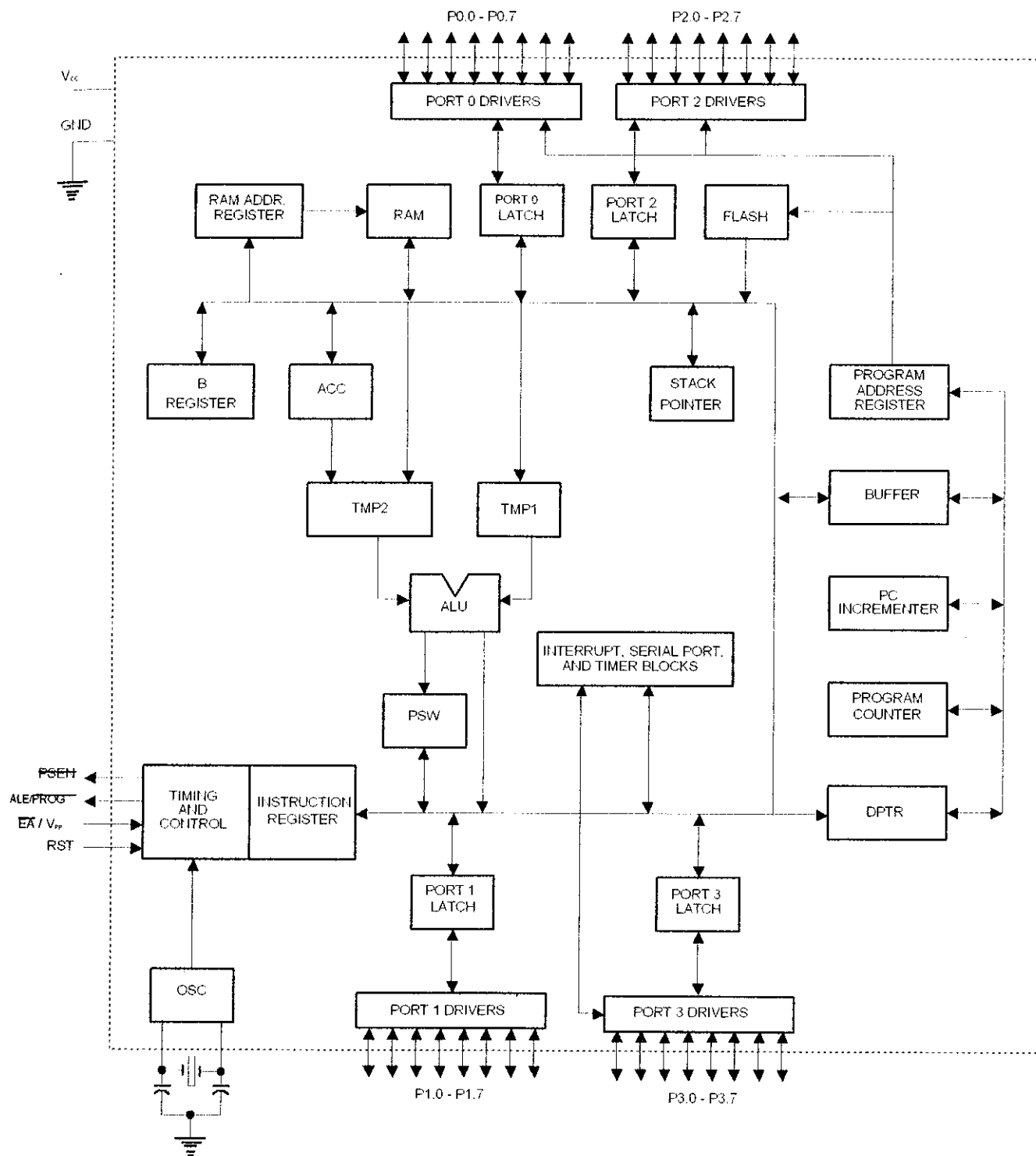
**Not Recommended**

**for New Designs.  
Use AT89S51.**

Rev. 0265G-02/00



### Block Diagram



# AT89C51



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 @ RI), Port 2 emits the contents of the P2 Special Function Register.

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

### Port 3

Port 3 is an 8-bit bi-directional I/O port with internal pullups.

The Port 3 output buffers can 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	INT0 (external interrupt 0)
P3.3	INTT (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

Port 3 also receives some control signals for Flash programming and verification.

### RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

### ALE/PROG

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

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE





pulse is skipped during each access to external Data Memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

### **PSEN**

Program Store Enable is the read strobe to external program memory.

When the AT89C51 is executing code from external program memory,  $\overline{\text{PSEN}}$  is activated twice each machine cycle, except that two  $\overline{\text{PSEN}}$  activations are skipped during each access to external data memory.

### **$\overline{\text{EA}}$ /VPP**

External Access Enable.  $\overline{\text{EA}}$  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,  $\overline{\text{EA}}$  will be internally latched on reset.

$\overline{\text{EA}}$  should be strapped to  $V_{cc}$  for internal program executions.

This pin also receives the 12-volt programming enable voltage ( $V_{PP}$ ) during Flash programming, for parts that require 12-volt  $V_{PP}$ .

### **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

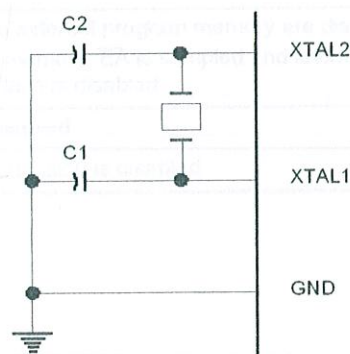
unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

## **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



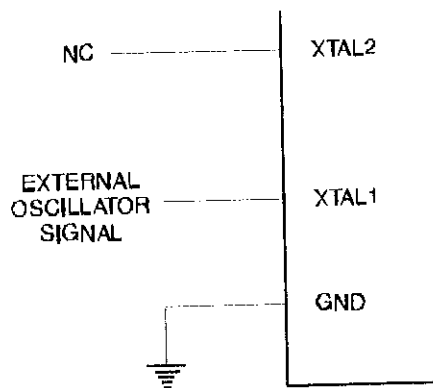
Note: C1, C2 = 30 pF  $\pm$  10 pF for Crystals  
= 40 pF  $\pm$  10 pF for Ceramic Resonators

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

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



Figure 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 EA pin<sup>1</sup> is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of EA be in agreement with the current logic level at that pin in order for the device to function properly.

### Lock Bit Protection Modes

Program Lock Bits				Protection Type
	LB1	LB2	LB3	
1	U	U	U	No program lock features
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, 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



## Programming the Flash

The AT89C51 is normally shipped with the on-chip Flash memory array in the erased state (that is, contents = FFH) and ready to be programmed. The programming interface accepts either a high-voltage (12-volt) or a low-voltage

(V<sub>CC</sub>) program enable signal. The low-voltage programming mode provides a convenient way to program the AT89C51 inside the user's system, while the high-voltage programming mode is compatible with conventional third-party Flash or EPROM programmers.

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

	V <sub>PP</sub> = 12V	V <sub>PP</sub> = 5V
Top-side Mark	AT89C51 xxxx yyww	AT89C51 xxxx-5 yyww
Signature	(030H) = 1EH (031H) = 51H (032H) = FFH	(030H) = 1EH (031H) = 51H (032H) = 05H

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

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

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

and data for the entire array or until the end of the object file is reached.

**Data Polling:** The AT89C51 features Data Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written datum on P0.7. Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. Data Polling may begin any time after a write cycle has been initiated.

**Ready/Busy:** The progress of byte programming can also be monitored by the RDY/ $\overline{BSY}$  output signal. P3.4 is pulled low after ALE goes high during programming to indicate BUSY. P3.4 is pulled high again when programming is done to indicate READY.

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

**Chip Erase:** The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/ $\overline{PROM}$  low for 10 ms. The code array is written with all "1"s. The chip erase operation must be executed before the code memory can be re-programmed.

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






- (030H) = 1EH indicates manufactured by Atmel
- (031H) = 51H indicates 89C51
- (032H) = FFH indicates 12V programming
- (032H) = 05H indicates 5V programming

## Programming Interface

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

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

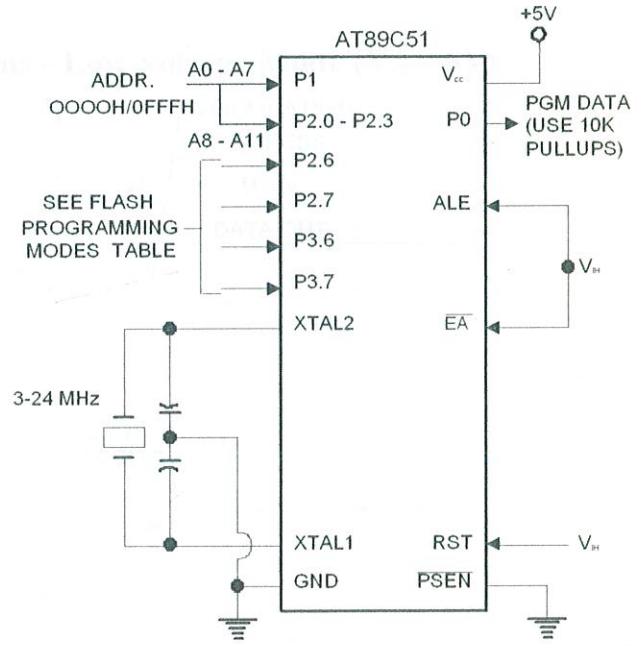
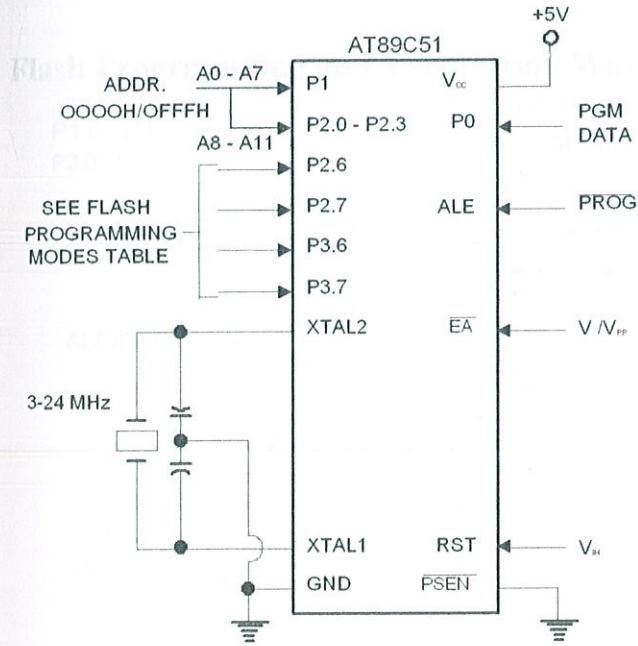
# Flash Programming Modes

Mode		RST	PSEN	ALE/PROG	EA/V <sub>PP</sub> P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L		H/12V	L	H	H
Read Code Data		H	L	H	H	L	H	H
Write Lock	Bit - 1	H	L		H/12V	H	H	H
	Bit - 2	H	L		H/12V	H	H	L
	Bit - 3	H	L		H/12V	H	L	H
Chip Erase		H	L	 (1)	H/12V	H	L	L
Read Signature Byte		H	L	H	H	L	L	L

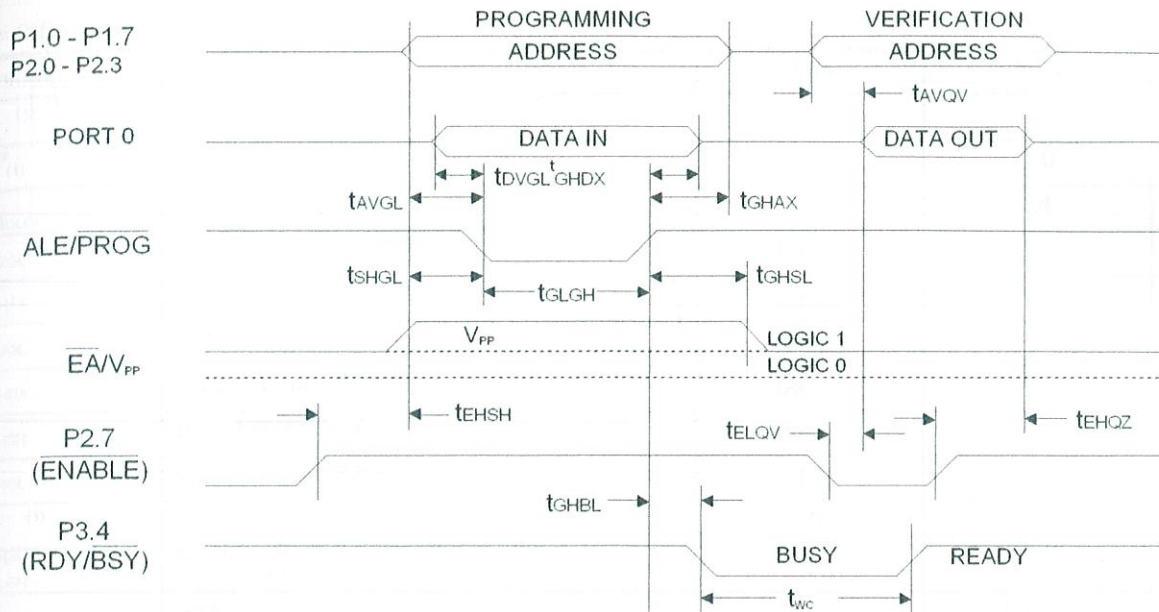
Note: 1. Chip Erase requires a 10 ms PROG pulse.

Figure 3. Programming the Flash

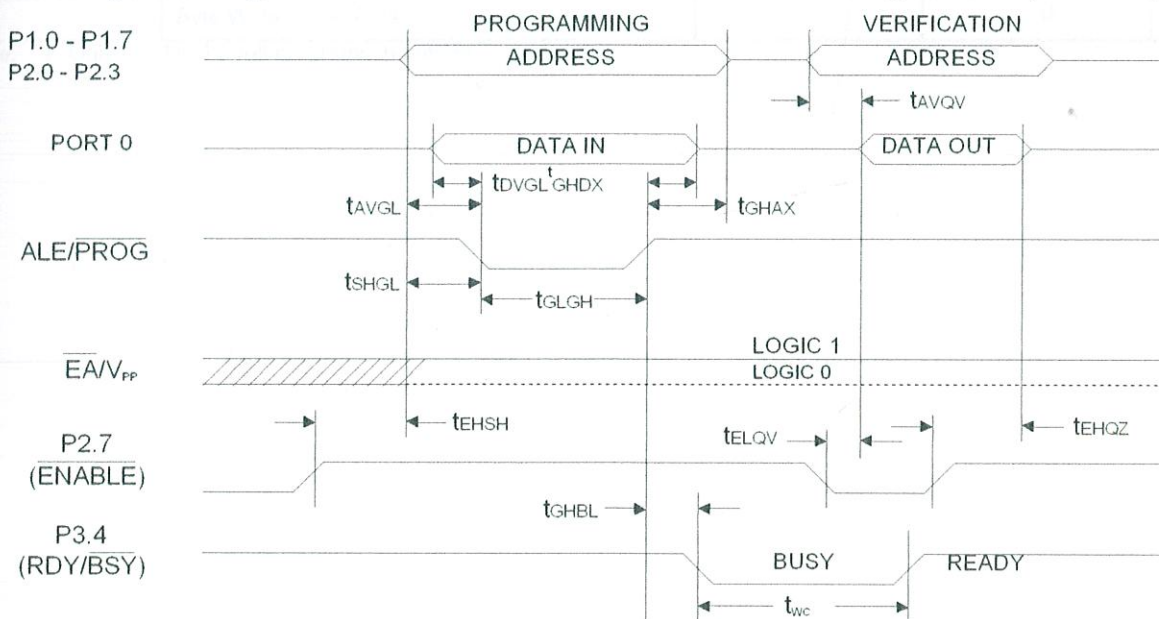
Figure 4. Verifying the Flash



## Flash Programming and Verification Waveforms - High-voltage Mode ( $V_{PP} = 12V$ )



## Flash Programming and Verification Waveforms - Low-voltage Mode ( $V_{PP} = 5V$ )





## Flash Programming and Verification Characteristics

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

Symbol	Parameter	Min	Max	Units
(1)	Programming Enable Voltage	11.5	12.5	V
$V_{PP}$ (1)	Programming Enable Current		1.0	mA
$f_{PP}$ $1/t_{CLCL}$	Oscillator Frequency	3	24	MHz
$t_{AVGL}$	Address Setup to <b>PROG</b> Low	48t <sub>CLCL</sub>		
$t_{GHAX}$	Address Hold after <b>PROG</b>	48t <sub>CLCL</sub>		
$t_{DVGL}$	Data Setup to <b>PROG</b> Low	48t <sub>CLCL</sub>		
$t_{GHDX}$	Data Hold after <b>PROG</b>	48t <sub>CLCL</sub>		
$t_{ESHM}$	P2.7 ( <b>ENABLE</b> ) High to $V_{PP}$	48t <sub>CLCL</sub>		
$t_{SHGL}$	$V_{PP}$ Setup to <b>PROG</b> Low	10		$\mu\text{s}$
(1)		10		$\mu\text{s}$
$t_{GHSL}$ $t_{GLGH}$	$V_{PP}$ Hold after <b>PROG</b> <b>PROG</b> Width	1	110	$\mu\text{s}$
$t_{AVOV}$	Address to Data Valid		48t <sub>CLCL</sub>	
$t_{ELQV}$	<b>ENABLE</b> Low to Data Valid		48t <sub>CLCL</sub>	
$t_{EHQZ}$	Data Float after <b>ENABLE</b>	0	48t <sub>CLCL</sub>	
$t_{GHBL}$	<b>PROG</b> High to <b>BUSY</b> Low		1.0	$\mu\text{s}$
$t_{wc}$	Byte Write Cycle Time		2.0	ms

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



## Absolute Maximum Ratings\*

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

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

## DC Characteristics

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

Symbol	Parameter	Condition	Min	Max	Units
$V_L$	Input Low-voltage	(Except $\overline{EA}$ )	-0.5	$0.2 V_{CC} - 0.1$	V
$V_{L1}$	Input Low-voltage ( $\overline{EA}$ )		-0.5	$0.2 V_{CC} - 0.3$	V
$V_H$	Input High-voltage	(Except XTAL1, RST)	$0.2 V_{CC} + 0.9$	$V_{CC} + 0.5$	V
$V_{H1}$	Input High-voltage	(XTAL1, RST)	$0.7 V_{CC}$	$V_{CC} + 0.5$	V
$V_{OL}$	Output Low-voltage <sup>(1)</sup> (Ports 1,2,3)	$I_{OL} = 1.6 \text{ mA}$		0.45	V
$V_{OL1}$	Output Low-voltage <sup>(1)</sup> (Port 0, ALE, $\overline{PSEN}$ )	$I_{OL} = 3.2 \text{ mA}$		0.45	V
$V_{OH}$	Output High-voltage (Ports 1,2,3, ALE, $\overline{PSEN}$ )	$I_{OH} = -60 \mu\text{A}$ , $V_{CC} = 5\text{V} \pm 10\%$	2.4		V
		$I_{OH} = -25 \mu\text{A}$	$0.75 V_{CC}$		V
		$I_{OH} = -10 \mu\text{A}$	$0.9 V_{CC}$		V
$V_{OH1}$	Output High-voltage (Port 0 in External Bus Mode)	$I_{OH} = -800 \mu\text{A}$ , $V_{CC} = 5\text{V} \pm 10\%$	2.4		V
		$I_{OH} = -300 \mu\text{A}$	$0.75 V_{CC}$		V
		$I_{OH} = -80 \mu\text{A}$	$0.9 V_{CC}$		V
$I_L$	Logical 0 Input Current (Ports 1,2,3)	$V_H = 0.45\text{V}$		-50	$\mu\text{A}$
$I_{L1}$	Logical 1 to 0 Transition Current (Ports 1,2,3)	$V_H = 2\text{V}$ , $V_{CC} = 5\text{V} \pm 10\%$		-650	$\mu\text{A}$
$I_{L2}$	Input Leakage Current (Port 0, $\overline{EA}$ )	$0.45 < V_H < V_{CC}$		$\pm 10$	$\mu\text{A}$
RRST	Reset Pull-down Resistor		50	300	k $\Omega$
$C_{in}$	Pin Capacitance	Test Freq. = 1 MHz, $T_A = 25^\circ\text{C}$		10	pF
$I_{CC}$	Power Supply Current	Active Mode, 12 MHz		20	mA
		Idle Mode, 12 MHz		5	mA
	Power-down Mode <sup>(2)</sup>	$V_{CC} = 6\text{V}$		100	$\mu\text{A}$
		$V_{CC} = 3\text{V}$		40	$\mu\text{A}$

Notes: 1. Under steady state (non-transient) conditions,  $I_{OL}$  must be externally limited as follows:

Maximum  $I_{OL}$  per port pin: 10 mA

Maximum  $I_{OL}$  per 8-bit port: Port 0: 26 mA

Ports 1, 2, 3: 15 mA

Maximum total  $I_{OL}$  for all output pins: 71 mA

If  $I_{OL}$  exceeds the test condition,  $V_{OL}$  may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

2. Minimum  $V_{CC}$  for Power-down is 2V.

# AT89C51



## AC Characteristics

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

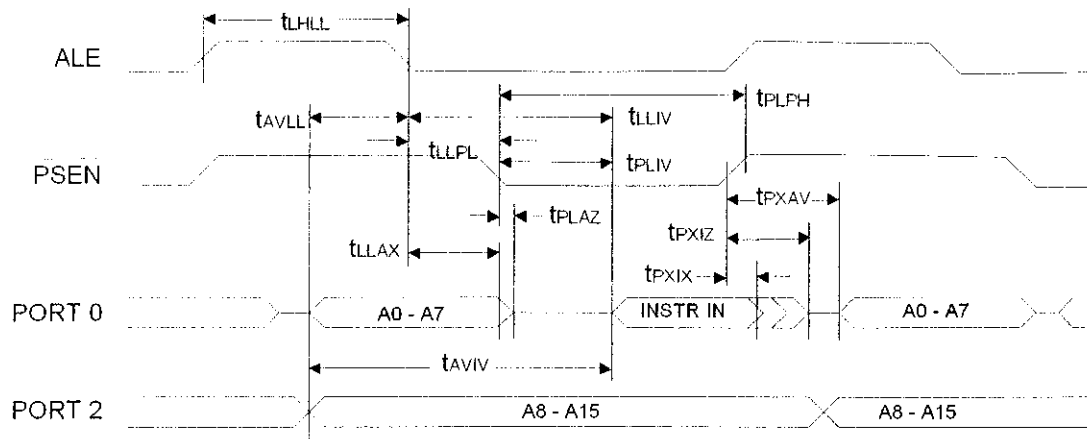
## External Program and Data Memory Characteristics

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

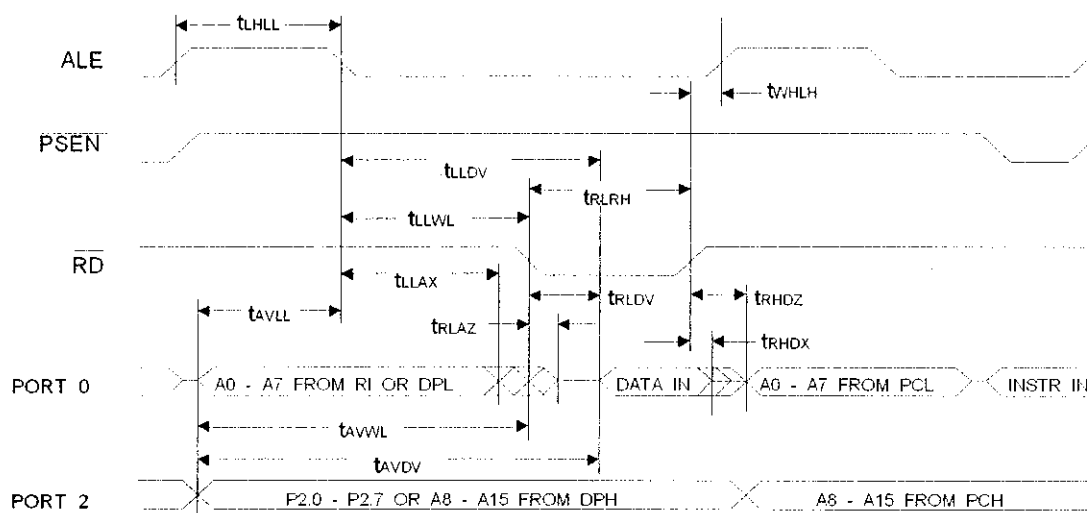




## External Program Memory Read Cycle

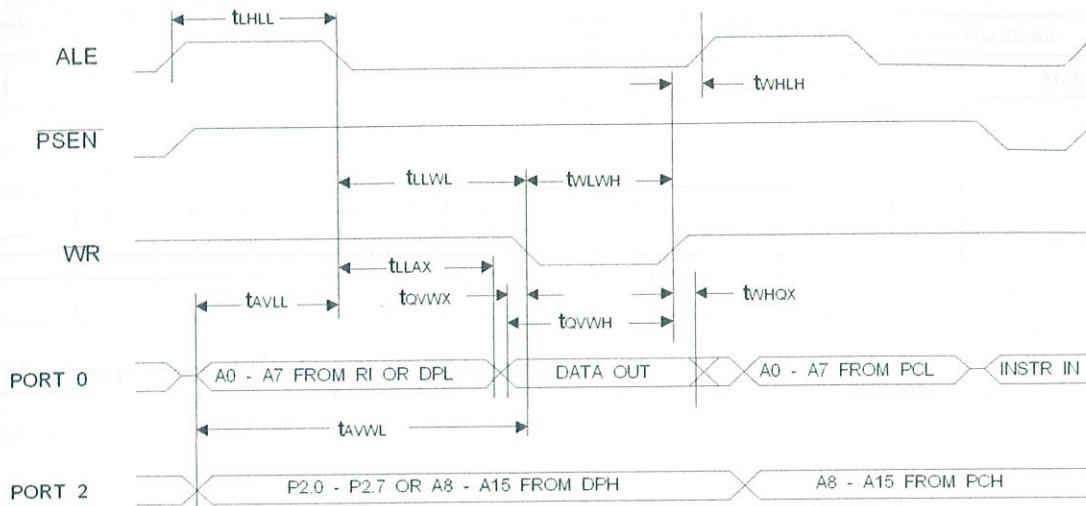


## External Data Memory Read Cycle

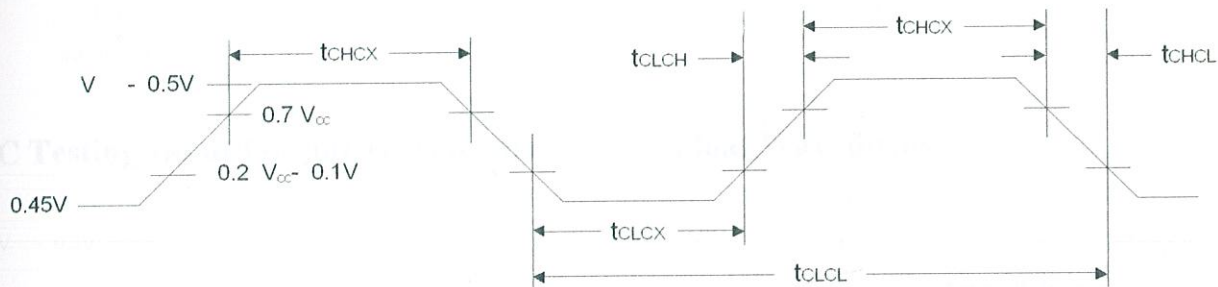


**AT89C51**

## External Data Memory Write Cycle



## External Clock Drive Waveforms



## External Clock Drive

Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
$t_{CLCL}$	Clock Period	41.6		ns
$t_{CHCX}$	High Time	15		ns
$t_{CLCX}$	Low Time	15		ns
$t_{CLCH}$	Rise Time		20	ns
$t_{CHCL}$	Fall Time		20	ns

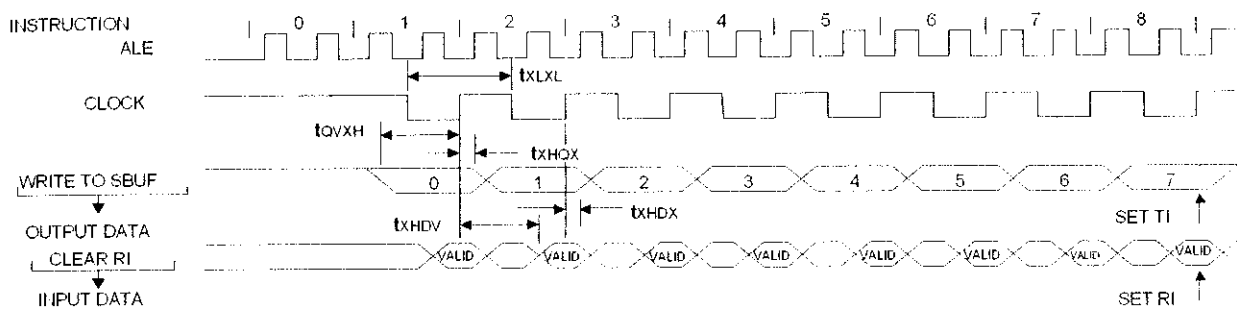


## Serial Port Timing: Shift Register Mode Test Conditions

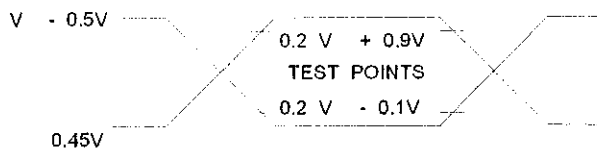
( $V_{CC} = 5.0 \text{ V} \pm 20\%$ ; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
$t_{XLXL}$	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		$\mu\text{s}$
$t_{OVXH}$	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}-133$		ns
$t_{XHX}$	Output Data Hold after Clock Rising Edge	50		$2t_{CLCL}-117$		ns
$t_{XHDx}$	Input Data Hold after Clock Rising Edge	0		0		ns
$t_{XHDV}$	Clock Rising Edge to Input Data Valid		700		$10t_{CLCL}-133$	ns

## Shift Register Mode Timing Waveforms

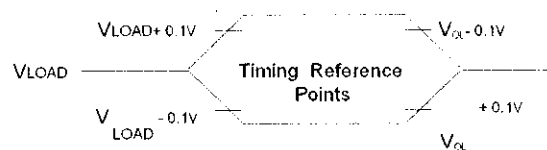


## AC Testing Input/Output Waveforms<sup>(1)</sup>



Note: 1. AC Inputs during testing are driven at  $V_{CC} - 0.5 \text{ V}$  for a logic 1 and  $0.45 \text{ V}$  for a logic 0. Timing measurements are made at  $V_{H, \text{min}}$  for a logic 1 and  $V_{L, \text{max}}$  for a logic 0.

## Float Waveforms<sup>(1)</sup>



Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when 100 mV change from the loaded  $V_{OH}/V_{OL}$  level occurs.

**Ordering Information**

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	5V $\pm$ 20%	AT89C51-12AC	44A	Commercial (0° C to 70° C)
		AT89C51-12JC	44J	
		AT89C51-12PC	40P6	
		AT89C51-12QC	44Q	
		AT89C51-12AI	44A	Industrial (-40° C to 85° C)
		AT89C51-12JI	44J	
		AT89C51-12PI	40P6	
		AT89C51-12QI	44Q	
16	5V $\pm$ 20%	AT89C51-16AC	44A	Commercial (0° C to 70° C)
		AT89C51-16JC	44J	
		AT89C51-16PC	40P6	
		AT89C51-16QC	44Q	
		AT89C51-16AI	44A	Industrial (-40° C to 85° C)
		AT89C51-16JI	44J	
		AT89C51-16PI	40P6	
		AT89C51-16QI	44Q	
20	5V $\pm$ 20%	AT89C51-20AC	44A	Commercial (0° C to 70° C)
		AT89C51-20JC	44J	
		AT89C51-20PC	40P6	
		AT89C51-20QC	44Q	
		AT89C51-20AI	44A	Industrial (-40° C to 85° C)
		AT89C51-20JI	44J	
		AT89C51-20PI	40P6	
		AT89C51-20QI	44Q	
24	5V $\pm$ 20%	AT89C51-24AC	44A	Commercial (0° C to 70° C)
		AT89C51-24JC	44J	
		AT89C51-24PC	40P6	
		AT89C51-24QC	44Q	
		AT89C51-24AI	44A	Industrial (-40° C to 85° C)
		AT89C51-24JI	44J	
		AT89C51-24PI	40P6	
		AT89C51-24QI	44Q	

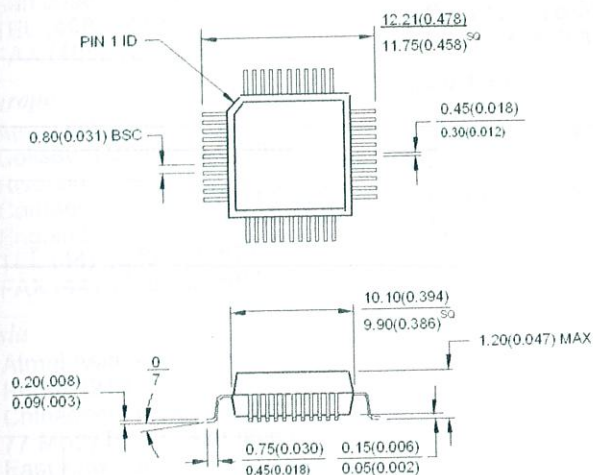
Package Type	
44A	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
44J	44-lead, Plastic J-leaded Chip Carrier (PLCC)
40P6	40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
44Q	44-lead, Plastic Gull Wing Quad Flatpack (PQFP)





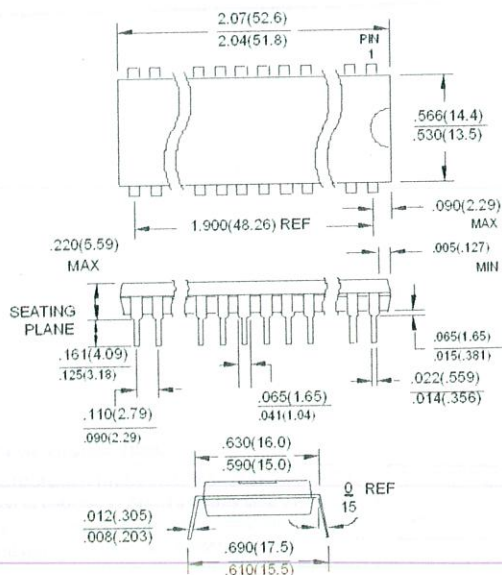
## Packaging Information

**44A**, 44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flatpack (TQFP)  
Dimensions in Millimeters and (Inches)\*  
JEDEC STANDARD MS-026 ACB

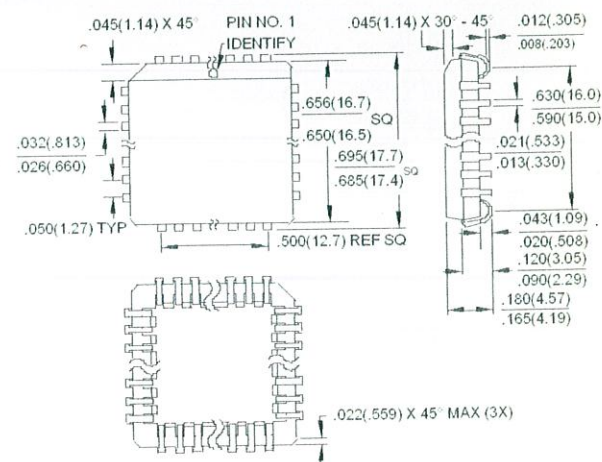


Controlling dimension: millimeters

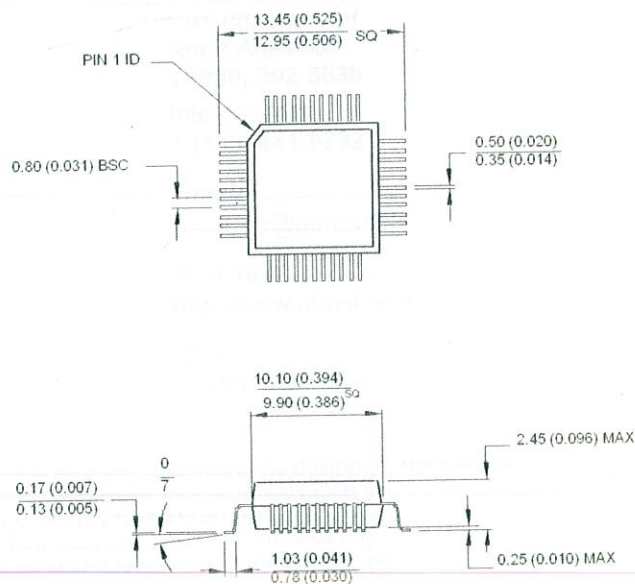
**40P6**, 40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)  
Dimensions in Inches and (Millimeters)



**44J**, 44-lead, Plastic J-leaded Chip Carrier (PLCC)  
Dimensions in Inches and (Millimeters)  
JEDEC STANDARD MS-018 AC



**44Q**, 44-lead, Plastic Quad Flat Package (PQFP)  
Dimensions in Millimeters and (Inches)\*  
JEDEC STANDARD MS-022 AB



Controlling dimension: millimeters



## **Atmel Headquarters**

### **Corporate Headquarters**

2325 Orchard Parkway  
San Jose, CA 95131  
TEL (408) 441-0311  
FAX (408) 487-2600

### **Europe**

Atmel U.K., Ltd.  
Coliseum Business Centre  
Riverside Way  
Camberley, Surrey GU15 3YL  
England  
TEL (44) 1276-686-677  
FAX (44) 1276-686-697

### **Asia**

Atmel Asia, Ltd.  
Room 1219  
Chinachem Golden Plaza  
77 Mody Road Tsimhatsui  
East Kowloon  
Hong Kong  
TEL (852) 2721-9778  
FAX (852) 2722-1369

### **Japan**

Atmel Japan K.K.  
9F, Tonetsu Shinkawa Bldg.  
1-24-8 Shinkawa  
Chuo-ku, Tokyo 104-0033  
Japan  
TEL (81) 3-3523-3551  
FAX (81) 3-3523-7581

## **Atmel Operations**

### **Atmel Colorado Springs**

1150 E. Cheyenne Mtn. Blvd.  
Colorado Springs, CO 80906  
TEL (719) 576-3300  
FAX (719) 540-1759

### **Atmel Rousset**

Zone Industrielle  
13106 Rousset Cedex  
France  
TEL (33) 4-4253-6000  
FAX (33) 4-4253-6001

---

### **Fax-on-Demand**

North America:  
1-(800) 292-8635  
International:  
1-(408) 441-0732

### **e-mail**

[literature@atmel.com](mailto:literature@atmel.com)

### **Web Site**

<http://www.atmel.com>

### **BBS**

1-(408) 436-4309

### **© Atmel Corporation 2000.**

Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

Marks bearing \* and/or ™ are registered trademarks and trademarks of Atmel Corporation.

Terms and product names in this document may be trademarks of others.



Printed on recycled paper.

0265G-02/00/xM



- D Featuring Unitrode L293 and L293D Products Now From Texas Instruments**
- D Wide Supply-Voltage Range: 4.5 V to 36 V**
- D Separate Input-Logic Supply**
- D Internal ESD Protection**
- D Thermal Shutdown**
- D High-Noise-Immunity Inputs**
- D Functional Replacements for SGS L293 and SGS L293D**
- D Output Current 1 A Per Channel (600 mA for L293D)**
- D Peak Output Current 2 A Per Channel (1.2 A for L293D)**
- D Output Clamp Diodes for Inductive Transient Suppression (L293D)**

### description

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

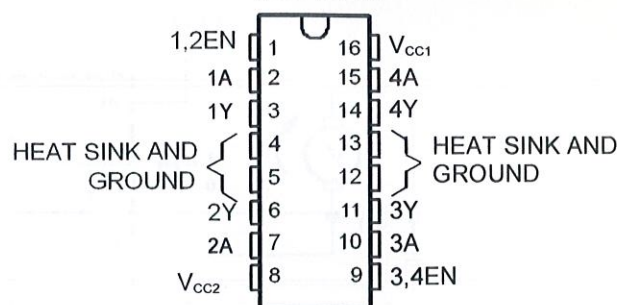
All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression.

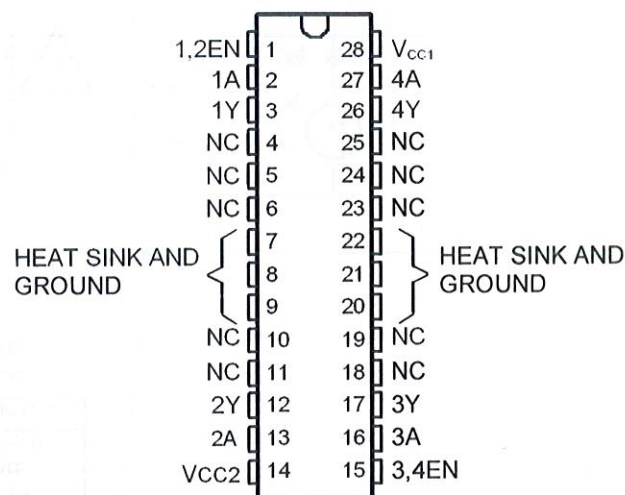
A  $V_{CC1}$  terminal, separate from  $V_{CC2}$ , is provided for the logic inputs to minimize device power dissipation.

The L293 and L293D are characterized for operation from 0°C to 70°C.

**N, NE PACKAGE  
(TOP VIEW)**



**DWP PACKAGE  
(TOP VIEW)**



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication

Copyright | 2002, Texas Instruments Incorporated

date.  
Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

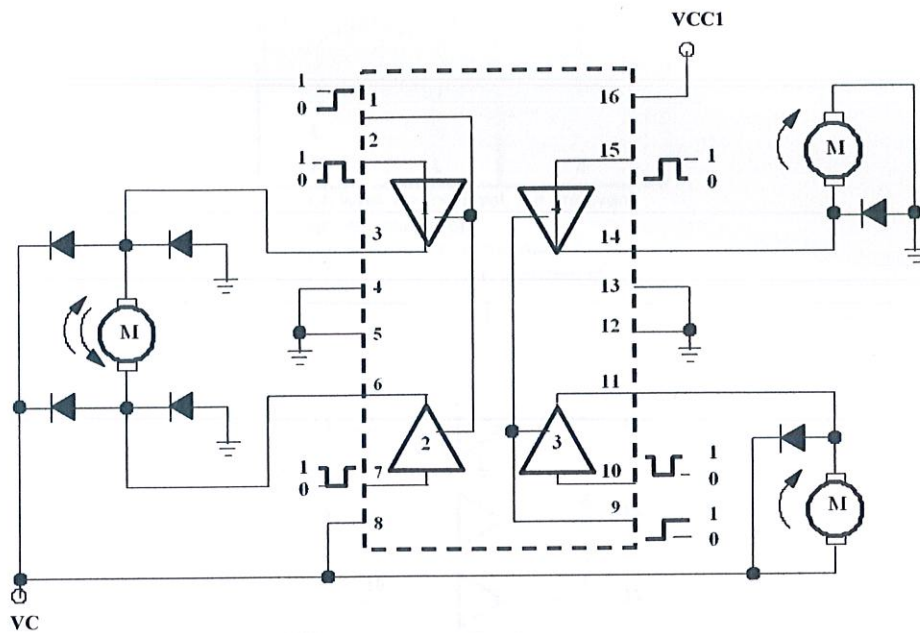
**TEXAS  
INSTRUMENTS**



# L293, L293D QUADRUPLE HALF-H DRIVERS

SLRS008B – SEPTEMBER 1986 – REVISED JUNE 2002

## block diagram



NOTE: Output diodes are internal in L293D.

### TEXAS INSTRUMENTS AVAILABLE OPTIONS

TA	PACKAGE
	PLASTIC DIP (NE)
0°C to 70°C	L293NE
	L293DNE

### Unitrode Products from Texas Instruments AVAILABLE OPTIONS

TA	PACKAGED DEVICES	
	SMALL OUTLINE (DWP)	PLASTIC DIP (N)
0°C to 70°C	L293DWP	L293N
	L293DDWP	L293DN

The DWP package is available taped and reeled. Add the suffix TR to device type (e.g., L293DWPTR).

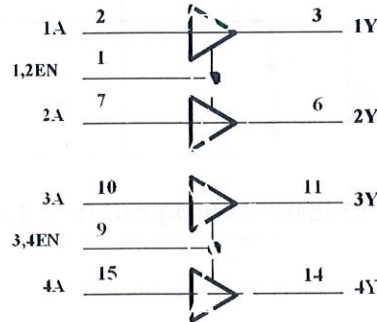
FUNCTION TABLE  
(each driver)

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

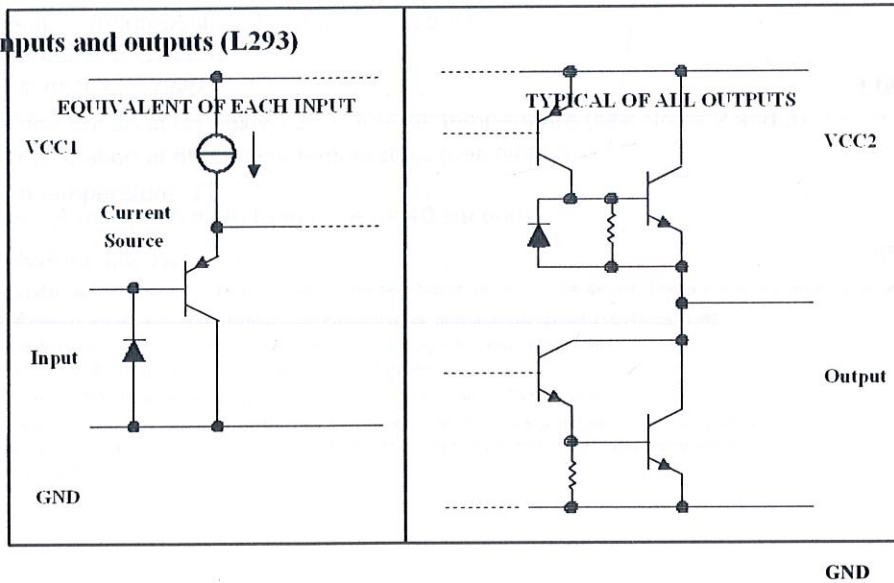
H = high level, L = low level, X = irrelevant,  
Z = high impedance (off)

† In the thermal shutdown mode, the output is  
in the high-impedance state, regardless of  
the input levels.

## logic diagram



## schematics of inputs and outputs (L293)

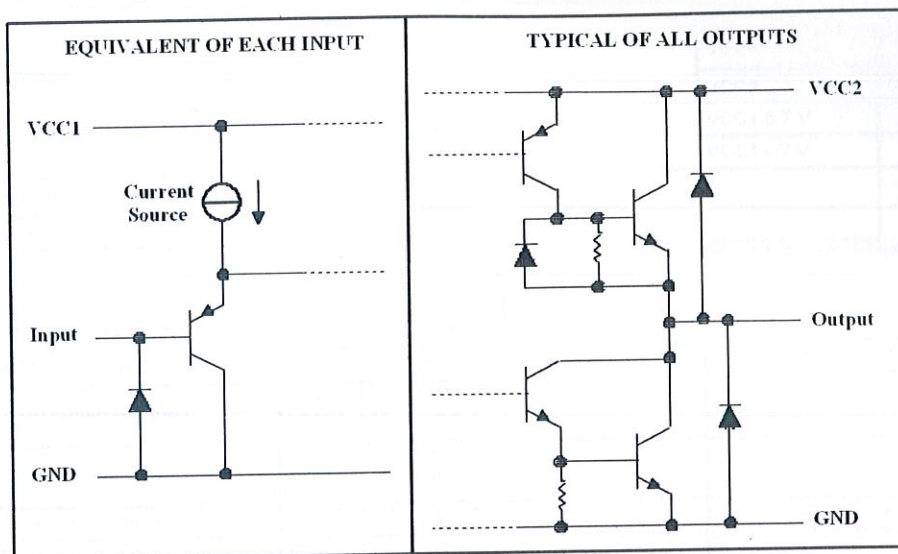




# L293, L293D QUADRUPLE HALF-H DRIVERS

SLRS008B – SEPTEMBER 1986 – REVISED JUNE 2002

## schematics of inputs and outputs (L293D)



### absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, VCC1 (see Note 1)	36 V
Output supply voltage, VCC2	36 V
Input voltage, V <sub>I</sub>	7 V
Output voltage range, V <sub>O</sub>	–3 V to VCC2 + 3 V
Peak output current, I <sub>O</sub> (nonrepetitive, t ≤ 5 ms): L293	±2 A
Peak output current, I <sub>O</sub> (nonrepetitive, t ≤ 100 μs): L293D	±1.2 A
Continuous output current, I <sub>O</sub> : L293	±1 A
Continuous output current, I <sub>O</sub> : L293D	±600 mA
Continuous total dissipation at (or below) 25°C free-air temperature (see Notes 2 and 3)	2075 mW
Continuous total dissipation at 80°C case temperature (see Note 3)	5000 mW
Maximum junction temperature, T <sub>J</sub>	150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds	260°C
Storage temperature range, T <sub>stg</sub>	–65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltage values are with respect to the network ground terminal.

2. For operation above 25°C free-air temperature, derate linearly at the rate of 16.6 mW/°C.

3. For operation above 25°C case temperature, derate linearly at the rate of 71.4 mW/°C. Due to variations in individual device electrical characteristics and thermal resistance, the built-in thermal overload protection may be activated at power levels slightly above or below the rated dissipation.



**recommended operating conditions**

		MIN	MAX	UNIT
Supply voltage	VCC1	4.5	7	V
	VCC2	VCC1	36	
VIH High level input voltage	VCC1 $\pm$ 7 V	2.3	VCC1	V
	VCC1 $\pm$ 7 V	2.3	7	V
VIL Low-level output voltage		-0.3†	1.5	V
TA Operating free-air temperature		0	70	°C

† The algebraic convention, in which the least positive (most negative) designated minimum, is used in this data sheet for logic voltage levels.

**electrical characteristics,  $V_{CC1} = 5\text{ V}$ ,  $V_{CC2} = 24\text{ V}$ ,  $T_A = 25^\circ\text{C}$**

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
VOH	High-level output voltage	L293: IOH = −1 A L293D: IOH = − 0.6 A		VCC2−1.8	VCC2−1.4		V
VOL	Low-level output voltage	L293: IOL = 1 A L293D: IOL = 0.6 A			1.2	1.8	V
VOKH	High-level output clamp voltage	L293D: IOK = − 0.6 A		VCC2 + 1.3			V
VOKL	Low-level output clamp voltage	L293D: IOK = 0.6 A		1.3			V
IIH	High level input current	A	VI = 7 V		0.2	100	αA
		EN			0.2	10	
IIL	Low level input current	A	VI = 0		−3	−10	αA
		EN			−2	−100	
ICC1	Logic supply current	IO = 0	All outputs at high level		13	22	mA
			All outputs at low level		35	60	
			All outputs at high impedance		8	24	
ICC2	Output supply current	IO = 0	All outputs at high level		14	24	mA
			All outputs at low level		2	6	
			All outputs at high impedance		2	4	

**switching characteristics,  $V_{CC1} = 5\text{ V}$ ,  $V_{CC2} = 24\text{ V}$ ,  $T_A = 25^\circ\text{C}$**

PARAMETER		TEST CONDITIONS	L293NE, L293DNE			UNIT
			MIN	TYP	MAX	
tPLH Propagation delay time, low-to-high-level output from A input		CL = 30 pF See Figure 1		800		ns
tPHL Propagation delay time, high-to-low-level output from A input				400		ns
tTLH Transition time, low-to-high-level output				300		ns
tTHL Transition time, high-to-low-level output				300		ns

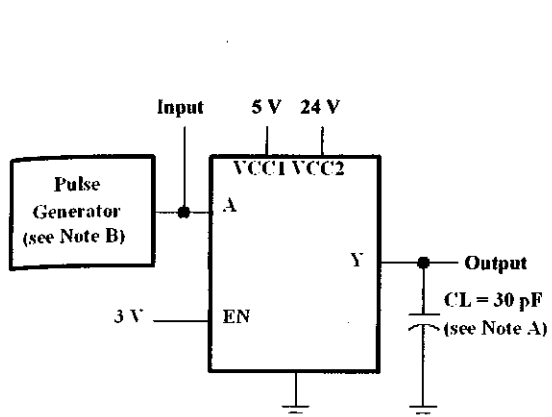
**switching characteristics,  $V_{CC1} = 5\text{ V}$ ,  $V_{CC2} = 24\text{ V}$ ,  $T_A = 25^\circ\text{C}$**

PARAMETER		TEST CONDITIONS	L293DWP, L293N L293DDWP, L293DN			UNIT
			MIN	TYP	MAX	
tPLH Propagation delay time, low-to-high-level output from A input		CL = 30 pF See Figure 1		750		ns
tPHL Propagation delay time, high-to-low-level output from A input				200		ns
tTLH Transition time, low-to-high-level output				100		ns
tTHL Transition time, high-to-low-level output				350		ns



POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

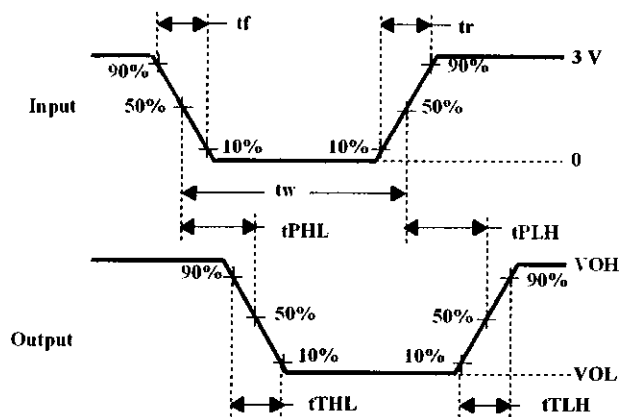
**PARAMETER MEASUREMENT INFORMATION**



**TEST CIRCUIT**

NOTES: A. CL includes probe and jig capacitance.

B. The pulse generator has the following characteristics:  $t_r \leq 10$  ns,  $t_f \leq 10$  ns,  $t_w = 10$   $\mu$ s, PRR = 5 kHz,  $Z_O = 50 \Omega$ .



**VOLTAGE WAVEFORMS**

**Figure 1. Test Circuit and Voltage Waveforms**



APPLICATION INFORMATION

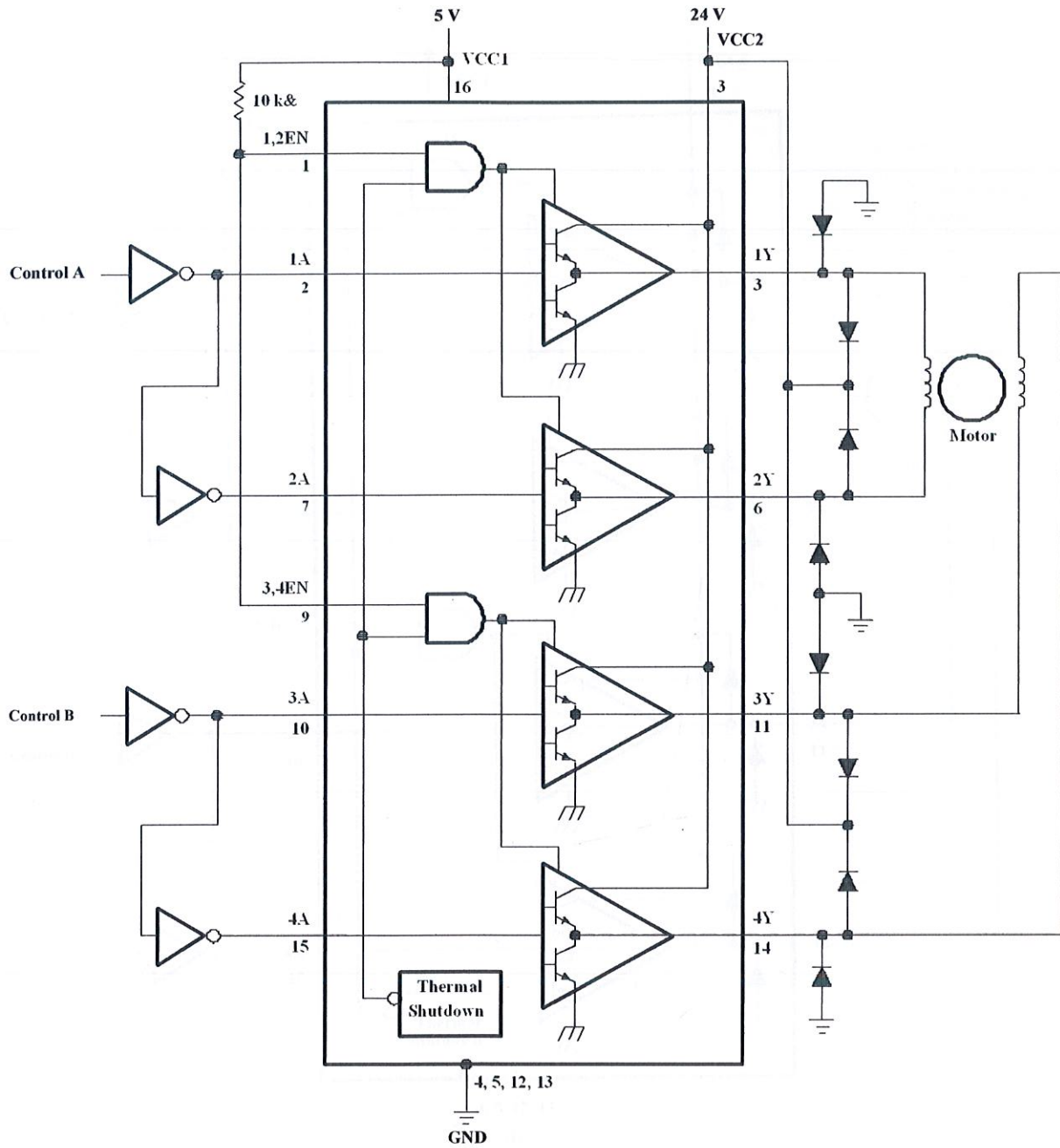


Figure 2. Two-Phase Motor Driver (L293)



# APPLICATION INFORMATION

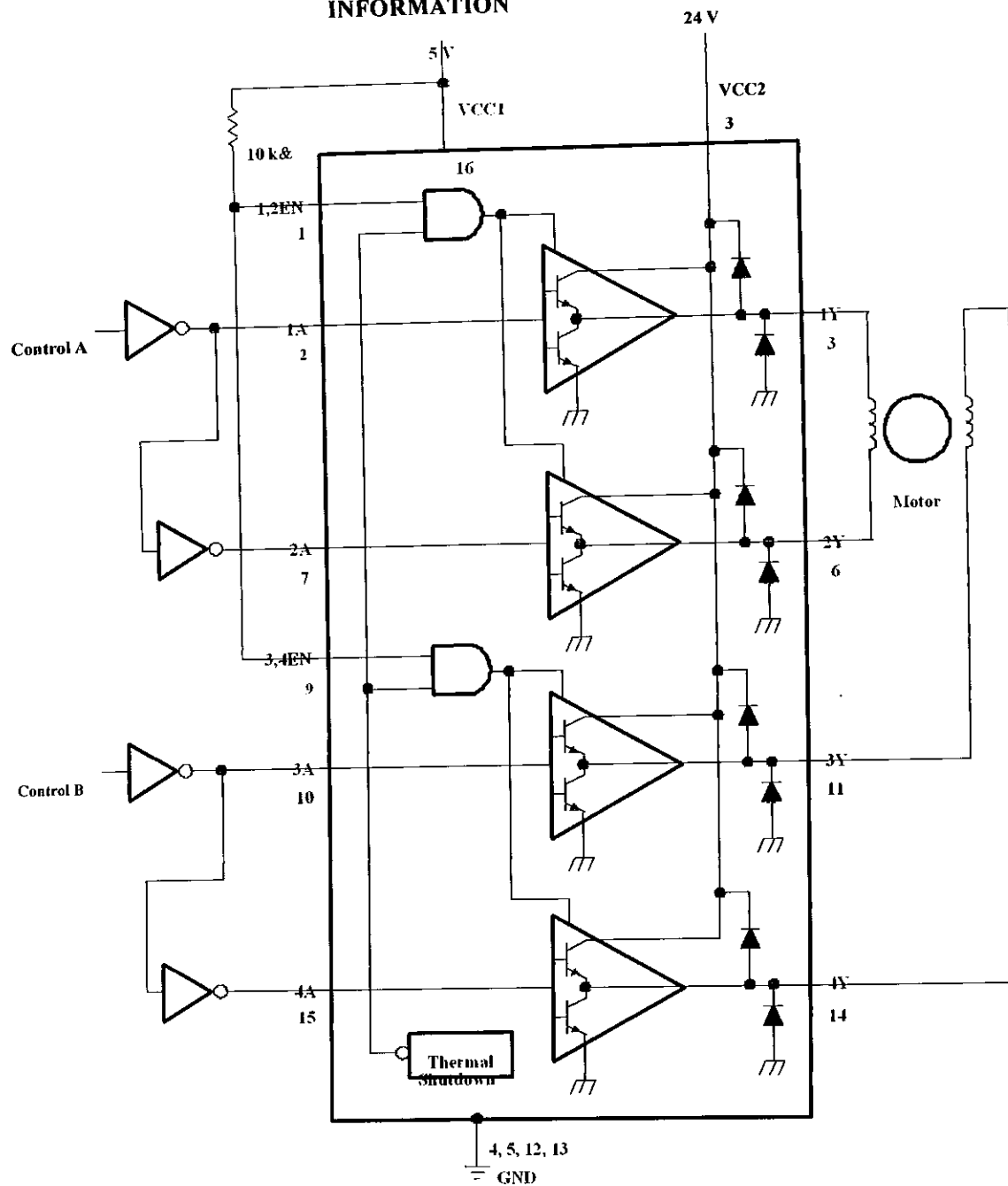
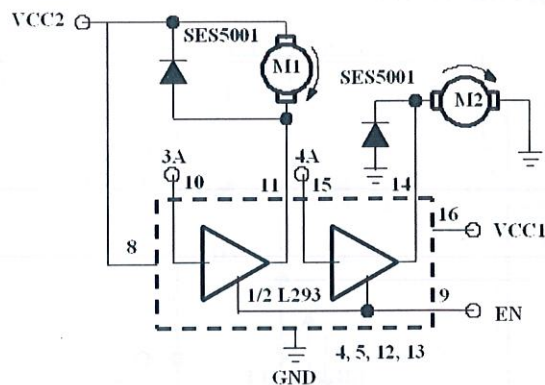


Figure 3. Two-Phase Motor Driver (L293D)



POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

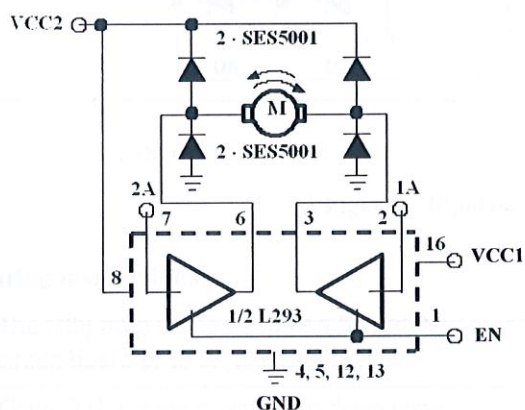
# APPLICATION INFORMATION



EN	3A	M1	4A	M2
H	H	Fast motor stop	H	Run
H	L	Run	L	Fast motor stop
L	X	Free-running motor stop	X	Free-running motor stop

L = low, H = high, X = don't care

Figure 4. DC Motor Controls  
(connections to ground and to supply voltage)

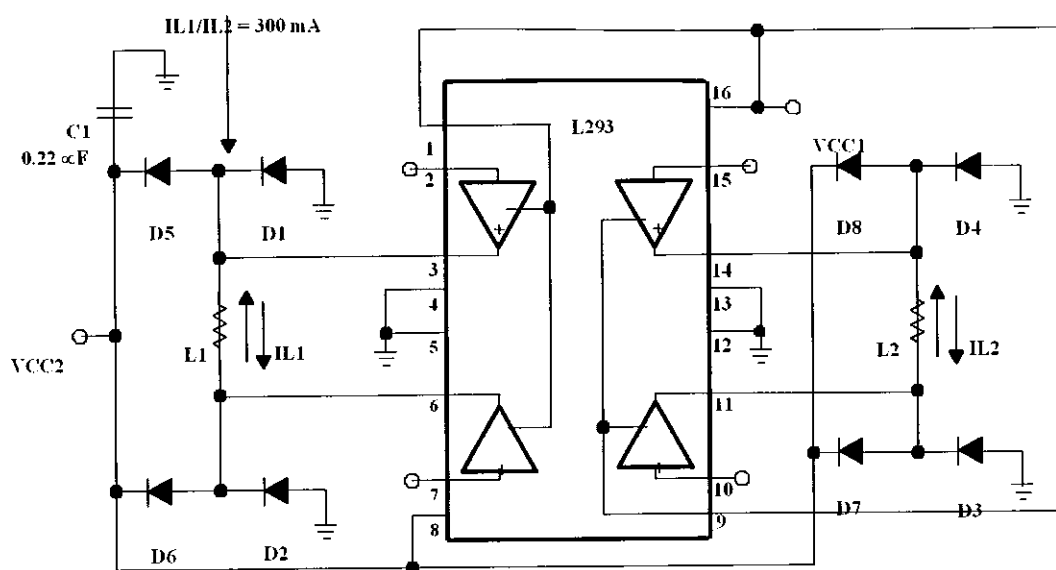


EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care

Figure 5. Bidirectional DC Motor Control

**APPLICATION  
 INFORMATION**



D1-D8 = SES5001

**Figure 6. Bipolar Stepping-Motor Control**

**mounting instructions**

The  $R_{thj-amp}$  of the L293 can be reduced by soldering the GND pins to a suitable copper area of the printed circuit board or to an external heatsink.

Figure 9 shows the maximum package power  $P_{TOT}$  and the  $\theta_{JA}$  as a function of the side of two equal square copper areas having a thickness of 35  $\mu$ m (see Figure 7). In addition, an external heat sink can be used (see Figure 8).

During soldering, the pin temperature must not exceed 260°C, and the soldering time must not be longer than 12 seconds.

The external heatsink or printed circuit copper area must be connected to electrical ground.



# APPLICATION INFORMATION

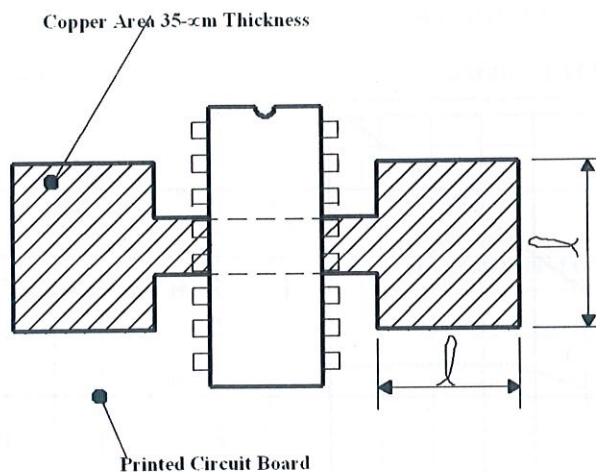


Figure 7. Example of Printed Circuit Board Copper Area  
 (used as heat sink)

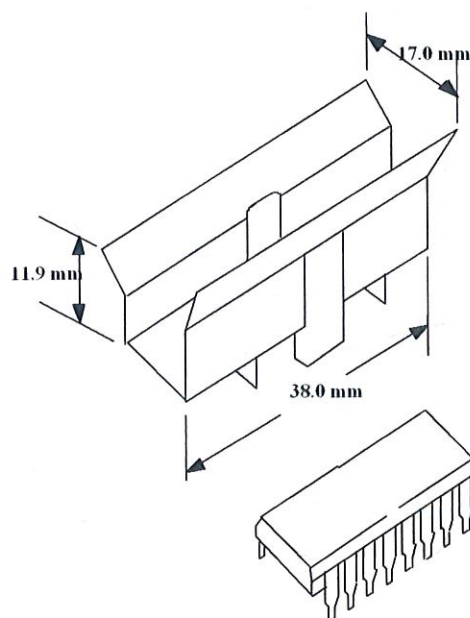


Figure 8. External Heat Sink Mounting Example  
 ( $\theta_{JA} = 25^{\circ}\text{C/W}$ )



# APPLICATION INFORMATION

## MAXIMUM POWER AND JUNCTION

vs

## THERMAL RESISTANCE

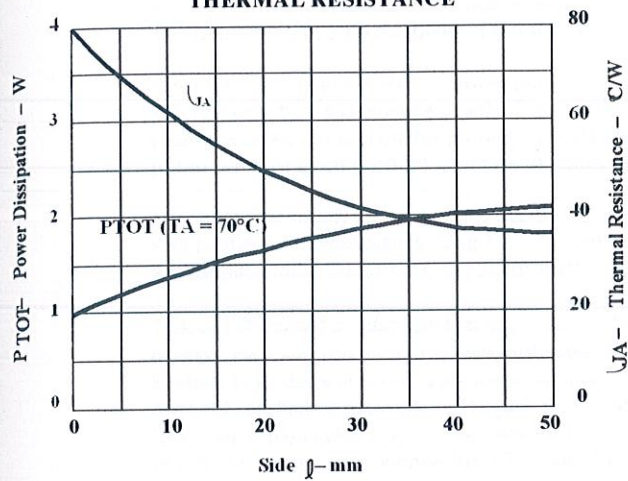


Figure 9

## MAXIMUM POWER DISSIPATION

vs

## AMBIENT TEMPERATURE

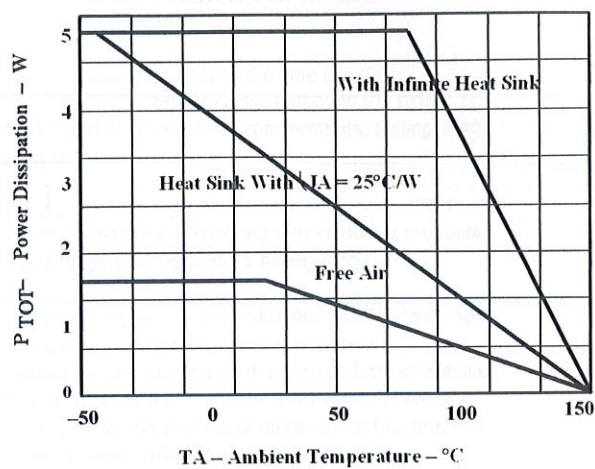


Figure 10

### IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

#### Mailing Address:

Texas Instruments  
Post Office Box 655303  
Dallas, Texas 75265

Copyright © 2002, Texas Instruments Incorporated





## LOW POWER QUAD OPERATIONAL AMPLIFIERS

- WIDE GAIN BANDWIDTH : 1.3MHz
- INPUT COMMON-MODE VOLTAGE RANGE INCLUDES GROUND
- LARGE VOLTAGE GAIN : 100dB
- VERY LOW SUPPLY CURRENT/AMPLI : 375 $\mu$ A
- LOW INPUT BIAS CURRENT : 20nA
- LOW INPUT OFFSET VOLTAGE : 5mV max.  
(for more accurate applications, use the equivalent parts LM124A-LM224A-LM324A which feature 3mV max.)
- LOW INPUT OFFSET CURRENT : 2nA
- WIDE POWER SUPPLY RANGE :  
SINGLE SUPPLY : +3V TO +30V  
DUAL SUPPLIES :  $\pm 1.5V$  TO  $\pm 15V$

### DESCRIPTION

These circuits consist of four independent, high gain, internally frequency compensated operational amplifiers. They operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage.

### ORDER CODE

Part Number	Temperature Range	Package		
		N	D	P
LM124	-55°C, +125°C	•	•	•
LM224	-40°C, +105°C	•	•	•
LM324	0°C, +70°C	•	•	•
Example : LM224N				

N = Dual In Line Package (DIP)

D = Small Outline Package (SO) - also available in Tape & Reel (DT)

P = Thin Shrink Small Outline Package (TSSOP) - only available in Tape & Reel (PT)



**N**  
**DIP14**  
(Plastic Package)

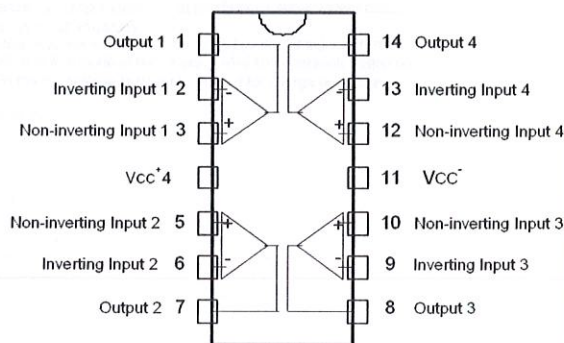


**D**  
**SO14**  
(Plastic Micropackage)

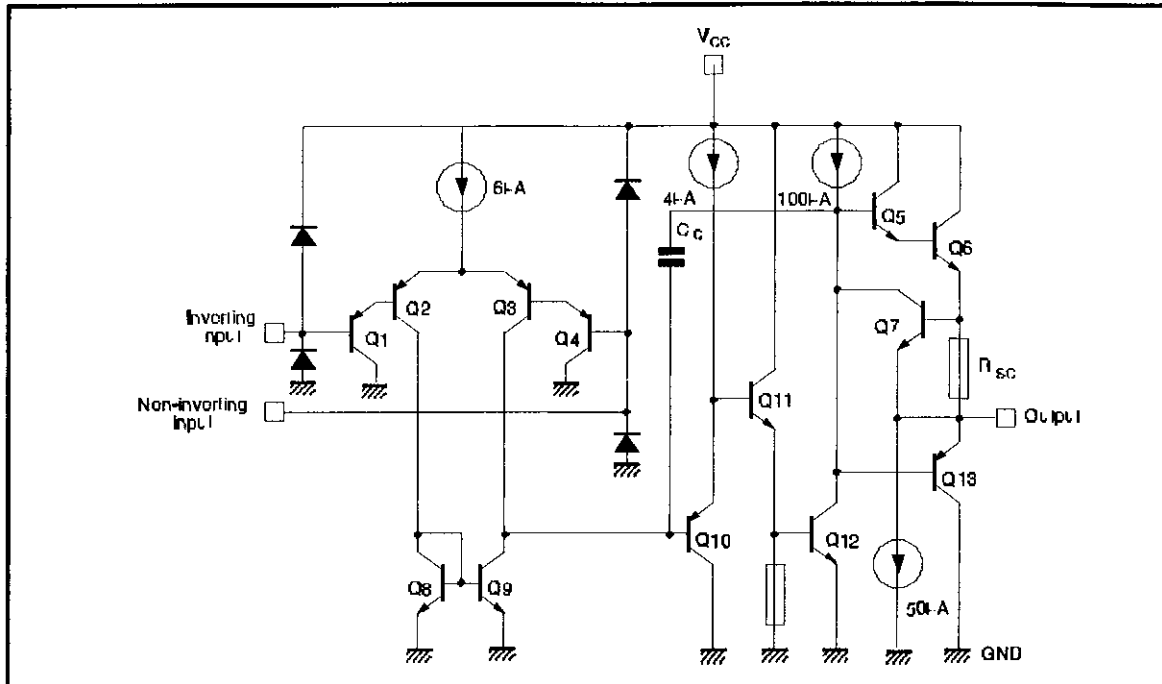


**P**  
**TSSOP14**  
(Thin Shrink Small Outline Package)

### PIN CONNECTIONS (top view)



## SCHEMATIC DIAGRAM (1/4 LM124)



## ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	LM124	LM224	LM324	Unit
$V_{cc}$	Supply voltage	$\pm 16$ or 32			V
$V_i$	Input Voltage	-0.3 to +32			V
$V_{id}$	Differential Input Voltage <sup>1</sup>	+32			V
$P_{tot}$	Power Dissipation	500	500	500	mW
	N Suffix	500	400	400	mW
	D Suffix	500	400	400	mW
	Output Short-circuit Duration <sup>2</sup>	Infinite			
$I_{in}$	Input Current <sup>3</sup>	50	50	50	mA
$T_{oper}$	Operating Free-air Temperature Range	-55 to +125	-40 to +105	0 to +70	°C
$T_{stg}$	Storage Temperature Range	-65 to +150			°C

1. Either or both input voltages must not exceed the magnitude of  $V_{cc}+$  or  $V_{cc-}$ .
2. Short-circuits from the output to  $V_{cc}$  can cause excessive heating if  $V_{cc} > 15V$ . The maximum output current is approximately 40mA independent of the magnitude of  $V_{cc}$ . Destructive dissipation can result from simultaneous short-circuit on all amplifiers.
3. This input current only exists when the voltage at any of the input leads is driven negative. It is due to the collector-base junction of the input PNP transistor becoming forward biased and thereby acting as input diodes clamps. In addition to this diode action, there is also NPN parasitic action on the IC chip. this transistor action can cause the output voltages of the Op-amps to go to the  $V_{cc}$  voltage level (or to ground for a large overdrive) for the time duration than an input is driven negative. This is not destructive and normal output will set up again for input voltage higher than -0.3V.



## ELECTRICAL CHARACTERISTICS

 $V_{CC+} = +5V$ ,  $V_{CC-} = \text{Ground}$ ,  $V_o = 1.4V$ ,  $T_{amb} = +25^\circ C$  (unless otherwise specified)

Symbol	Parameter	Min.	Typ.	Max.	Unit
$V_{io}$	Input Offset Voltage - note 1) $T_{amb} = +25^\circ C$ LM324 $T_{min} \delta T_{amb} \delta T_{max}$ LM324		2	5 7 7 9	mV
$I_{io}$	Input Offset Current $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$		2	30 100	nA
$I_{ib}$	Input Bias Current - note 2) $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$		20	150 300	nA
$A_{vd}$	Large Signal Voltage Gain $V_{CC+} = +15V$ , $R_L = 2k\Omega$ , $V_o = 1.4V$ to $11.4V$ $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$	50 25	100		V/mV
SVR	Supply Voltage Rejection Ratio ( $R_L \delta 10k\Omega$ ) $V_{CC+} = 5V$ to $30V$ $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$	65 65	110		dB
$I_{CC}$	Supply Current, all Amp, no load $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$ $V_{CC} = +5V$ $V_{CC} = +30V$ $V_{CC} = +5V$ $V_{CC} = +30V$		0.7 1.5 0.8 1.5	1.2 3 1.2 3	mA
$V_{ICM}$	Input Common Mode Voltage Range $V_{CC} = +30V$ - note 3) $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$	0 0		$V_{CC} - 1.5$ $V_{CC} - 2$	V
CMR	Common Mode Rejection Ratio ( $R_L \delta 10k\Omega$ ) $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$	70 60	80		dB
$I_{source}$	Output Current Source ( $V_{id} = +1V$ ) $V_{CC} = +15V$ , $V_o = +2V$	20	40	70	mA
$I_{sink}$	Output Sink Current ( $V_{id} = -1V$ ) $V_{CC} = +15V$ , $V_o = +2V$ $V_{CC} = +15V$ , $V_o = +0.2V$	10 12	20 50		mA $\propto A$
$V_{OH}$	High Level Output Voltage $V_{CC} = +30V$ $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$ $R_L = 2k\Omega$ $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$ $R_L = 10k\Omega$ $V_{CC} = +5V$ , $R_L = 2k\Omega$ $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$	26 26 27 27 3.5 3	27 28		V

Symbol	Parameter	Min.	Typ.	Max.	Unit
VOL	Low Level Output Voltage ( $R_L = 10k\Omega$ ) $T_{amb} = +25^\circ C$ $T_{min} \delta T_{amb} \delta T_{max}$		5	20 20	mV
SR	$V_{CC} = 15V$ , $V_i = 0.5$ to $3V$ , $R_L = 2k\Omega$ , $C_L = 100pF$ , unity Gain		0.4		V/ $\mu s$
GBW Rate	Gain Bandwidth Product $V_{CC} = 30V$ , $f = 100kHz$ , $V_o = 10mV$ , $R_L = 2k\Omega$ , $C_L = 100pF$		1.3		MHz
THD	Total Harmonic Distortion $f = 1kHz$ , $A_v = 20dB$ , $R_L = 2k\Omega$ , $V_o = 2V_{pp}$ , $C_L = 100pF$ , $V_{CC} = 30V$		0.015		%
$e_n$	Equivalent Input Noise Voltage $f = 1kHz$ , $R_i = 100\Omega$ , $V_{CC} = 30V$		40		nV/ $\sqrt{Hz}$
$DV_{io}$	Input Offset Voltage Drift		7	30	$\mu V/^\circ C$
$DI_{io}$	Input Offset Current Drift		10	200	pA/ $^\circ C$

$V_{oi}/V_{oz}$  Channel Separation - note

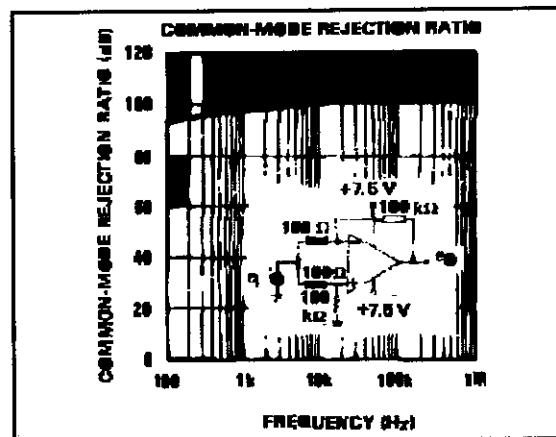
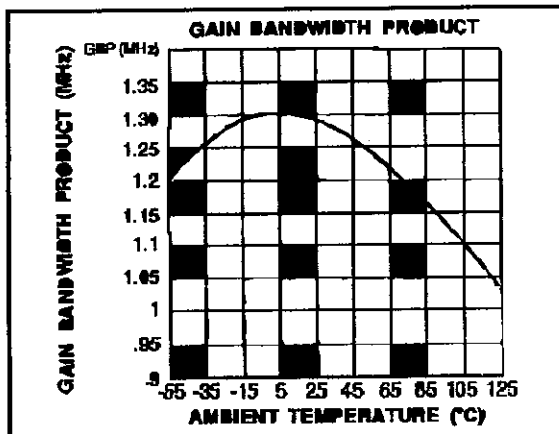
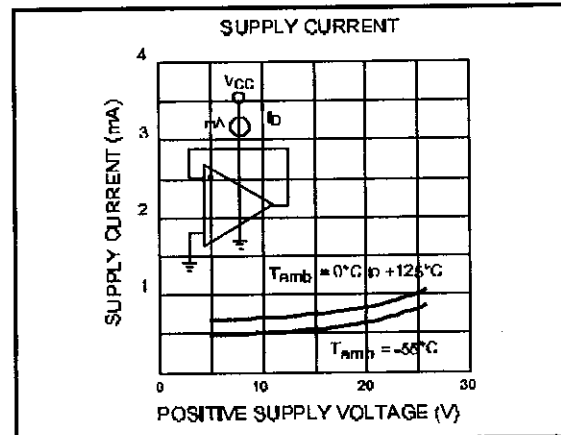
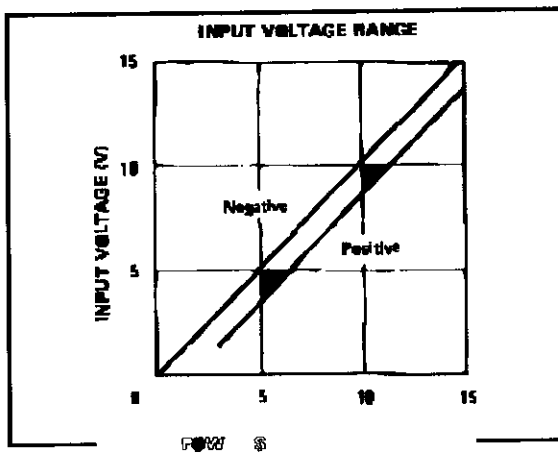
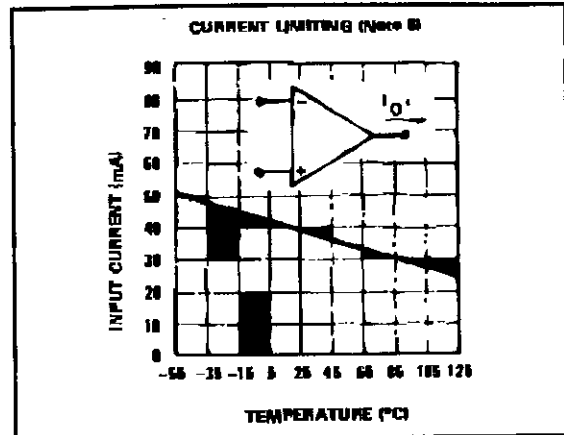
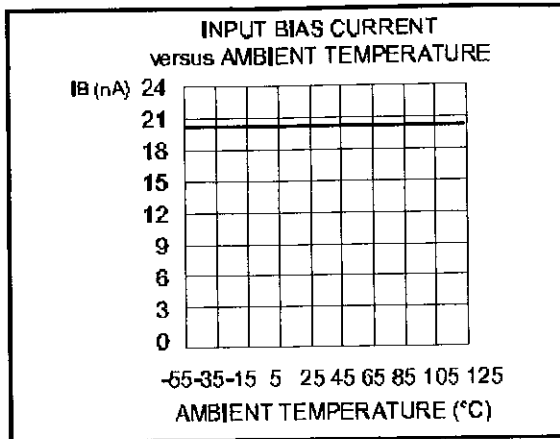
1kHz  $\delta f \delta 20kHz$

120

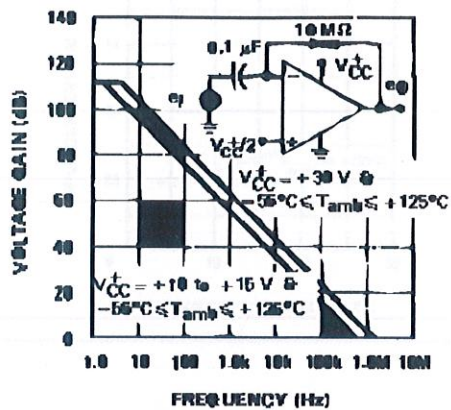
dB

- $V_o = 1.4V$ ,  $R_s = 0\Omega$ ,  $5V < V_{CC} < 30V$ ,  $0 < V_e < V_{CC} - 1.5V$
- The direction of the input current is out of the IC. This current is essentially constant, independent of the state of the output so no loading change exists on the input lines.
- The input common-mode voltage of either input signal voltage should not be allowed to go negative by more than 0.3V. The upper end of the common-mode voltage range is  $V_{CC} - 1.5V$ , but either or both inputs can go to +32V without damage.
- Due to the proximity of external components insure that coupling is not originating via stray capacitance between these external parts. This typically can be detected as this type of capacitance increases at higher frequencies.

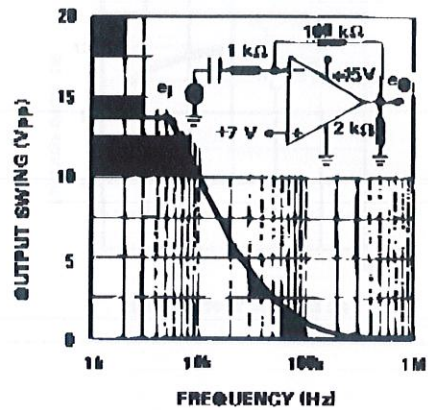




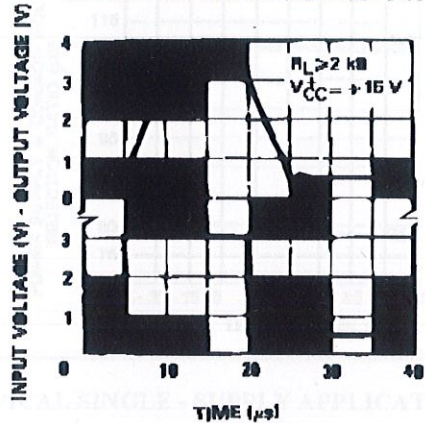
OPEN LOOP FREQUENCY RESPONSE



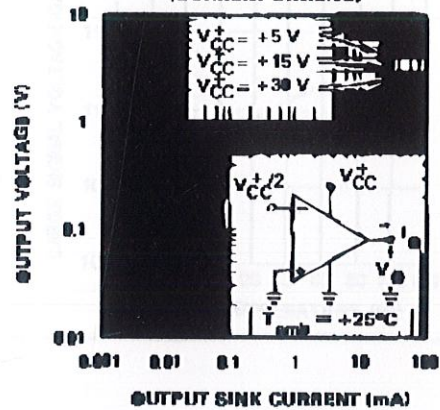
LARGE SIGNAL FREQUENCY RESPONSE



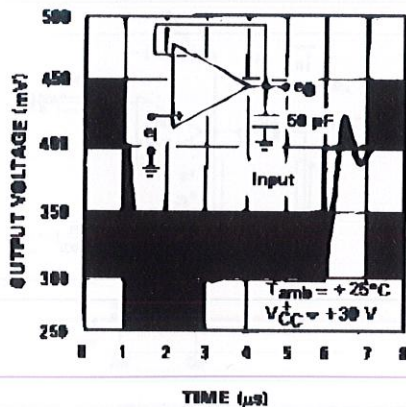
VOLTAGE FOLLOWER PULSE RESPONSE



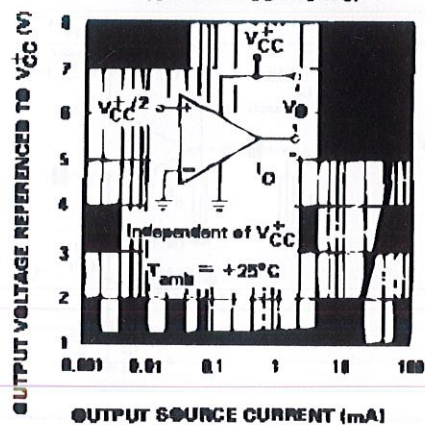
OUTPUT CHARACTERISTICS (CURRENT SINKING)



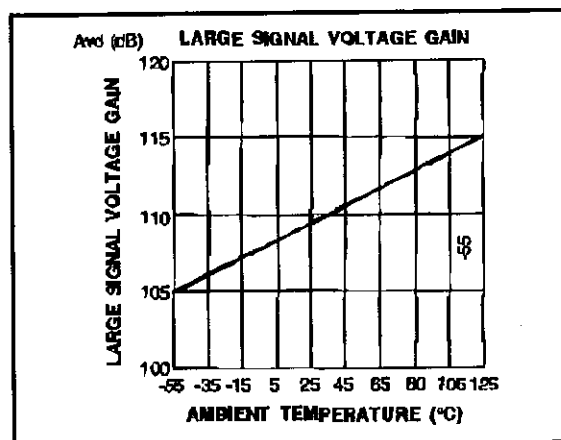
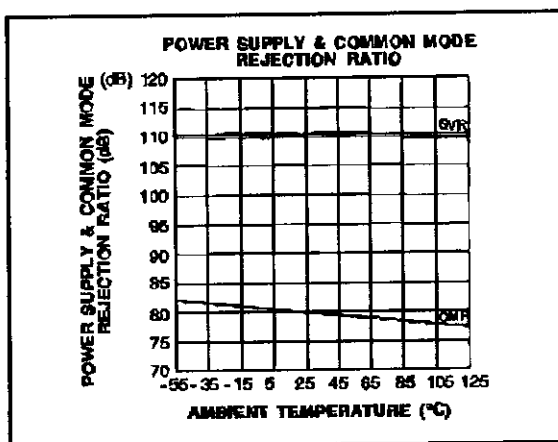
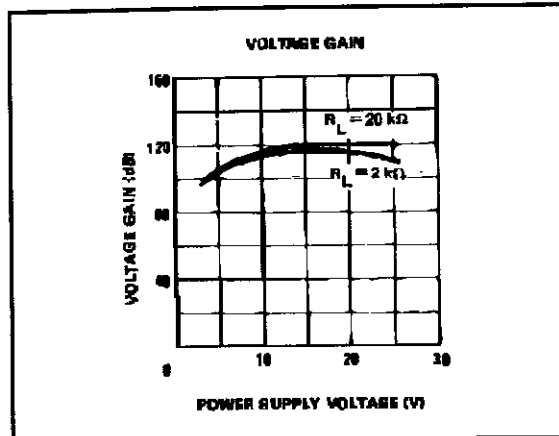
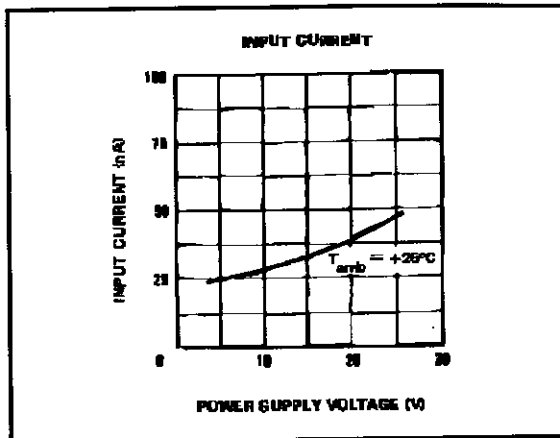
VOLTAGE FOLLOWER PULSE RESPONSE (SMALL SIGNAL)



OUTPUT CHARACTERISTICS (CURRENT SOURCING)

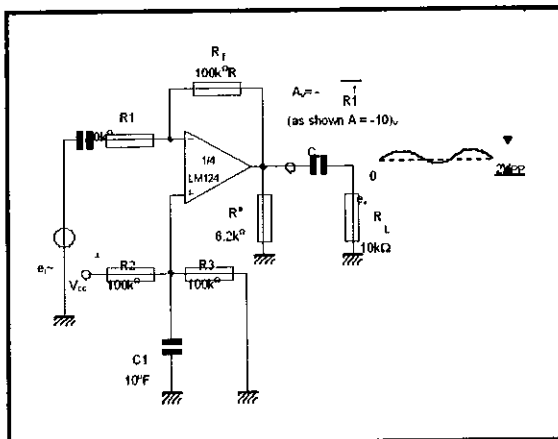




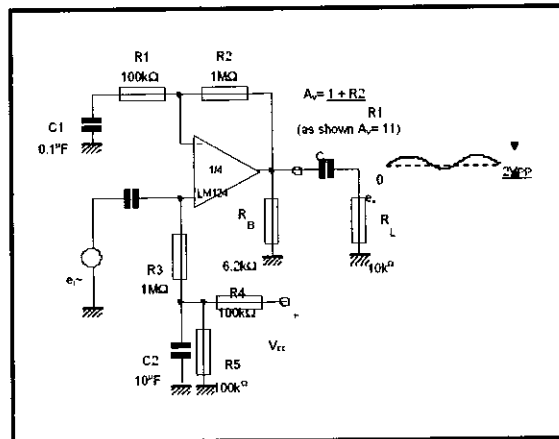


### TYPICAL SINGLE - SUPPLY APPLICATIONS

#### AC COUPLED INVERTING AMPLIFIER

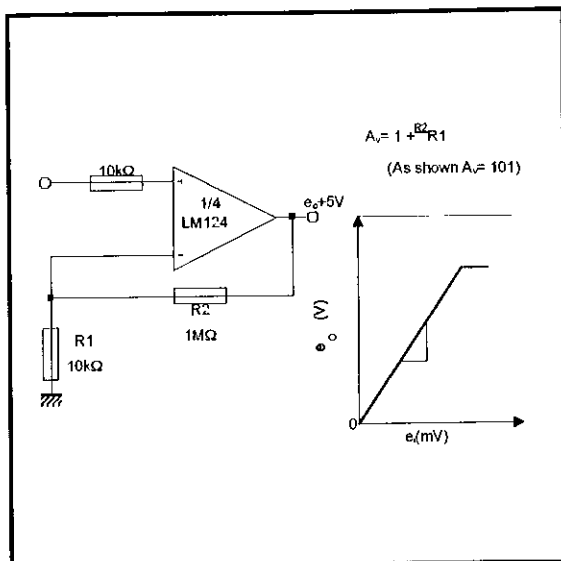


#### AC COUPLED NON INVERTING AMPLIFIER

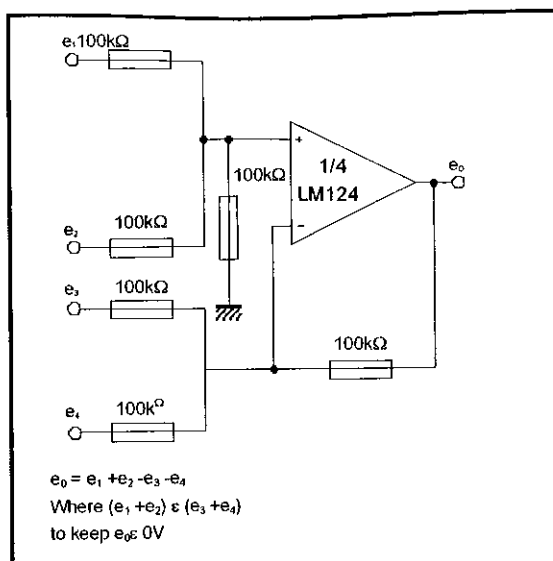


## TYPICAL SINGLE - SUPPLY APPLICATIONS

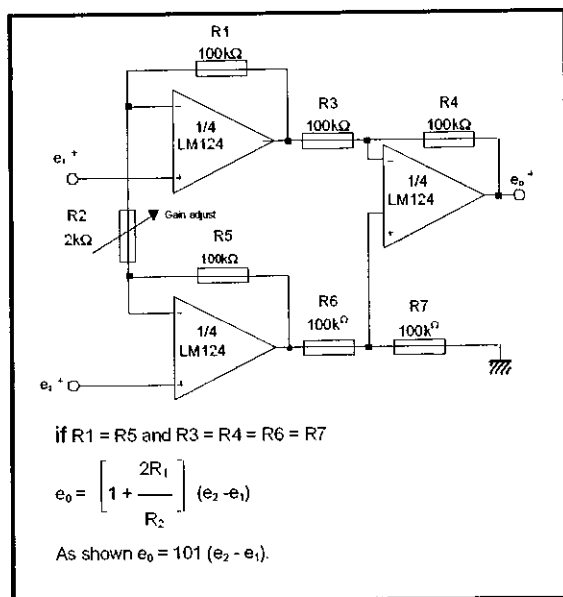
## NON-INVERTING DC GAIN



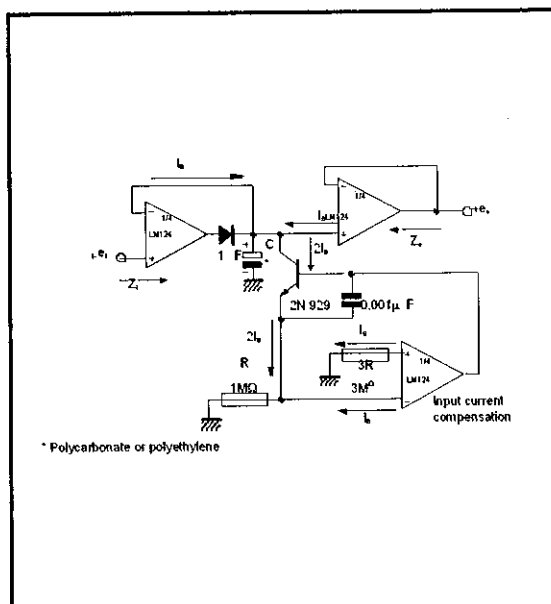
## DC SUMMING AMPLIFIER



## HIGH INPUT Z ADJUSTABLE GAIN DC INSTRUMENTATION AMPLIFIER

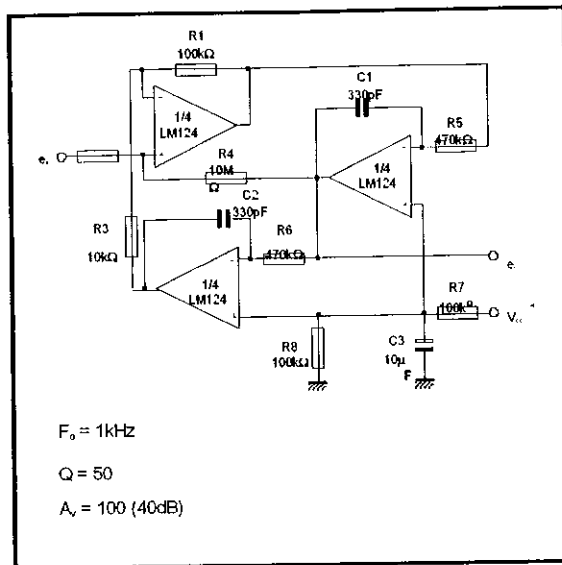


## LOW DRIFT PEAK DETECTOR

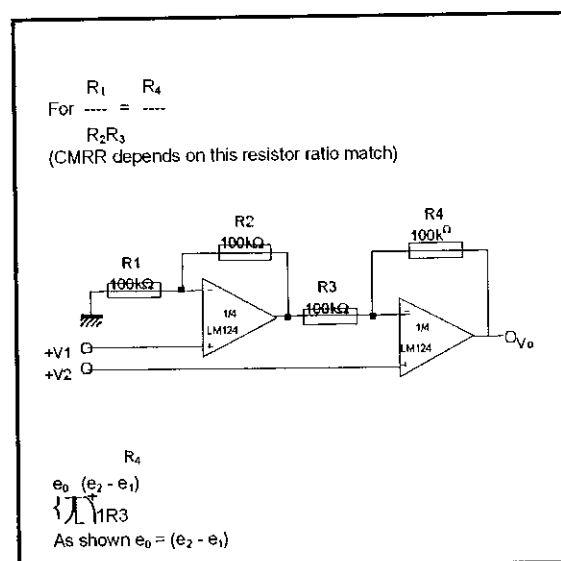


## TYPICAL SINGLE - SUPPLY APPLICATIONS

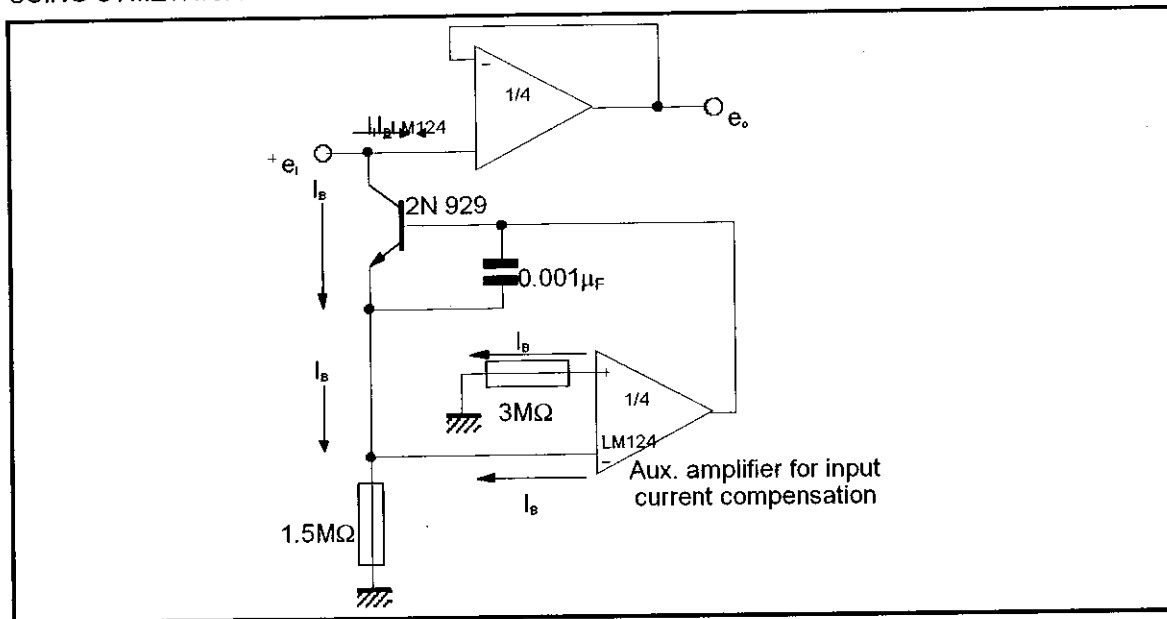
## ACTIVER BANDPASS FILTER



## HIGH INPUT Z, DC DIFFERENTIAL AMPLIFIER



## USING SYMETRICAL AMPLIFIERS TO REDUCE INPUT CURRENT (GENERAL CONCEPT)



**MACROMODEL**

\*\* Standard Linear Ics Macromodels, 1993.

\*\* CONNECTIONS :

- \* 1 INVERTING INPUT
- \* 2 NON-INVERTING INPUT
- \* 3 OUTPUT
- \* 4 POSITIVE POWER SUPPLY
- \* 5 NEGATIVE POWER SUPPLY

.SUBCKT LM124 1 3 2 4 5 (analog)

\*\*\*\*\*

.MODEL MDTH D IS=1E-8 KF=3.104131E-15  
CJO=10F

\* INPUT STAGE

CIP 2 5 1.000000E-12  
 CIN 1 5 1.000000E-12  
 EIP 10 5 2 5 1  
 EIN 16 5 1 5 1  
 RIP 10 11 2.600000E+01  
 RIN 15 16 2.600000E+01  
 RIS 11 15 2.003862E+02  
 DIP 11 12 MDTH 400E-12  
 DIN 15 14 MDTH 400E-12  
 VOFP 12 13 DC 0  
 VOFN 13 14 DC 0  
 IPOL 13 5 1.000000E-05  
 CPS 11 15 3.783376E-09  
 DINN 17 13 MDTH 400E-12

VIN 17 5 0.000000E+00  
 DINR 15 18 MDTH 400E-12  
 VIP 4 18 2.000000E+00  
 FCP 4 5 VOFP 3.400000E+01  
 FCN 5 4 VOFN 3.400000E+01  
 FIBP 2 5 VOFN 2.000000E-03  
 FIBN 5 1 VOFP 2.000000E-03  
 \* AMPLIFYING STAGE  
 FIP 5 19 VOFP 3.600000E+02  
 FIN 5 19 VOFN 3.600000E+02  
 RG1 19 5 3.652997E+06  
 RG2 19 4 3.652997E+06  
 CC 19 5 6.000000E-09  
 DOPM 19 22 MDTH 400E-12  
 DONM 21 19 MDTH 400E-12  
 HOPM 22 28 VOUT 7.500000E+03  
 VIPM 28 4 1.500000E+02  
 HONM 21 27 VOUT 7.500000E+03  
 VINM 5 27 1.500000E+02  
 EOUT 26 23 19 5 1  
 VOUT 23 5 0  
 ROUT 26 3 20  
 COUT 3 5 1.000000E-12  
 DOP 19 25 MDTH 400E-12  
 VOP 4 25 2.242230E+00  
 DON 24 19 MDTH 400E-12  
 VON 24 5 7.922301E-01  
 .ENDS

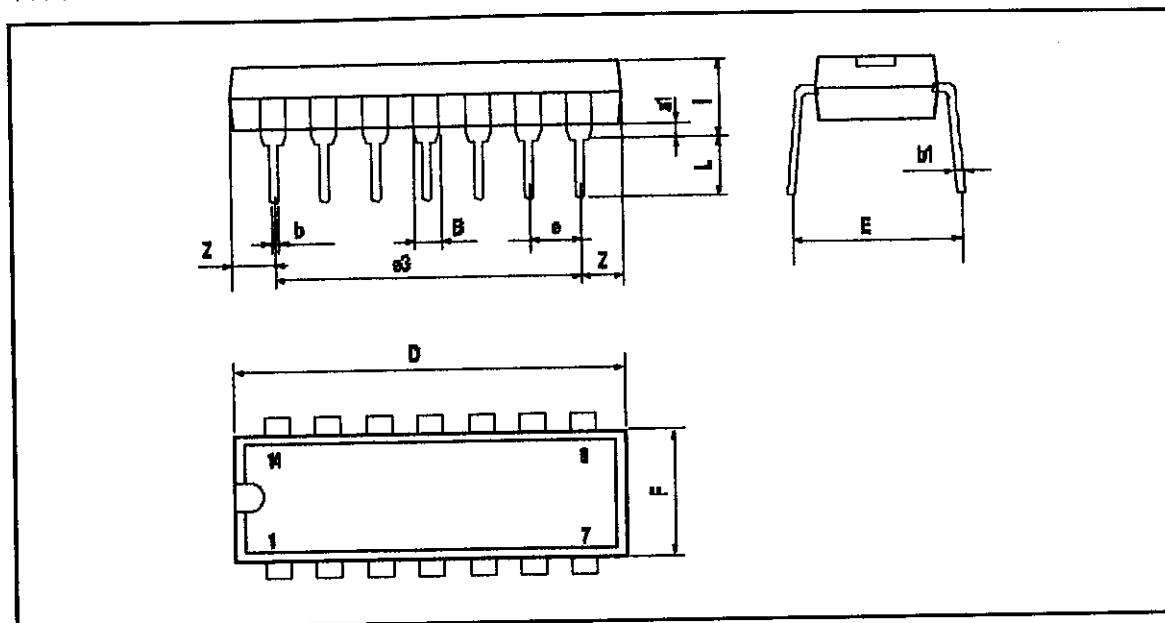
**ELECTRICAL CHARACTERISTICS** $V_{cc+} = +15V$ ,  $V_{cc-} = 0V$ ,  $T_{amb} = 25^{\circ}C$  (unless otherwise specified)

Symbol	Conditions	Value	Unit
$V_{io}$		0	mV
$A_{vd}R_L = 2k\Omega$		100	V/mV
$I_{cc}$ No load, per amplifier		350	$\mu A$
$V_{icm}$		-15 to +13.5	V
$V_{OH}R_L = 2k\Omega$ ( $V_{cc+} = +15V$ )		+13.5	V
$V_{OL}R_L = 10k\Omega$		5	mV
$I_{os}V_o = +2V$ , $V_{cc} = +15V$		+40	$\mu A$
GBP	$R_L = 2k\Omega$ , $C_L = 100pF$	1.3	MHz
SR	$R_L = 2k\Omega$ , $C_L = 100pF$	0.4	V/ $\mu s$





PACKAGE MECHANICAL DATA  
14 PINS - PLASTIC DIP



Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
a1	0.51			0.020		
B	1.39		1.65	0.055		0.06
b		0.5			0.020	5
b1		0.25			0.010	
D			20			
E		8.5			0.335	0.78
e		2.54			0.100	7
e3		15.24			0.600	
F			7.1			
I			5.1			
L		3.3			0.130	
Z	1.27		2.54	0.050		0.28

0

0.20

1

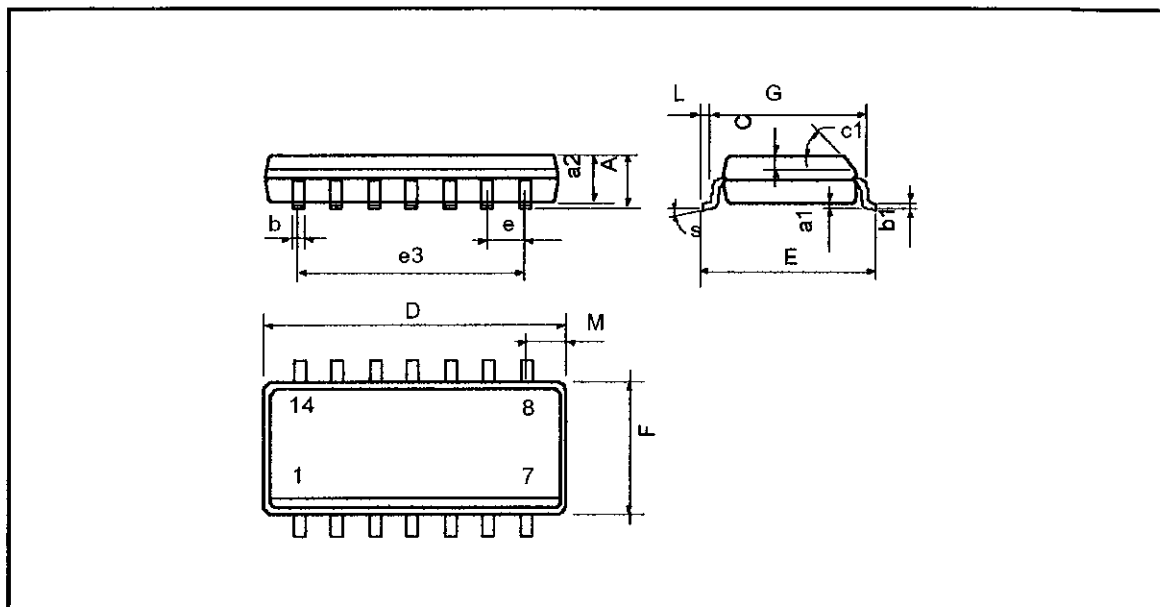
0.10

0



## PACKAGE MECHANICAL DATA

14 PINS - PLASTIC MICROPACKAGE (SO)



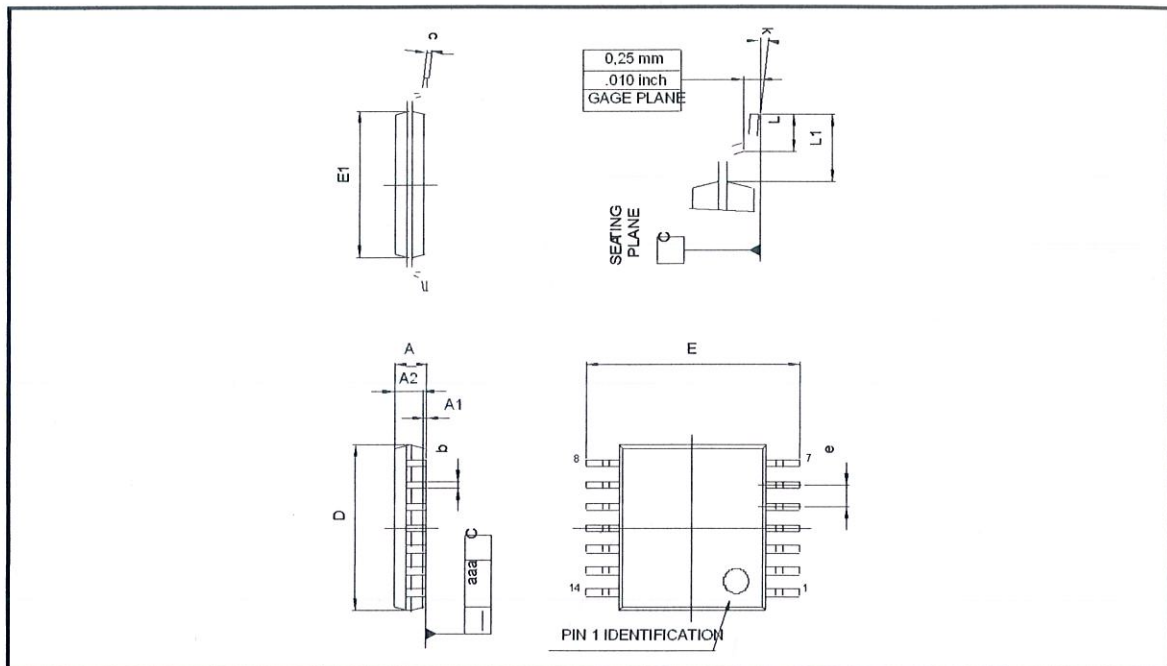
Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A			1.75			0.069
a1	0.1		0.2	0.004		0.008
a2			1.6			0.063
b	0.35		0.46	0.014		0.018
b1	0.19		0.25	0.007		0.010
C		0.5			0.020	
c1	45° (typ.)					
D (1)	8.55		8.75	0.336		0.344
E	5.8		6.2	0.228		0.244
e		1.27			0.050	
e3		7.62			0.300	
F (1)	3.8		4.0	0.150		0.157
G	4.6		5.3	0.181		0.208
L	0.5		1.27	0.020		0.050
M			0.68			0.027
S	8° (max.)					

Note: (1) D and F do not include mold flash or protrusions - Mold flash or protrusions shall not exceed 0.15mm (.006 inc) ONLY FOR DATA BOOK.



## PACKAGE MECHANICAL DATA

## 14 PINS - THIN SHRINK SMALL OUTLINE PACKAGE (TSSOP)



Dimensions	Millimeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A			1.20			0.05
A1	0.05		0.15	0.01		0.006
A2	0.80	1.00	1.05	0.031	0.039	0.041
b	0.19		0.30	0.007		0.15
c	0.09		0.20	0.003		0.012
D	4.90	5.00	5.10	0.192	0.196	0.20
E		6.40			0.252	
E1	4.30	4.40	4.50	0.169	0.173	0.177
e		0.65			0.025	
k	0°		8°	0°		8°
L	0.450	0.600	0.750	0.018	0.024	0.030
L1		1.00			0.039	
aaa			0.100			0.004

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

© The ST logo is a registered trademark of STMicroelectronics

© 2001 STMicroelectronics - Printed in Italy - All Rights Reserved

STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - Canada - China - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia  
Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States

© <http://www.st.com>







## 4. Datasheets and details of the ICs used

### 4.4. 8051

#### 8051: Types of Memory

The 8051 has three very general types of memory. To effectively program the 8051 it is necessary to have a basic understanding of these memory types

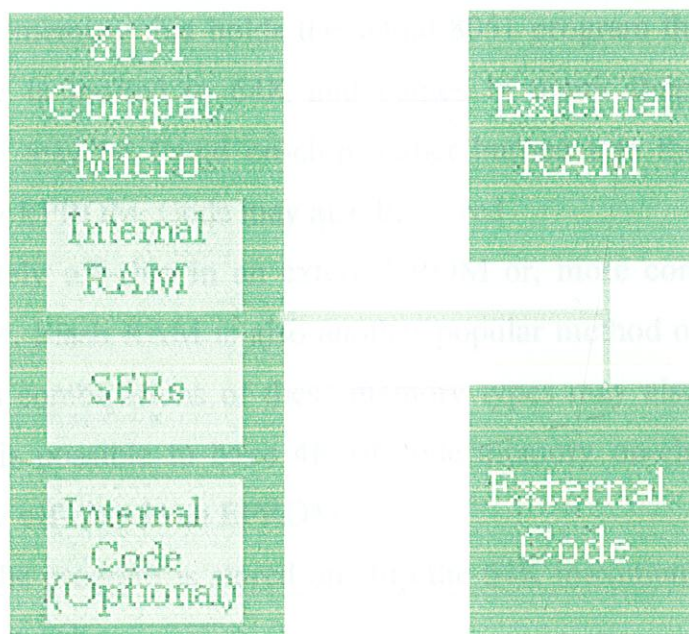


Figure 4.4.1

**On-Chip Memory** refers to any memory (Code, RAM, or other) that physically exists on the microcontroller itself. On-chip memory can be of several types,

1. **External Code Memory** is code (or program) memory that resides off-chip. This is often in the form of an external EPROM.
2. **External RAM** is RAM memory that resides off chip. This is often in the form of standard static RAM or flash RAM.

### **Code memory**

It is the memory that holds the actual 8051 program that is to be run. This memory is limited to 64K and comes in many shapes and sizes: Code memory may be found on-chip, either burned into the microcontroller as ROM or EPROM. Code may also be stored completely off-chip in an external ROM or, more commonly, an external EPROM. Flash RAM is also another popular method of storing a program. Various combinations of these memory types may also be used--that is to say, it is possible to have 4K of code memory on-chip and 64k of code memory off-chip in an EPROM.

When the program is stored on-chip the 64K maximum is often reduced to 4k, 8k, or 16k.

This varies depending on the version of the chip that is being used. Each version offers specific capabilities and one of the distinguishing factors from chip to chip is how much ROM/EPROM space the chip has. However, code memory is most commonly implemented as off-chip EPROM. This is especially true in low-cost development systems and in systems developed by students.

## **External RAM**

As an obvious opposite of Internal RAM, the 8051 also supports what is called External RAM. As the name suggests, External RAM is any random access memory which is found off-chip.

Since the memory is off-chip it is not as flexible in terms of accessing, and is also slower. For example, to increment an Internal RAM location by 1 requires only 1 instruction and 1 instruction cycle. To increment a 1-byte value stored in External RAM requires 4 instructions and 7 instruction cycles. In this case, external memory is 7 times slower!

What External RAM loses in speed and flexibility it gains in quantity. While Internal RAM is limited to 128 bytes the 8051 supports External RAM up to 64K.

## **On-Chip Memory**

8051 includes a certain amount of on-chip memory. On-chip memory is really one of two SFR) memory. The layout of the 8051's internal memory is presented in the following memory map:



IRAM Addr									Description
00	R0	R1	R2	R3	R4	R5	R6	R7	Reg. Bank 0
08	R0	R1	R2	R3	R4	R5	R6	R7	Reg. Bank 1
10	R0	R1	R2	R3	R4	R5	R6	R7	Reg. Bank 2
18	R0	R1	R2	R3	R4	R5	R6	R7	Reg. Bank 3
20	00	08	10	18	20	28	30	38	Bits 00-3F
28	40	48	50	58	60	68	70	78	Bits 40-7F
30	General User RAM & Stack Space (80 bytes, 30h-7Fh)								General IRAM
7F									
80	Special Function Registers (SFRs) (80h - FFh)								SFRs
:									
:									
:									

Figure 4.4.2

The 8051 has a bank of 128 bytes of Internal RAM. This Internal RAM is found on-chip on the 8051 so it is the fastest RAM available.

It is also the most flexible in terms of reading, writing and modifying it's contents. Internal RAM is volatile, so when the 8051 is reset this memory is cleared.

The 128 bytes of internal ram is subdivided as shown on the memory map. The first 8 bytes (00h - 07h) are "register bank 0". By manipulating certain SFRs, a program may choose to use register banks 1, 2, or 3. These alternative register banks are located in internal RAM in addresses 08h



through 1Fh. Bit memory actually resides in internal RAM, from addresses 20h through 2Fh.

The 80 bytes remaining of Internal RAM, from addresses 30h through 7Fh, may be used by user variables that need to be accessed frequently or at high-speed. This area is also utilized by the microcontroller as a storage area for the operating stack.

### **Register Banks**

The 8051 uses 8 "R" registers which are used in many of its instructions. These "R" registers are numbered from 0 through 7 (R0, R1, R2, R3, R4, R5, R6, and R7). These registers are generally used to assist in manipulating values and moving data from one memory location to another. For example, to add the value of R4 to the Accumulator, we would execute the following instruction:

**ADD A,R4**

However, as the memory map shows, the "R" Register R4 is really part of Internal RAM. Specifically, R4 is address 04h. This can be seen in the bright green section of the memory map. Thus the above instruction accomplishes the same thing as the following operation:

**ADD A,04h**

This instruction adds the value found in Internal RAM address 04h to the value of the Accumulator, leaving the result in the Accumulator. Since R4 is really Internal RAM 04h, the above instruction effectively accomplished the same thing.

The 8051 has four distinct register banks. When the 8051 is first booted up, register bank 0 (addresses 00h through 07h) is used by default. However, your program may instruct the 8051 to use one of the alternate register banks; i.e. register banks 1, 2, or 3. In this case, R4 will no longer be the same as Internal RAM address 04h.

For example, if your program instructs the 8051 to use register bank 3, "R" register R4 will now be synonymous with Internal RAM address 1Ch.

The concept of register banks adds a great level of flexibility to the 8051, especially when dealing with interrupts.

### **Bit Memory**

The 8051, being a communications-oriented microcontroller, gives the user the ability to access a number of bit variables. These variables may be either 1 or 0.

There are 128 bit variables available to the user, numbered 00h through 7Fh. The user may make use of these variables with commands such as SETB and CLR. It is important to note that Bit Memory is really a part of Internal RAM. In fact, the 128 bit

variables occupy the 16 bytes of Internal RAM from 20h through 2Fh. But since the 8051 provides special instructions to access these 16 bytes of memory on a bit by bit basis it is useful to think of it as a separate type of memory. However, always keep in mind that it is just a subset of Internal

RAM—and that operations performed on Internal RAM can change the values of the bit variables.

Bit variables 00h through 7Fh are for user-defined functions in their programs. However, bit variables 80h and above are actually used to access certain SFRs on a bit-by-bit basis. For example, if output lines P0.0 through P0.7 are all clear (0) and you want to turn on the P0.0 output line you may either execute:

```
MOV P0,#01h || SETB 80h
```

Both these instructions accomplish the same thing. However, using the SETB command will turn on the P0.0 line without effecting the status of any of the other P0 output lines. The MOV command effectively turns off all the other output lines which, in some cases, may not be acceptable.

### **Special Function Register (SFR) Memory**

Special Function Registers (SFRs) are areas of memory that control specific functionality of the 8051 processor. For example, four SFRs permit access to the 8051's 32 input/output lines.

Another SFR allows a program to read or write to the 8051's serial port. Other SFRs allow the user to set the serial baud rate, control and access timers, and configure the 8051's interrupt system.

When programming, SFRs have the illusion of being Internal Memory. For example, if you want to write the value "1" to Internal RAM location 50 hex you would execute the instruction:

```
MOV 50h,#01h
```



Similarly, if you want to write the value "1" to the 8051's serial port you would write this value to the SBUF SFR, which has an SFR address of 99 Hex. Thus, to write the value "1" to the serial port you would execute the instruction:

```
MOV 99h,#01h
```

As you can see, it appears that the SFR is part of Internal Memory. This is not the case. When using this method of memory access (it's called direct address), any instruction that has an address of 00h through 7Fh refers to an Internal RAM memory address; any instruction with an address of 80h through FFh refers to an SFR control register.

### **8051: SFRs**

The 8051 is a flexible microcontroller with a relatively large number of modes of operations. Your program may inspect and/or change the operating mode of the 8051 by manipulating the values of the 8051's Special Function Registers (SFRs).

SFRs are accessed as if they were normal Internal RAM. The only difference is that Internal RAM is from address 00h through 7Fh whereas SFR registers exist in the address range of 80h through FFh.

Each SFR has an address (80h through FFh) and a name. The



following chart provides a graphical presentation of the 8051's SFRs, their names, and their address.

80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	DPTR								F7
F8									FF

 Blue background are I/O port SFRs  
 Yellow background are control SFRs  
 Green background are other SFRs

Figure 4.4.3

As you can see, although the address range of 80h through FFh offer 128 possible addresses, there are only 21 SFRs in a standard 8051. All other addresses in the SFR range (80h through FFh) are considered invalid. Writing to or reading from these registers may produce undefined values or behavior.

### SFR Types

As mentioned in the chart itself, the SFRs that have a blue background are

SFRs related to the I/O ports. The 8051 has four I/O ports of 8 bits, for a total of 32 I/O lines. Whether a given I/O line is high or low and the value read from the line are controlled by the SFRs in green.

The SFRs with yellow backgrounds are SFRs which in some way control the operation or the configuration of some aspect of the 8051. For example, TCON controls the timers, SCON controls the serial port.

The remaining SFRs, with green backgrounds, are "other SFRs." These SFRs can be thought of as auxillary SFRs in the sense that they don't directly configure the 8051 but obviously the 8051 cannot operate without them. For example, once the serial port has been configured using SCON, the program may read or write to the serial port using the SBUF register.

### **SFR Descriptions**

This section will endeavor to quickly overview each of the standard SFRs found in the above SFR chart map. It is not the intention of this section to fully explain the functionality of each SFR--this information will be covered in separate

chapters of the tutorial. This section is to just give you a general idea of what each SFR does.

#### **P0 (Port 0, Address 80h, Bit-Addressable):**

This is input/output port 0. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 0 is pin P0.0, bit 7 is

pin P0.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

#### **SP (Stack Pointer, Address 81h):**

This is the stack pointer of the microcontroller. This SFR indicates where the next value to be taken from the stack will be read from in Internal RAM. If you push a value onto the stack, the value will be written to the address of  $SP + 1$ . That is to say, if SP holds the value 07h, a PUSH instruction will push the value onto the stack at address 08h. This SFR is modified by all instructions which modify the stack, such as PUSH, POP, RET, RETI, and whenever interrupts are provoked by the microcontroller.

#### **DPL/DPH (Data Pointer Low/High, Addresses 82h/83h):**

The SFRs DPL and DPH work together to represent a 16-bit value called the Data Pointer. The data pointer is used in operations regarding external RAM and some instructions involving code memory. Since it is an unsigned two-byte integer value, it can represent values from 0000h to FFFFh (0 through 65,535 decimal).

#### **PCON (Power Control, Addresses 87h):**

The Power Control SFR is used to control the 8051's power control modes. Certain operation modes of the 8051 allow the 8051 to go into a type of "sleep" mode which requires much less power. These modes of operation are controlled through PCON. Additionally, one of the bits in PCON is used to double the effective baud rate of the 8051's serial port.

**TCON (Timer Control, Addresses 88h, Bit-Addressable):**

The Timer Control SFR is used to configure and modify the way in which the 8051's two timers operate. This SFR controls whether each of the two timers is running or stopped and contains a flag to indicate that each timer has overflowed. Additionally, some non-timer related bits are located in the TCON SFR. These bits are used to configure the way in which the external interrupts are activated and also contain the external interrupt flags which are set when an external interrupt has occurred.

**TMOD (Timer Mode, Addresses 89h):**

The Timer Mode SFR is used to configure the mode of operation of each of the two timers. Using this SFR your program may configure each timer to be a 16-bit timer, an 8-bit autoreload timer, a 13-bit timer, or two separate timers. Additionally, you may configure the timers to only count when an external pin is activated or to count "events" that are indicated on an external pin.

**TL0/TH0 (Timer 0 Low/High, Addresses 8Ah/8Bh):**

These two SFRs, taken together, represent timer 0. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is configurable is how and when they increment in value.

**TL1/TH1 (Timer 1 Low/High, Addresses 8Ch/8Dh):**



These two SFRs, taken together, represent timer 1. Their exact behavior depends on how the timer is configured in the TMOD SFR; however, these timers always count up. What is configurable is how and when they increment in value.

**P1 (Port 1, Address 90h, Bit-Addressable):**

This is input/output port 1. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 1 is pin P1.0, bit 7 is pin P1.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

**SCON (Serial Control, Addresses 98h, Bit-Addressable):**

The Serial Control SFR is used to configure the behavior of the 8051's on-board serial port. This SFR controls the baud rate of the serial port, whether the serial port is activated to receive data, and also contains flags that are set when a byte is successfully sent or received.

**SBUF (Serial Control, Addresses 99h):**

The Serial Buffer SFR is used to send and receive data via the on-board serial port. Any value written to SBUF will be sent out the serial port's TXD pin. Likewise, any value which the 8051 receives via the serial port's RXD pin will be delivered to the user program via SBUF. In other words, SBUF

serves as the output port when written to and as an input port when read from.

#### **P2 (Port 2, Address A0h, Bit-Addressable):**

This is input/output port 2. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 2 is pin P2.0, bit 7 is pin P2.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

#### **IE (Interrupt Enable, Addresses A8h):**

The Interrupt Enable SFR is used to enable and disable specific interrupts. The low 7 bits of the SFR are used to enable/disable the specific interrupts, where as the highest bit is used to enable or disable ALL interrupts. Thus, if the high bit of IE is 0 all interrupts are disabled regardless of whether an individual interrupt is enabled by setting a lower bit.

#### **P3 (Port 3, Address B0h, Bit-Addressable):**

This is input/output port 3. Each bit of this SFR corresponds to one of the pins on the microcontroller. For example, bit 0 of port 3 is pin P3.0, bit 7 is pin P3.7. Writing a value of 1 to a bit of this SFR will send a high level on the corresponding I/O pin whereas a value of 0 will bring it to a low level.

**IP (Interrupt Priority, Addresses B8h, Bit-Addressable):**

The Interrupt Priority SFR is used to specify the relative priority of each interrupt. On the 8051, an interrupt may either be of low (0) priority or high (1) priority. An interrupt may only interrupt interrupts of lower priority. For example, if we configure the 8051 so that all interrupts are of low priority except the serial interrupt, the serial interrupt will always be able to interrupt the system, even if another interrupt is currently executing. However, if a serial interrupt is executing no other interrupt will be able to interrupt the serial interrupt routine since the serial interrupt routine has the highest priority.

**PSW (Program Status Word, Addresses D0h, Bit-Addressable):**

The Program Status Word is used to store a number of important bits that are set and cleared by 8051 instructions. The PSW SFR contains the carry flag, the auxiliary carry flag, the overflow flag, and the parity flag. Additionally, the PSW register contains the register bank select flags which are used to select which of the "R" register banks are currently selected.

**8051: Basic Registers****The Accumulator**

The Accumulator, as its name suggests, is used as a general register to accumulate the results of a large number of instructions. It can hold an 8-bit (1-byte) value and is the most versatile register the 8051 has due to the sheer

number of instructions that make use of the accumulator. More than half of the 8051's 255 instructions manipulate or use the accumulator in some way. If you want to add the number 10 and 20, the resulting 30 will be stored in the Accumulator. Once you have a value in the Accumulator you may continue processing the value or you may store it in another register or in memory.

### **The "R" registers**

The "R" registers are a set of eight registers that are named R0, R1, etc. up to and including R7. These registers are used as auxiliary registers in many operations. To continue with the above example, perhaps you are adding 10 and 20. The original number 10 may be stored in the Accumulator whereas the value 20 may be stored in, say, register R4. To process the addition we would execute the command:

```
ADD A,R4
MOV A,R3
ADD A,R4
MOV R5,A
MOV A,R1
ADD A,R2
SUBB A,R5
```

```
;Move the value of R3 into the accumulator
;Add the value of R4
;Store the resulting value temporarily in R5
```



;Move the value of R1 into the accumulator  
;Add the value of R2  
;Subtract the value of R5 (which now contains  $R3 + R4$ )

As we can see, we used R5 to temporarily hold the sum of R3 and R4. This isn't the most efficient way to calculate.

Accumulator will contain the value 30.

You may think of the "R" registers as very important auxillary, or "helper", registers. The Accumulator alone would not be very useful if it were not for these "R" registers. The "R" registers are also used to temporarily store values. For example, let's say you want to add the values in R1 and R2 together and then subtract the values of R3 and R4. One way to do this would be:

$(R1+R2) - (R3 + R4)$  but it does illustrate the use of the "R" registers as a way to store values temporarily.

### **The "B" Register**

The "B" register is very similar to the Accumulator in the sense that it may hold an 8-bit (1-byte) value. The "B" register is only used by two 8051 instructions: MUL AB and DIV AB. Thus, if you want to quickly and easily multiply or divide A by another number, you may store the other number in "B" and make use of these two instructions. Aside from the MUL and DIV instructions, the "B" register is often used as yet another temporary storage register much like a ninth "R" register.

## **The Data Pointer (DPTR)**

The Data Pointer (DPTR) is the 8051's only user-accessible 16-bit (2-byte) register. The Accumulator, "R" registers, and "B" register are all 1-byte values. DPTR, as the name suggests, is used to point to data. It is used by a number of commands which allow the 8051 to access external memory. When the 8051 accesses external memory it will access external memory at the address indicated by DPTR.

While DPTR is most often used to point to data in external memory, many programmers often take advantage of the fact that it's the only true 16-bit register available. It is often used to store 2-byte values which have nothing to do with memory locations.

## **THE PROGRAM COUNTER (PC):**

The Program Counter (PC) is a 2-byte address which tells the 8051 where the next instruction to execute is found in memory. When the 8051 is initialized PC always starts at 0000h and is incremented each time an instruction is executed. It is important to note that PC isn't always incremented by one. Since some instructions require 2 or 3 bytes the PC will be incremented by 2 or 3 in these cases. The Program Counter is special in that there is no way to directly modify its value. That is to say, you can't do something like PC=2430h.

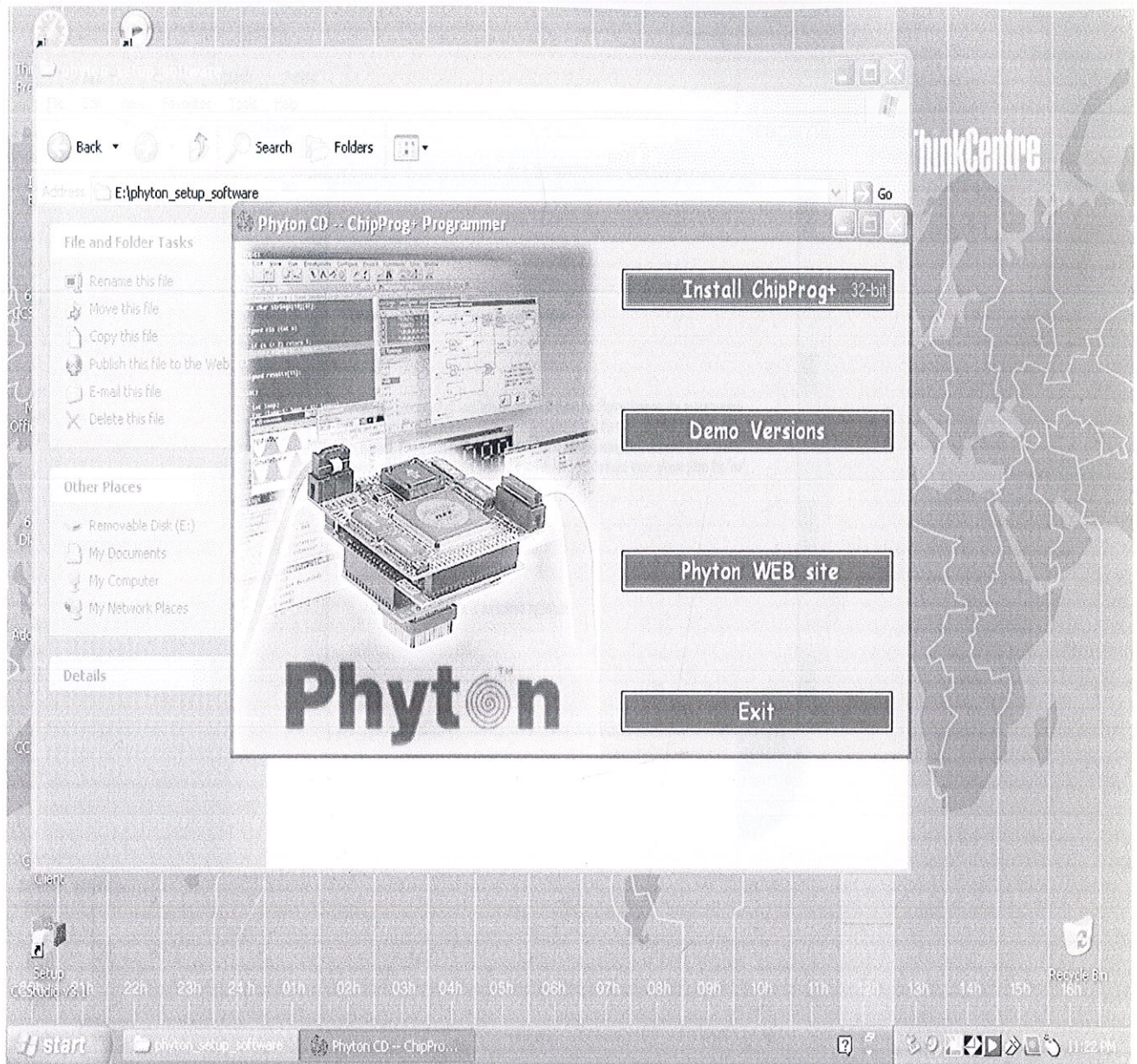
## **THE STACK POINTER (SP):**

The Stack Pointer, like all registers except DPTR and PC, may hold an 8-bit (1-byte) value. The Stack Pointer is used to indicate where the next value to be removed from the stack should be taken from. When you push a value onto the stack, the 8051 first increments the value of SP and then stores the value at the resulting memory location. When you pop a value off the stack, the 8051 returns the value from the memory location indicated by SP, and then decrements the value of SP. This order of operation is important. When the 8051 is initialized SP will be initialized to 07h.



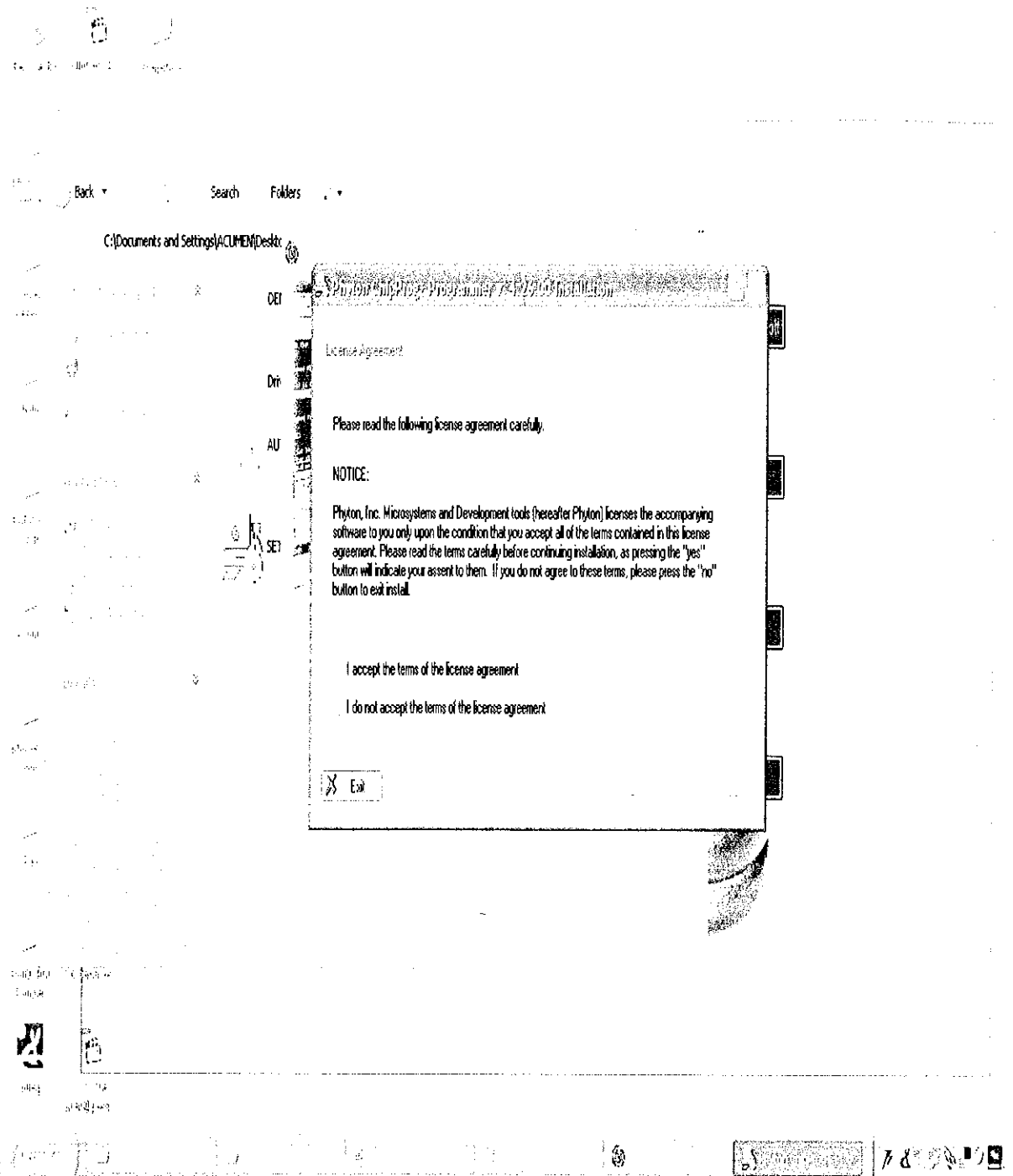


## 4.5. INSTALLATION GUIDE FOR THE PROGRAMMER KIT



### STEP 1:

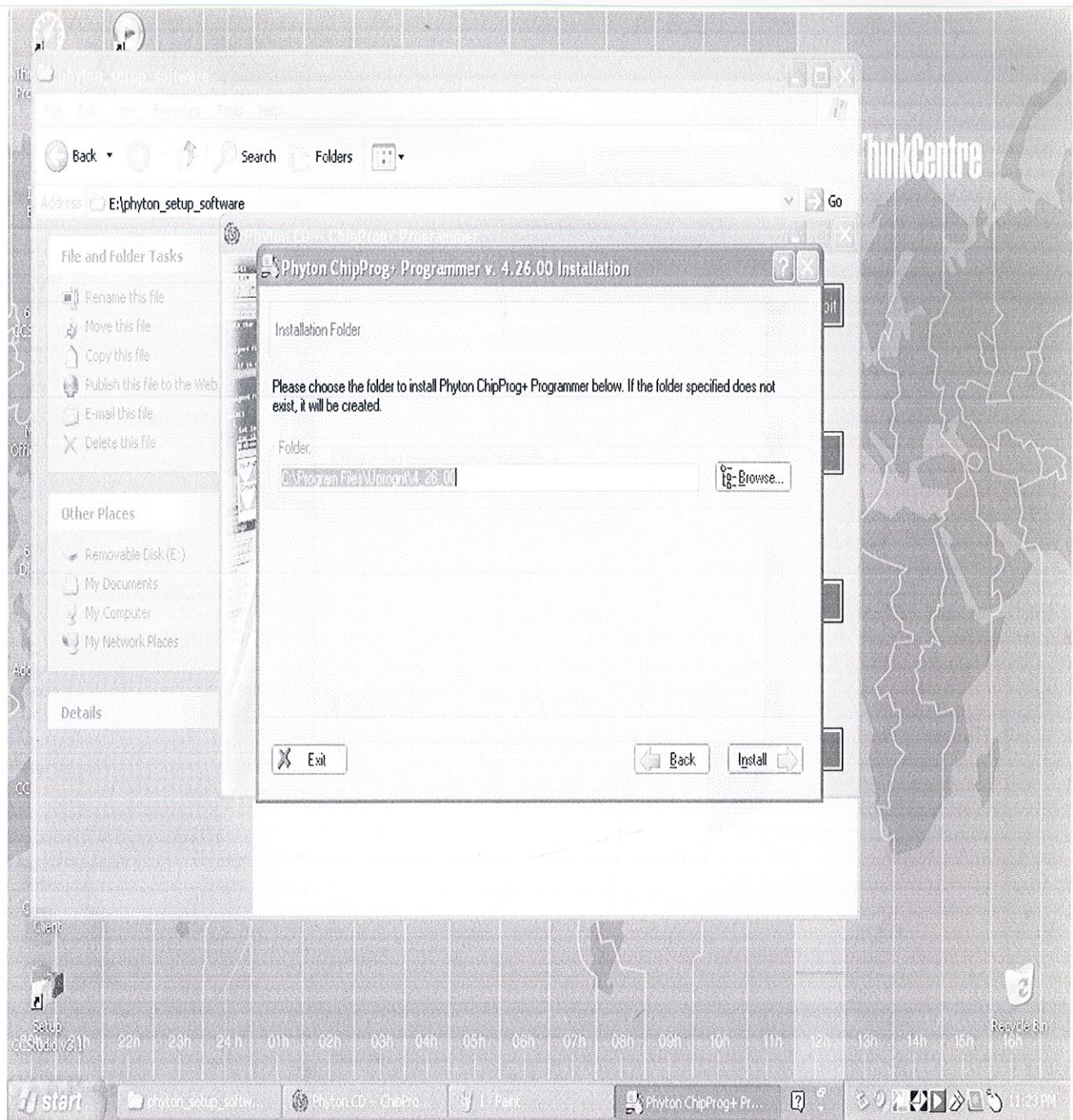
Insert the Compact Disk into the CD-ROM. You will have the above page popped up for you as soon as you insert the CD. Click on the first option to continue with the installation i.e. Install ChipProg+.



## STEP 2:

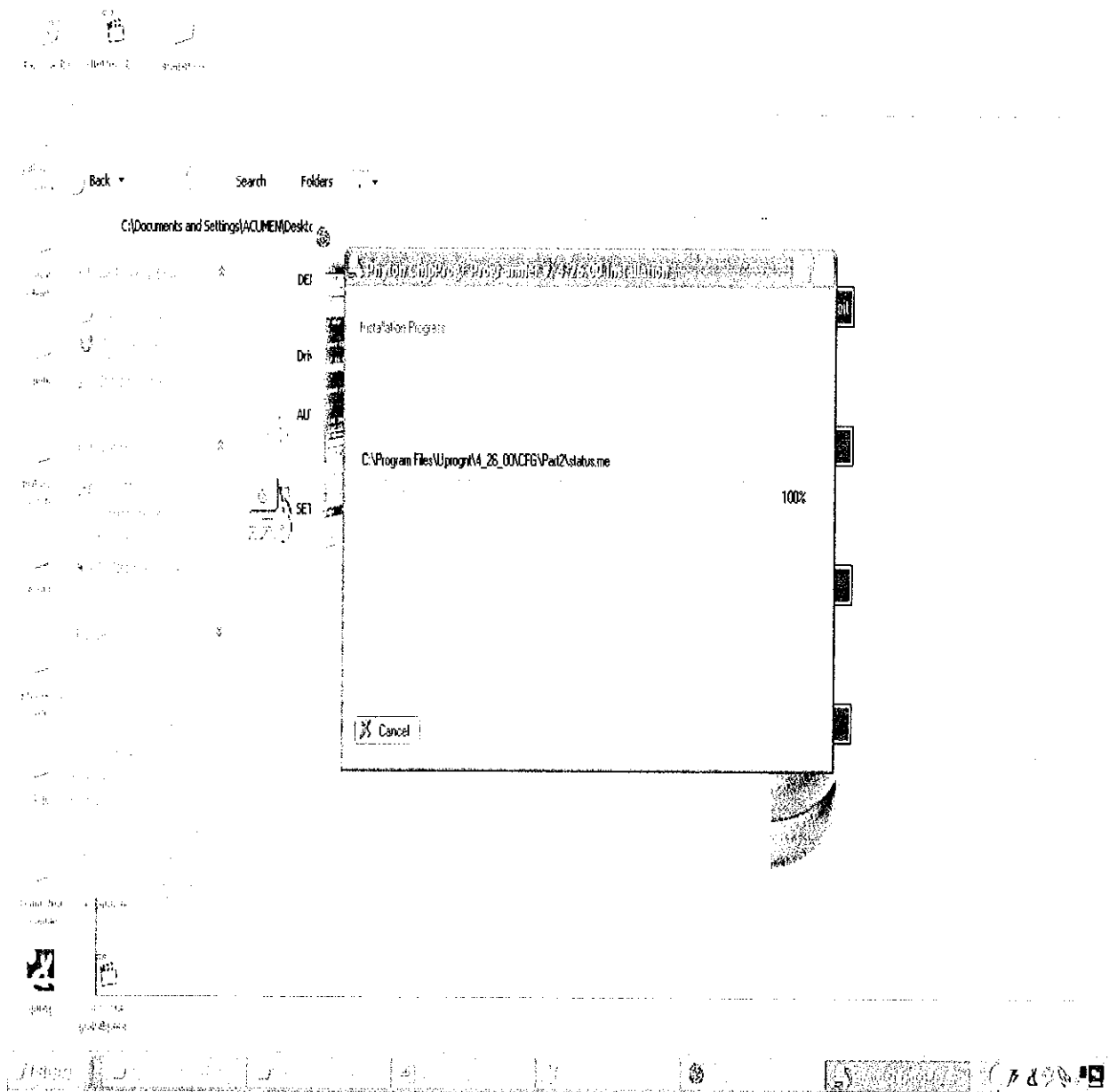
Carefully read and accordingly accept or reject the License agreement specified by the Phyton Inc. Microsystems. If you assert with them select the first check box i.e. accept to continue the installation else if you disagree you can stop the installation at that very moment.





### STEP 3:

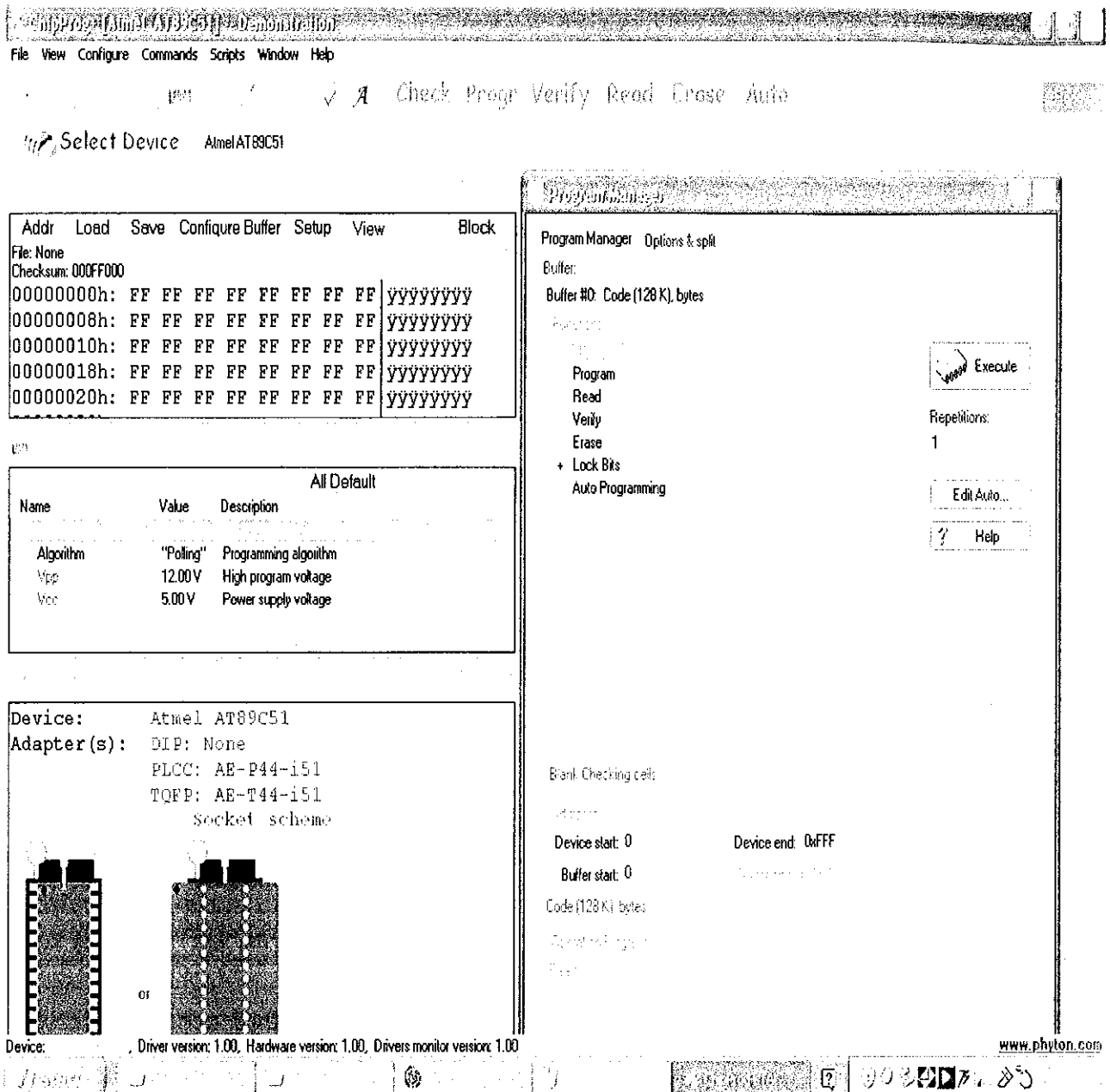
Select the folder where you want to install Phyton ChipProg+ programmer. If folder specified does not exist then it will be created. Browse if you want to change the default path.



#### **STEP 4:**

Wait till the automatic installation continues. If you wish to stop installation in between click on cancel.





## STEP 5:

Start→All Programs→Python ChipProg+ Programmer 4.26.00→Phyton ChipProg+  
to open the following frame in order to load the test file. Select the device i.e. the chip to be programmed from the drop down box. The Socket Scheme specifies the way to insert the chip into the programmer. Insert the chip into the programmer.

ChipPro - Atmel AT89C51 Demonstration

File View Configure Commands Scripts Window Help

Atmel AT89C51

```

00000010h: FF FF FF FF FF FF FF FF YYYYYYYY
00000018h: FF FF FF FF FF FF FF FF YYYYYYYY
00000020h: FF FF FF FF FF FF FF FF YYYYYYYY
  
```

Blank check  
Program  
Read  
Verify  
Erase  
+ Lock Bits  
Auto Programming

Stop

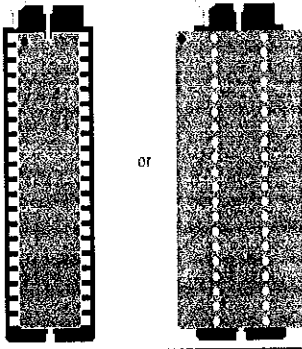
Repetitions:  
1

Edit Auto...

? Help

Name	Value	Description
Algorithm	"Polling"	Programming algorithm
Vpp	12.00V	High program voltage
Vcc	5.00V	Power supply voltage

Device: Atmel AT89C51  
Adapter(s): DIP: None  
              PLCC: AB-P44-i51  
              TQFP: AB-T44-i51  
              Socket scheme



or

Blank Checking cells

Device start: 0      Device end: 0xFFFF  
Buffer start: 0      Program 0xFFFF

Code (128 K): bytes

Download

Read

Checksum

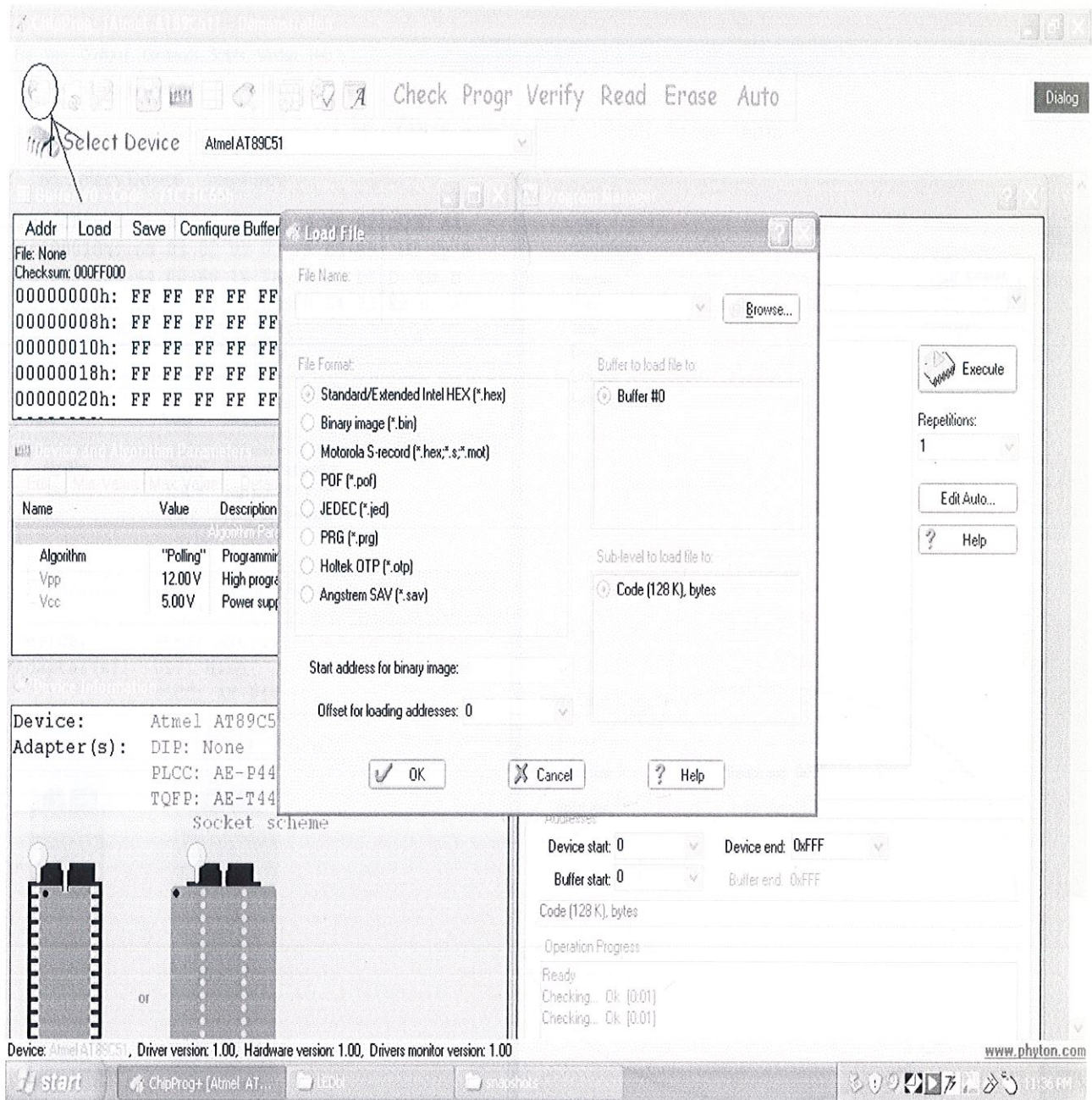
Erasing

Device: , Driver version: 1.00, Hardware version: 1.00, Drivers monitor version: 1.00

www.phyton.com

## STEP 6:

Click on "check" tab in order to check whether the chip is empty or not. If not erase it via the "erase" tab.

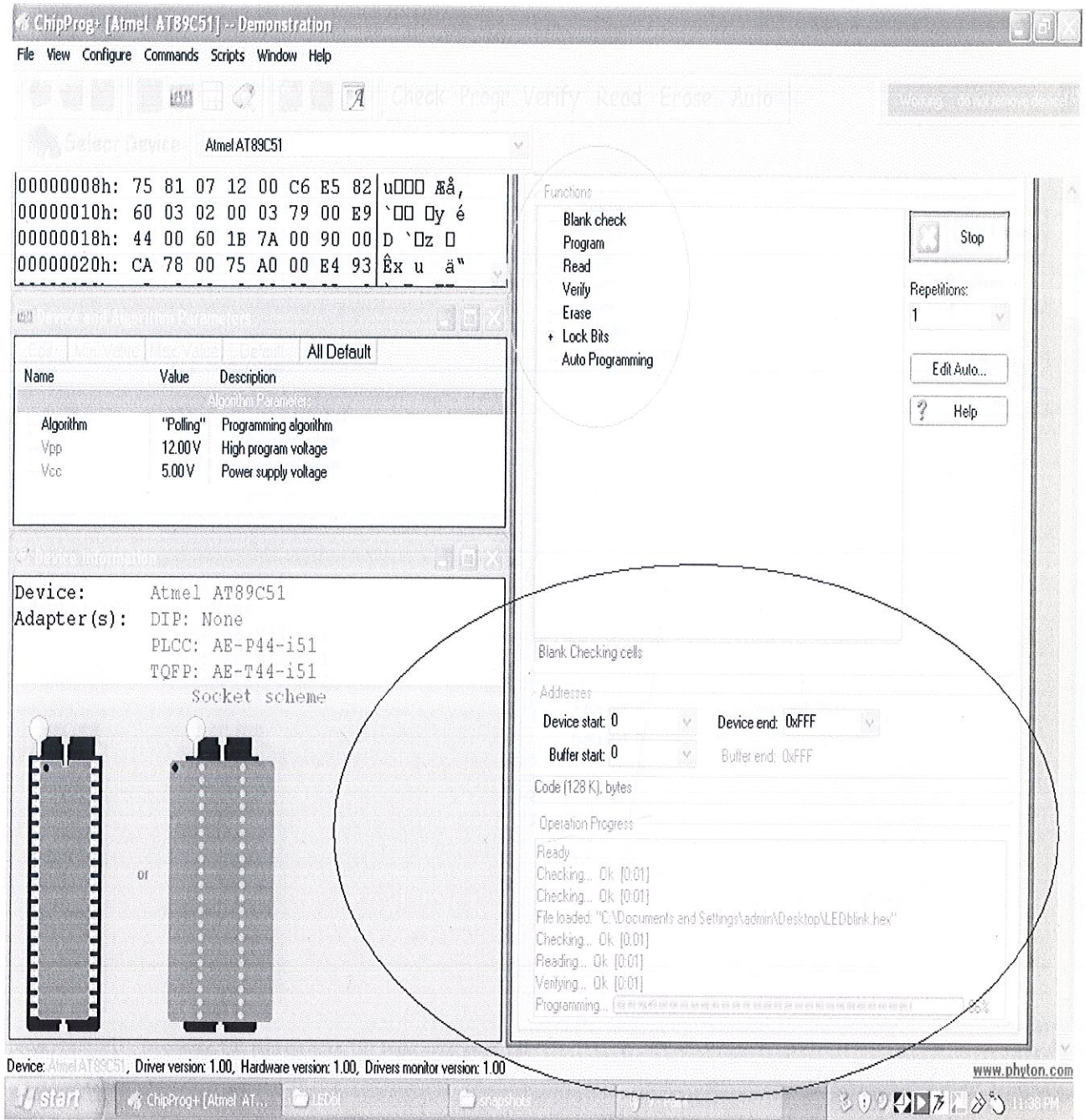


## STEP 7:

Click on the button highlighted by the ellipse in order to load your file onto the microcontroller. Select the file format that you wish to load on the microcontroller and specify the offset for loading address. Click on "OK" once you are through with the selection.







## STEP 9:

Click on execute in order to perform the list of functions specified in the red elliptical portion. While the program is being burned we see its progress in the Operation Progress specified in the black elliptical portion.

ChipProg AT89C51 Demonstration

File View Configure Commands Scripts Window Help

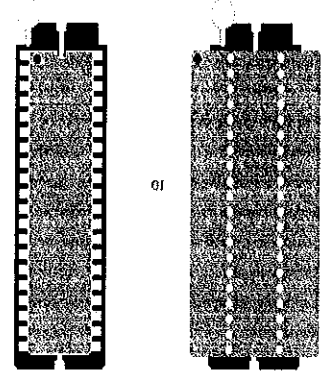
Check Progr Verify Read Erase Auto

Select Device Atmel AT89C51

00000008h: 75 81 07 12 00 C6 E5 82 u u u u Å Å,  
 00000010h: 60 03 02 00 03 79 00 E9 ` u u y é  
 00000018h: 44 00 60 1B 7A 00 90 00 D ` Qz u  
 00000020h: CA 78 00 75 A0 00 E4 93 Ê x u ä"

Name	Value	Description
Algorithm	"Polling"	Programming algorithm
Vpp	12.00V	High program voltage
Vcc	5.00V	Power supply voltage

Device: Atmel AT89C51  
 Adapter(s): DIP: None  
 PLCC: AE-P44-i51  
 TQFP: AE-T44-i51  
 Socket scheme



Blank: Checking cells

Addresses:

Device start: 0 Device end: 0xFFFF

Buffer start: 0 Buffer end: 0xFFFF

Code (128 K): bytes

Operation Progress:

Ready: 0% (0/1)

Checking: 0% (0/1)

Erasing: 0% (0/1)

Reading: 0% (0/1)

Verifying: 0% (0/1)

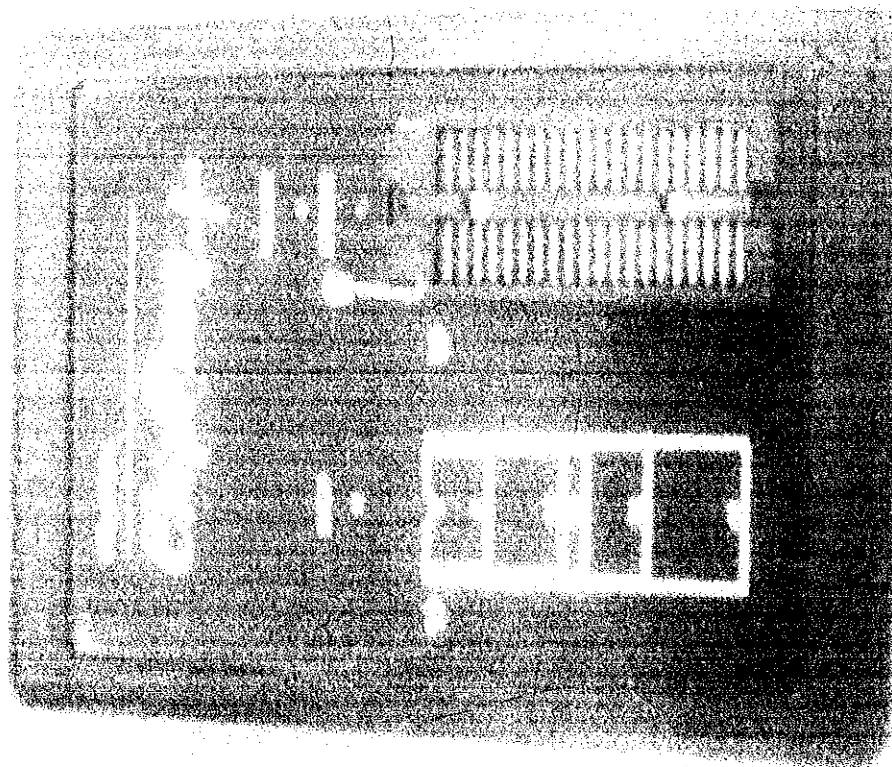
Programming: 0% (0/1)

Device: Driver version: 1.00, Hardware version: 1.00, Drivers monitor version: 1.00

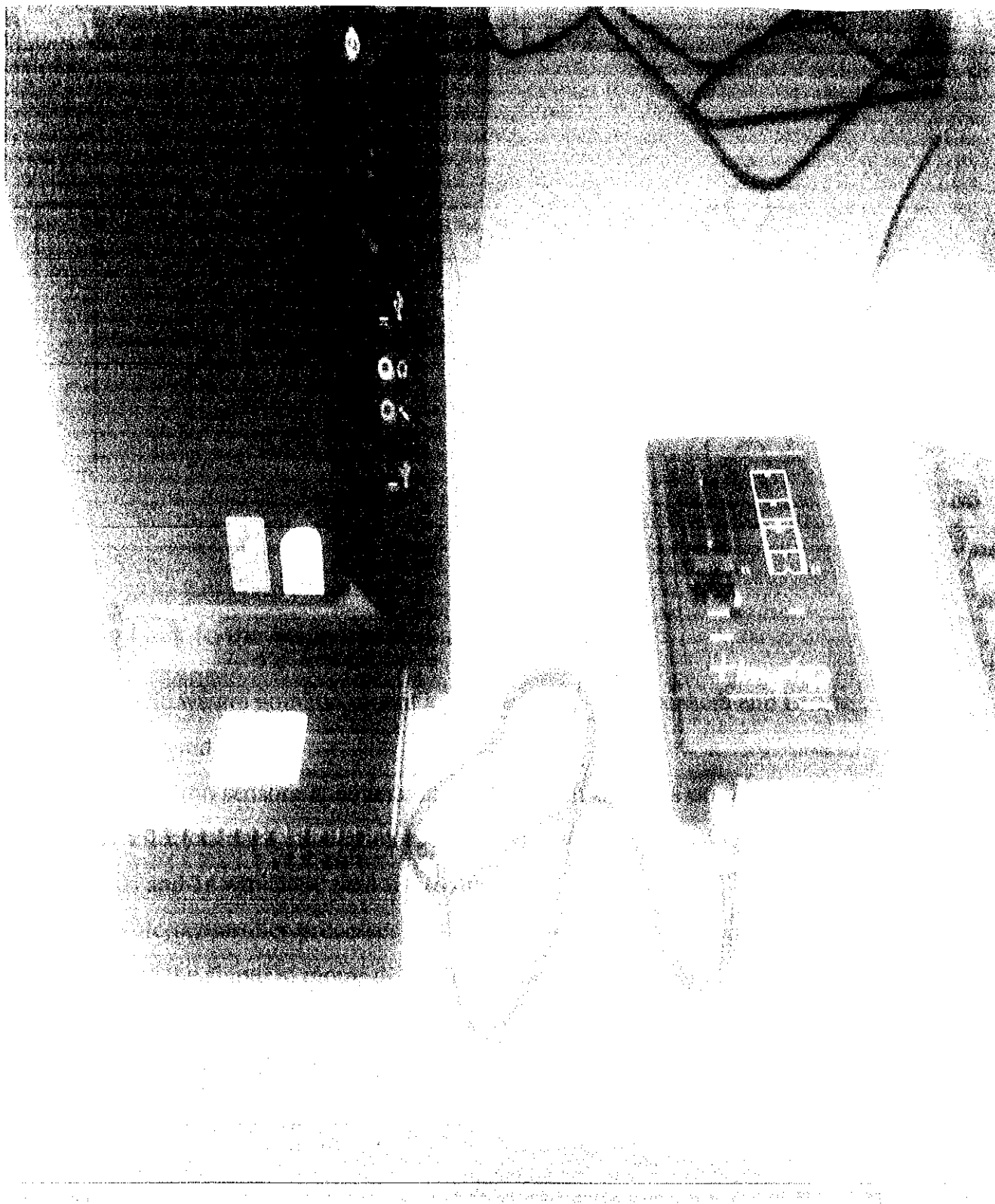
www.phyton.com

## STEP 10:

Prior to the burning we get the message accordingly specifying the operation as successful or failure and if failure then specifying the reason as highlighted.



**Python ChipProg+ kit used for burning the Microcontroller**



**Kit connected to the serial port of the computer.**



## 5. THE ALGORITHM FOR LINE FOLLOWER

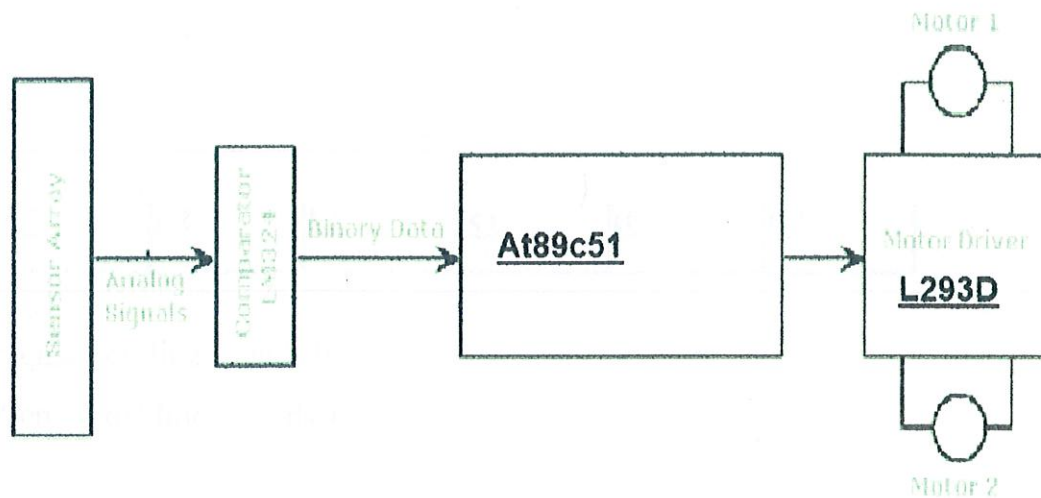


Figure 5..1

Coming back to the basic block diagram.

In our vehicle we are going to mount a Sensor Array- 6 IR sensors and LEDs facing the ground.

The output of the sensors is an analog signal which depends on the amount of light reflected back, this analog signal is given to the comparator to produce 0s and 1s which are then fed to the  $\mu\text{C}$ .

Now the microcontroller processes this output produced by the sensor array and provides the signal to motor driver to control the motion of the vehicle.

## 5.1. ALGORITHM

### Sensor array:

<u>L3</u>	<u>L2</u>	<u>L1</u>	<u>R1</u>	<u>R2</u>	<u>R3</u>
-----------	-----------	-----------	-----------	-----------	-----------

Sensor on line - reads 0

Sensor off line – reads 1

### Ideal position:

When the line is located at the centre of the array

<b>L3</b>	<b>L2</b>	<b>L1</b>	<b>R1</b>	<b>R2</b>	<b>R3</b>
1	1	0	0	1	1

So the microcontroller decides the next move to position the robot such that L1 and R1 reads 0 and the rest reads 1

Let,  $LV$ =leftmost sensor which reads 0,

$RV$ =rightmost sensor which reads 0.

If no sensor on left (or right) is 0, then  $LV$  (or  $RV$ )=0.

**For example:**

L3	L2	L1	R1	R2	R3
1	0	0	1	1	1

LV=2

RV=4

*If all sensor read 1,, then we are encountered with a condition of NO LINE.*

*else,*

*if LV>RV, move right*

*if LV<RV, move left*

*if LV=RV, move foward*

*then again read the sensor data , and perform the steps again.*

Now in case of NO LINE condition, the success of finding the line is more or less a matter of fluke.

**There are two options in case of no line:**

**Option 1:**

- Move Clockwise if line was last seen on Right
- Move Counter Clockwise if line was last seen on Left

Until line is found.

But as we can suggest this way is not very reliable

**Option 2:**

mentioned in section 5.1.A

## 5.1.A HYBRIDIZATION THROUGH HUMAN INSIGHT

### Option 2:

*Human interference, which basically mean enlightening the robot with human knowledge of the field in which line is located.*

*Now, we proposed a hybrid of the two options, in which the robot would follow option one for a fixed amount of time, and if during that period the robot is not able to locate the line, then it would take the input for human controller in the form of the probable direction of the line, and then move in that direction.*

*In our model the proximity of this human controller to the robot doesn't matter, because the input would be given through a GSM network.*

*We also propose to mount a GPS receiver on the robot, and this receiver would converge the 3D location information of the vehicle in the form of longitude, latitude and altitude.*

*This information would be sent to the human controller through the GSM network, using the 16 distinct DTMF tones.*

*And the human controller will then analyze the position of the vehicle through the received coordinates, compare the previously received position coordinated and will try to manually determine the nearest coordinates coinciding with the lost path.*

*Controller will then accordingly guide the robot in the direction of the line by controlling the direction of its motion by pressing the appropriate buttons for producing the desired DTMF tone.*



### 5.1.B. HYBRIDIZATION THROUGH ALGORITHM DESIGN

*One more thing that adds to hybrid nature of this vehicle is its ability of the algorithm to independently determine the degree of curvature of the line and then automatically making a decision as to which mode of turning to adopt.*

*Whether to take a normal turn or a sharp turn*

*In normal a left turn the left wheels will be activated by the forward current to the left motor and right wheels will be deactivated.*

*But in a sharp left turn the left wheels are activated in forward direction and the right wheels in backward direction.*

*Therefore the chances of losing the line while traversing due to any unexpected sharp curvatures in the line would be reduced further by making these amendments to the algorithm.*

*This can be implemented in the algorithm design by comparing the current sensor data with the previous sensor data and if there is a change of more than one sensor value in a single iteration then calling the algorithm block for sharp turn, otherwise calling the block for normal turn.*

## 5.2. PROPOSED CODE FOR LINE FOLLOWER IN C:

```
#include<at89x51.h> //microcontroller header file for at89xxx
```

```
sensor array configuration
```

```
#define R3 P0_7
```

```
#define R2 P0_6
```

```
#define R1 P0_5
```

```
#define L1 P0_4
```

```
#define L2 P0_3
```

```
#define L3 P0_2
```

```
//L293D configuration
```

```
#define EN12 P2_0
```

```
#define EN34 P2_1
```

```
#define IN1 P2_2
```

```
#define IN2 P2_3
```

```
#define IN3 P2_4
```

```
#define IN4 P2_5
```

```
// IR led configuration
```

```
#define R1led P1_0
```

```
#define R2led P1_1
```

```
#define R3led P1_2
```

```
#define L1led P1_3
```

```
#define L2led P1_4
```

```
#define L3led P1_5
```

```
#define LV 0 //leftmost sensor that reads 0
```

```
#define RV 0 //rightmost sensor that reads 0
```

```
void delay(int);
```

```
void time1ms();
```

```
void forward(void);
```

```
void reverse(void);
```

```
void turnl(void);
```

```
void turnr(void);  
void brake(void);
```

```
//main program sensor detection and respective movements
```

```
void main (void)
```

```
{  
P1=0x00;\\configuring port 1 for output  
P1_0=1;  
P1_1=1;  
P1_2=1;  
P1_3=1;  
P1_4=1;  
P1_5=1;
```

```
P0=0xFF;\\configuring port 0 for input
```

```
forward();  
delay(3);
```

```
while(1)  
{
```

```
if(P0_4==0)  
LV=1;  
else if(P0_3==0)  
LV=2;  
else if(P0_2==0)  
LV=3;  
else if(P0_5==0)  
LV=4;  
else if(P0_6==0)  
LV=5;  
else if(P0_7==0)  
LV=6;  
else  
LV=0;
```

```

if(P0_7==0)
RV=1;
else if(P0_6==0)
RV=2;
else if(P0_5==0)
RV=3;
else if(P0_2==0)
RV=4;
else if(P0_3==0)
RV=5;
else if(P0_4==0)
RV=6;
else
RV=0;

if(LV==3 && RV==3)
{
forward();
delay(3);
}

else if(LV<RV)
{
turnl();
delay(3);
}

else if(LV>RV)
{
turnr();
delay(3);
}

/*else if(LV==0 && RV==0)
{
no line code, human interfacing through GSM
}*/

}

```



```
}
```

```
void forward(void)
```

```
{  
  EN12=1;  
  EN34=1;  
  IN1=1;  
  IN2=0;  
  IN3=1;  
  IN4=0;  
}
```

```
void reverse(void)
```

```
{  
  EN12=1;  
  EN34=1;  
  IN1=1;  
  IN2=0;  
  IN3=1;  
  IN4=0;  
}
```

```
void turnl(void)
```

```
{  
  EN12=1;  
  EN34=1;  
  IN1=0;  
  IN2=1;  
  IN3=1;  
  IN4=0;  
}
```

```
void turnr(void)
```

```
{  
  EN12=1;  
  EN34=1;  
  IN1=1;  
  IN2=0;  
  IN3=0;  
  IN4=1;  
}
```

```
void brake(void)
```

```
{  
  EN12=0;  
  EN34=0;  
}
```

```
void delay(int n) /* do nothing n*1ms */
```

```
{  
  int i;  
  for (i=0; i< n ; i++)  
    time1ms();  
}
```

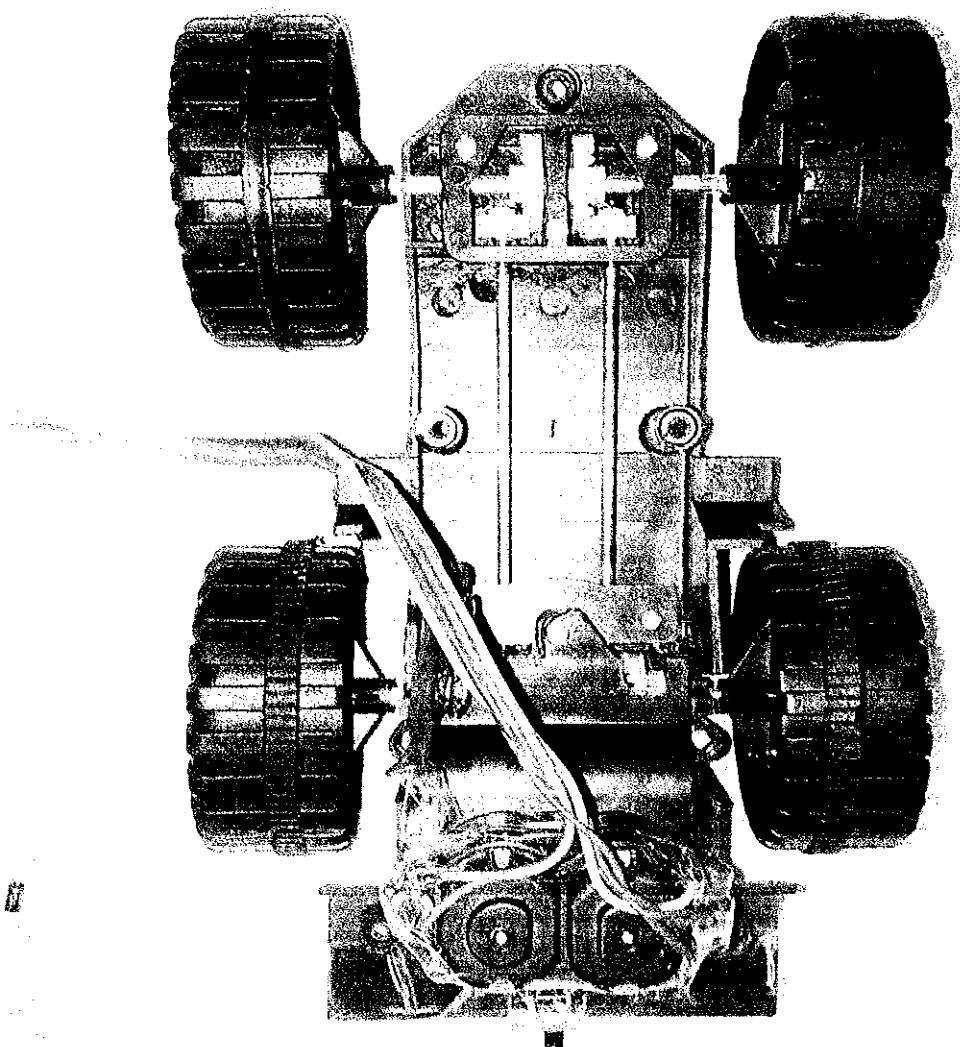
```
void time1ms() /* 1 ms delay with XTAL 11.0592MHz */
```

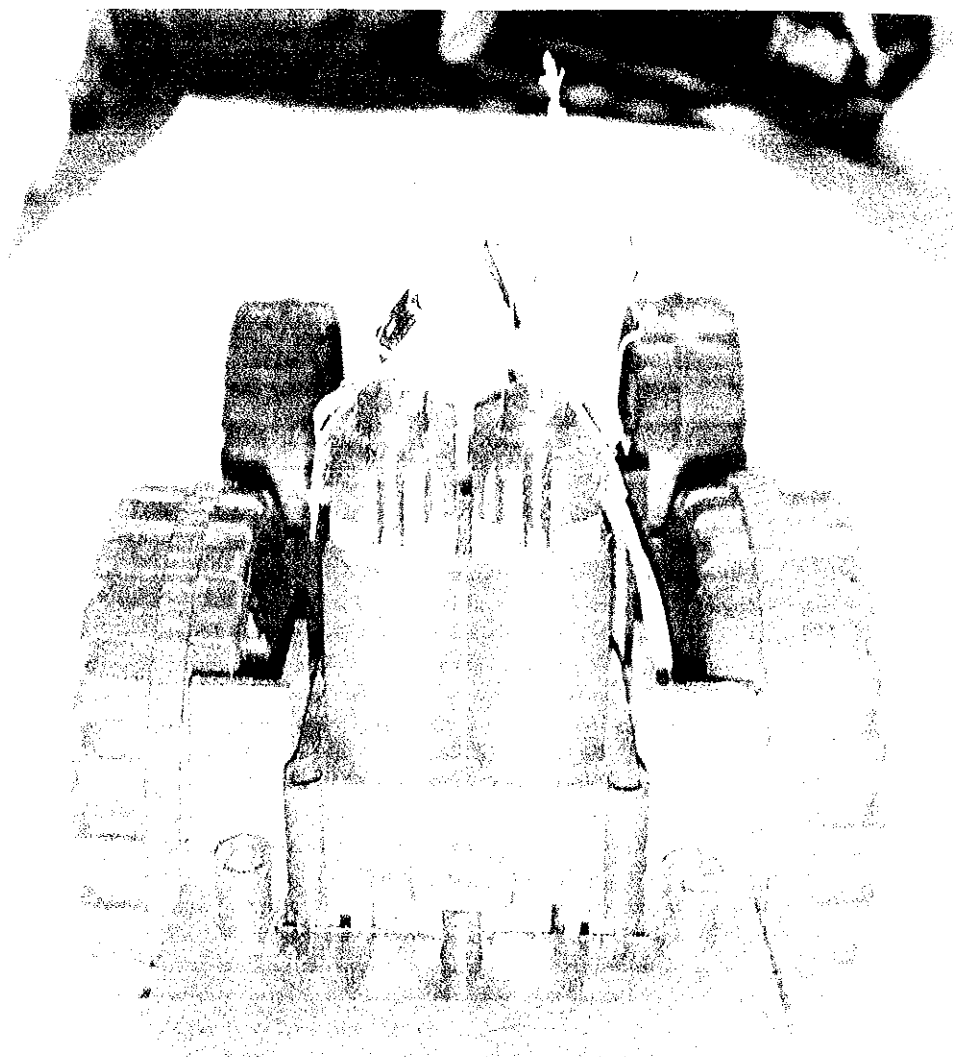
```
{  
  int i;  
  for (i = 0; i<1275; i++); // the value shown in this line, 1275 was
```

```
calibrated for 1ms  
// you may change it!  
}
```

## 6.ASSEMBLY OF ALL THE MODULES

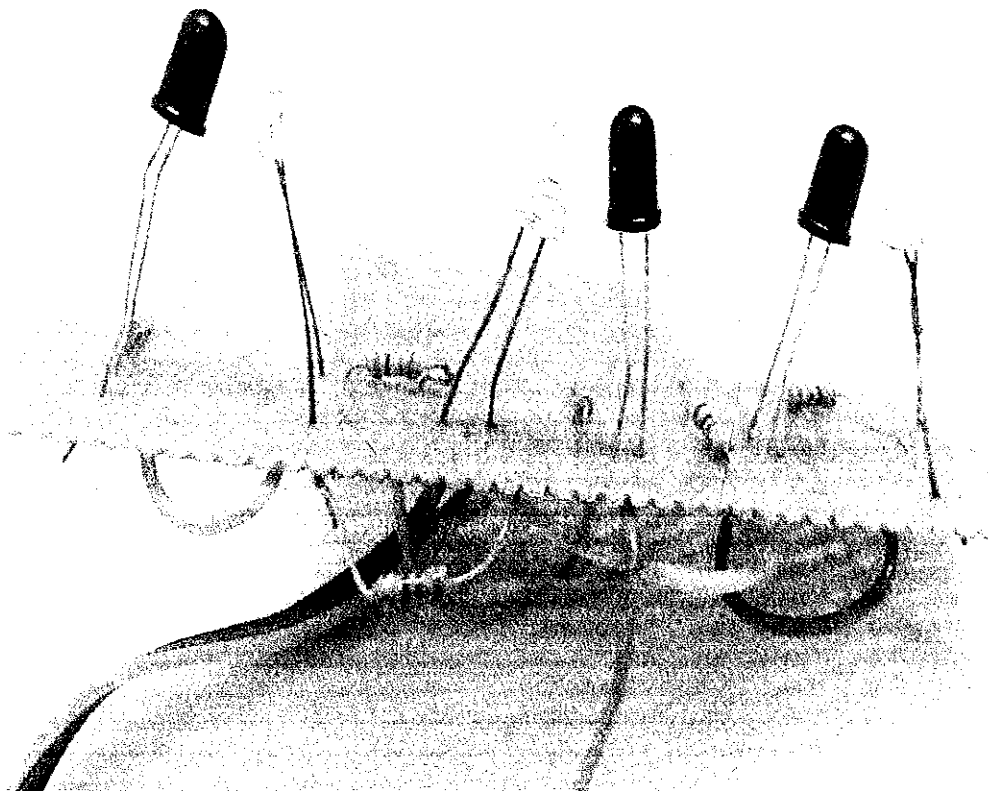
### Module 1: The motor setup driving the wheels of the robot

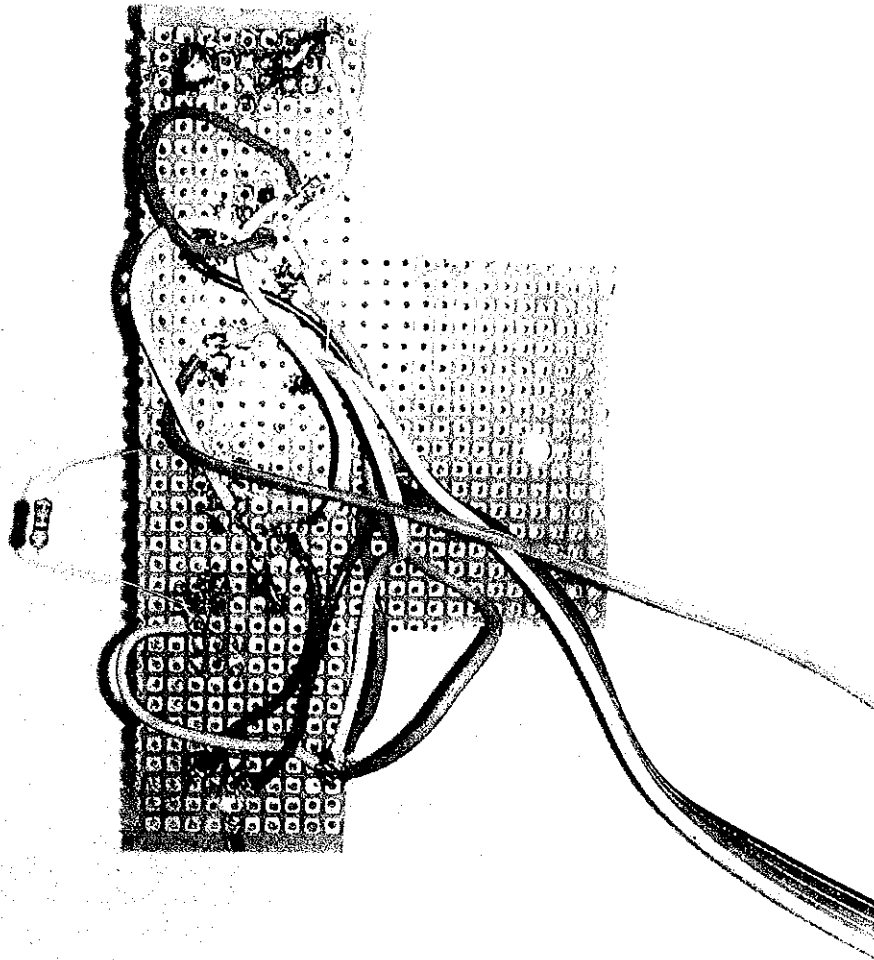


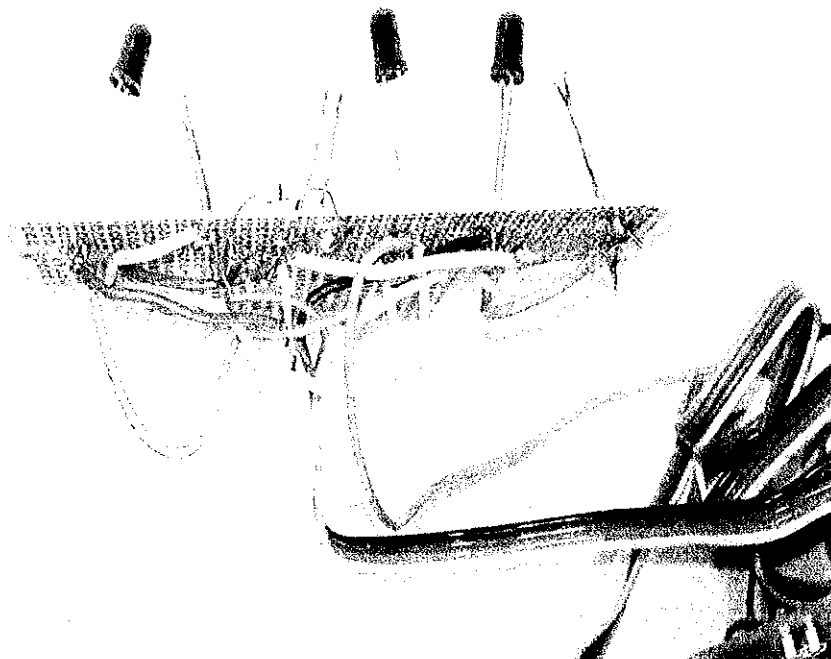




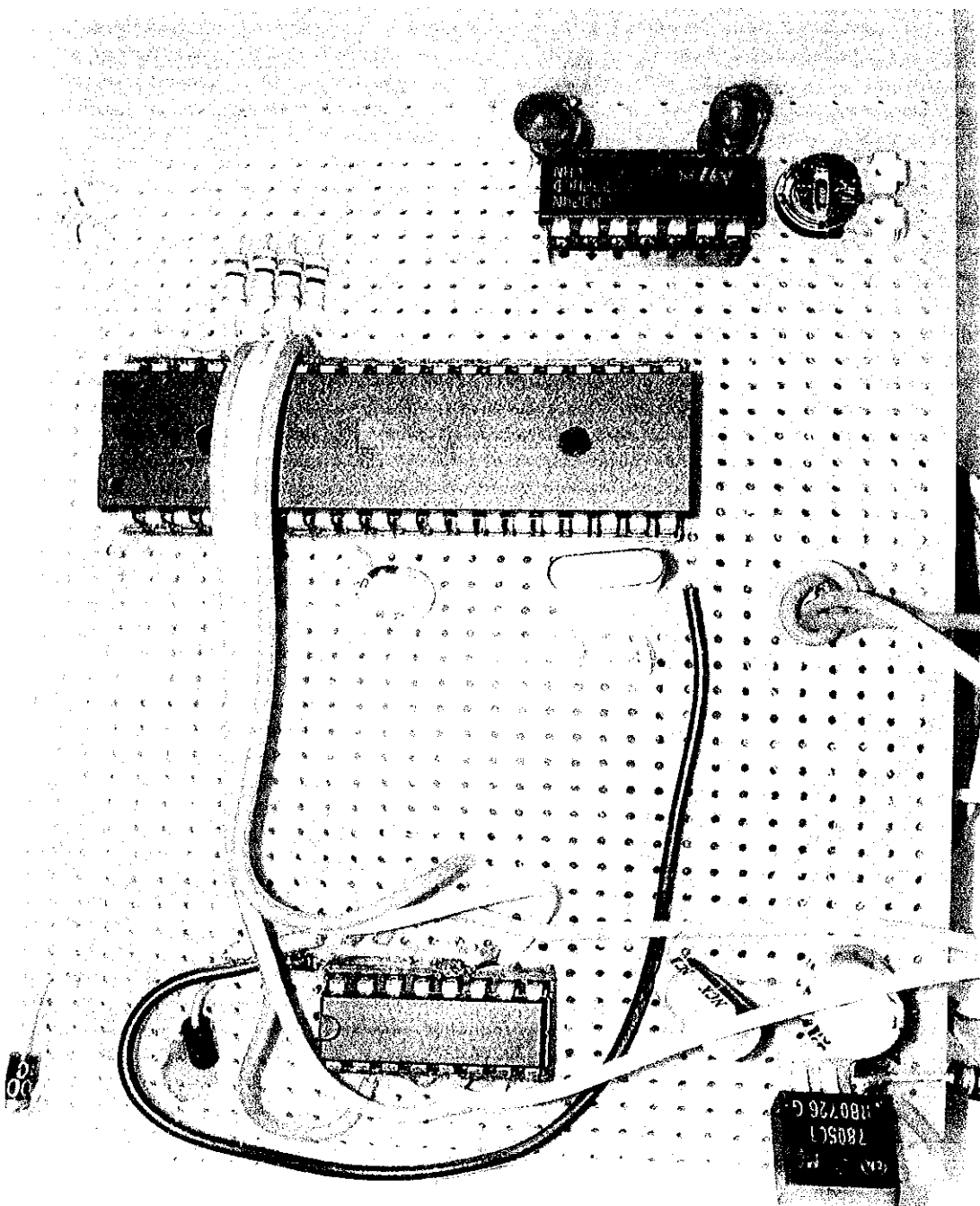
**Module 2: The IR sensor setup used for detecting the contrast change in the path and the forbidden area**



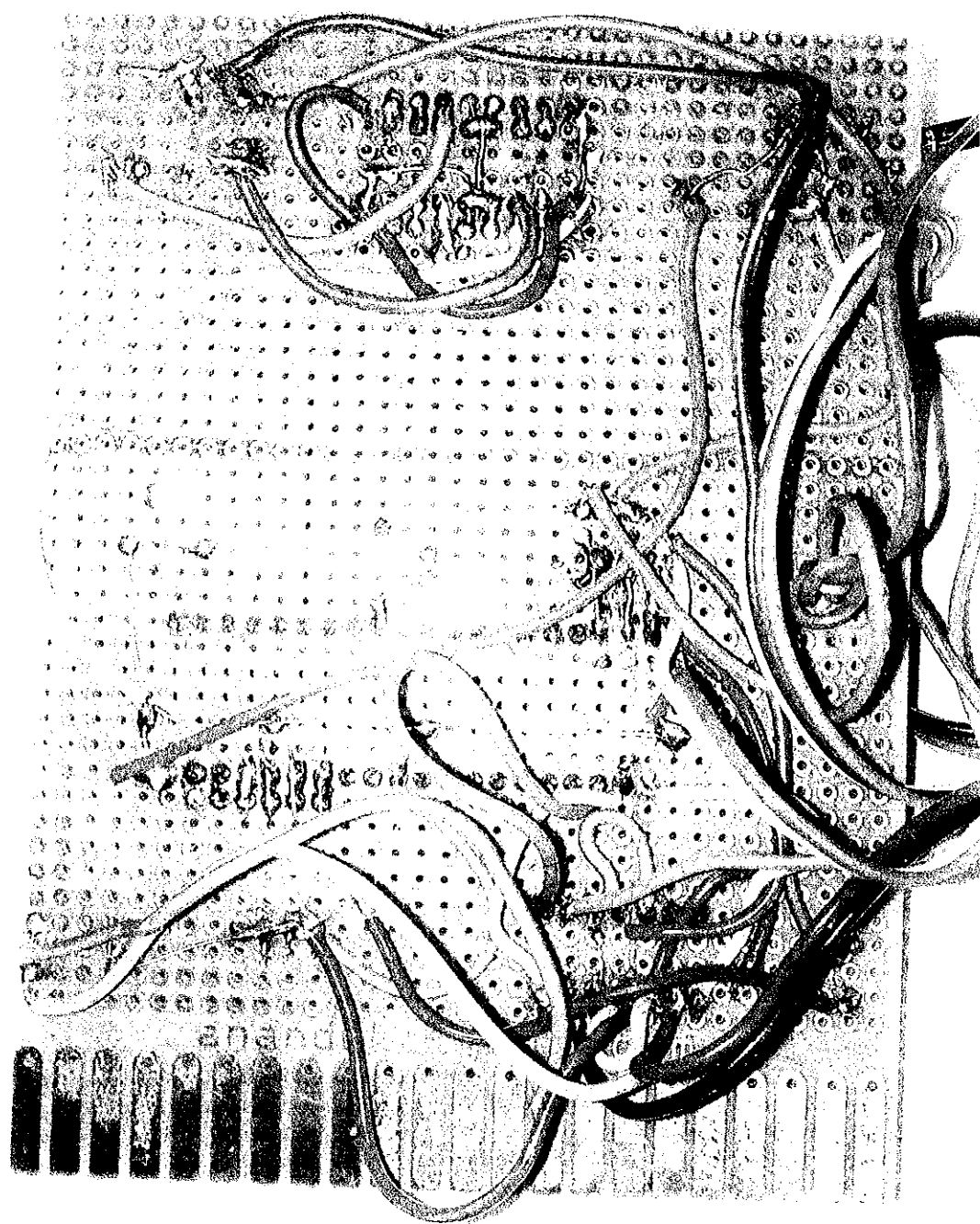




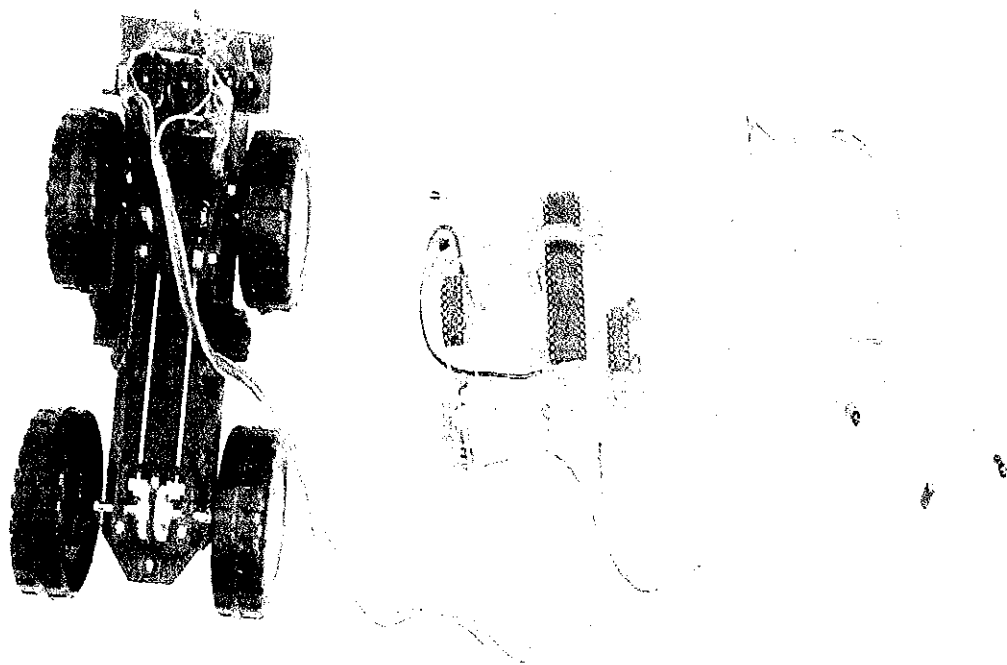
**Module 3:** The microcontroller unit that we programmed to control the signals coming and going to/from the above two mentioned modules







# **Assembly of all the three modules**



## 7. PROJECT DESIGN

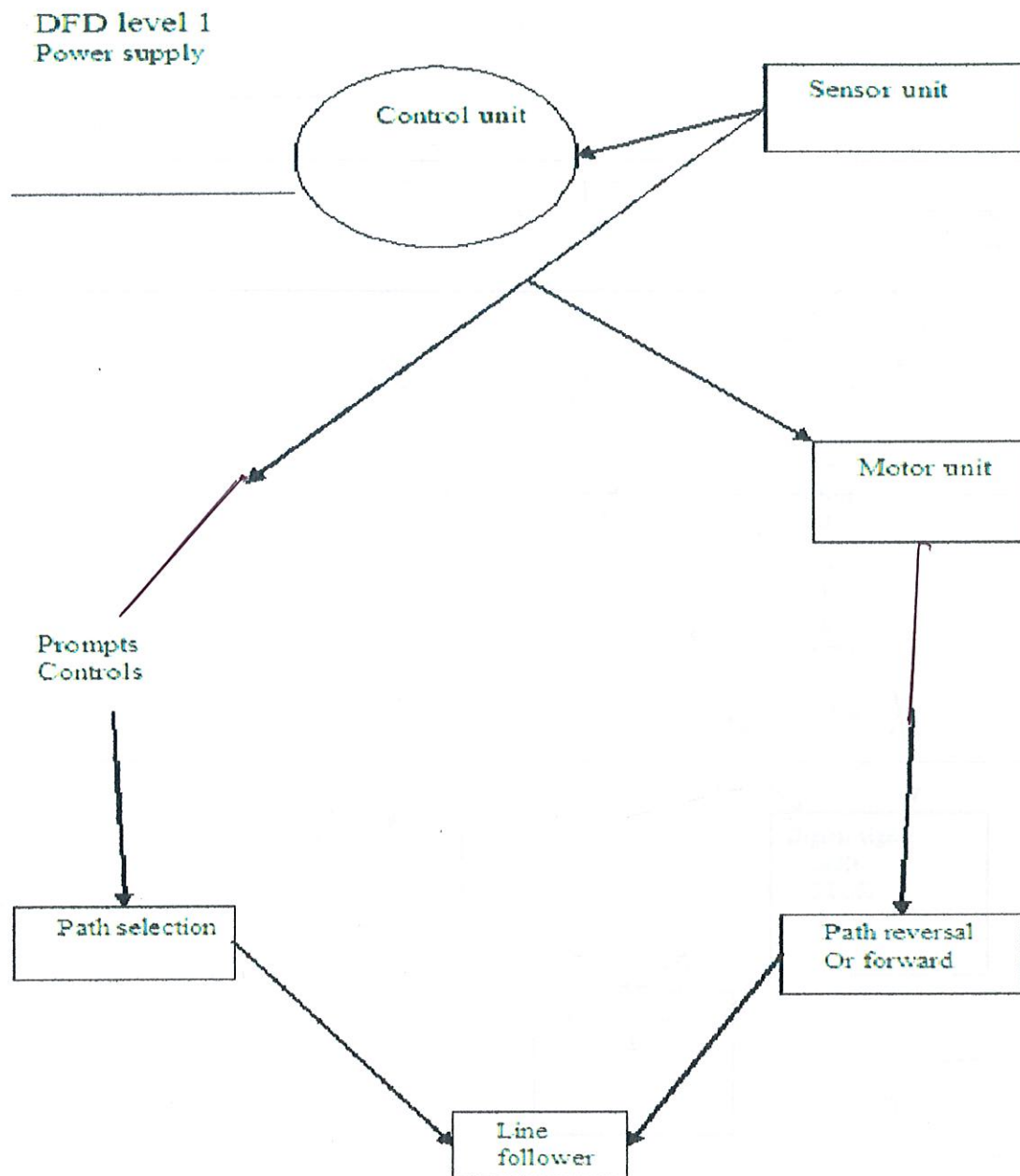


Figure 6.1

## DFD level 2

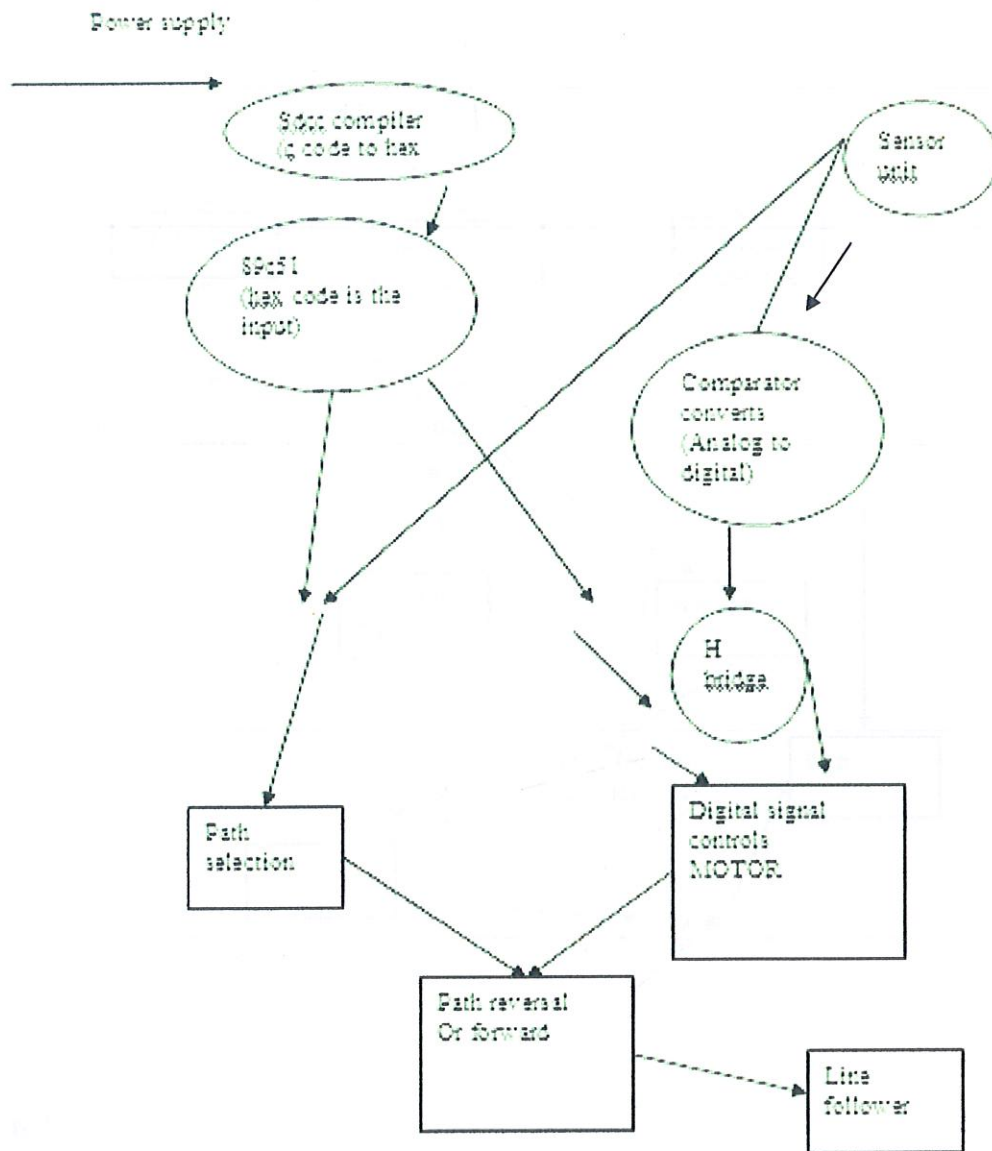


Figure 6.2



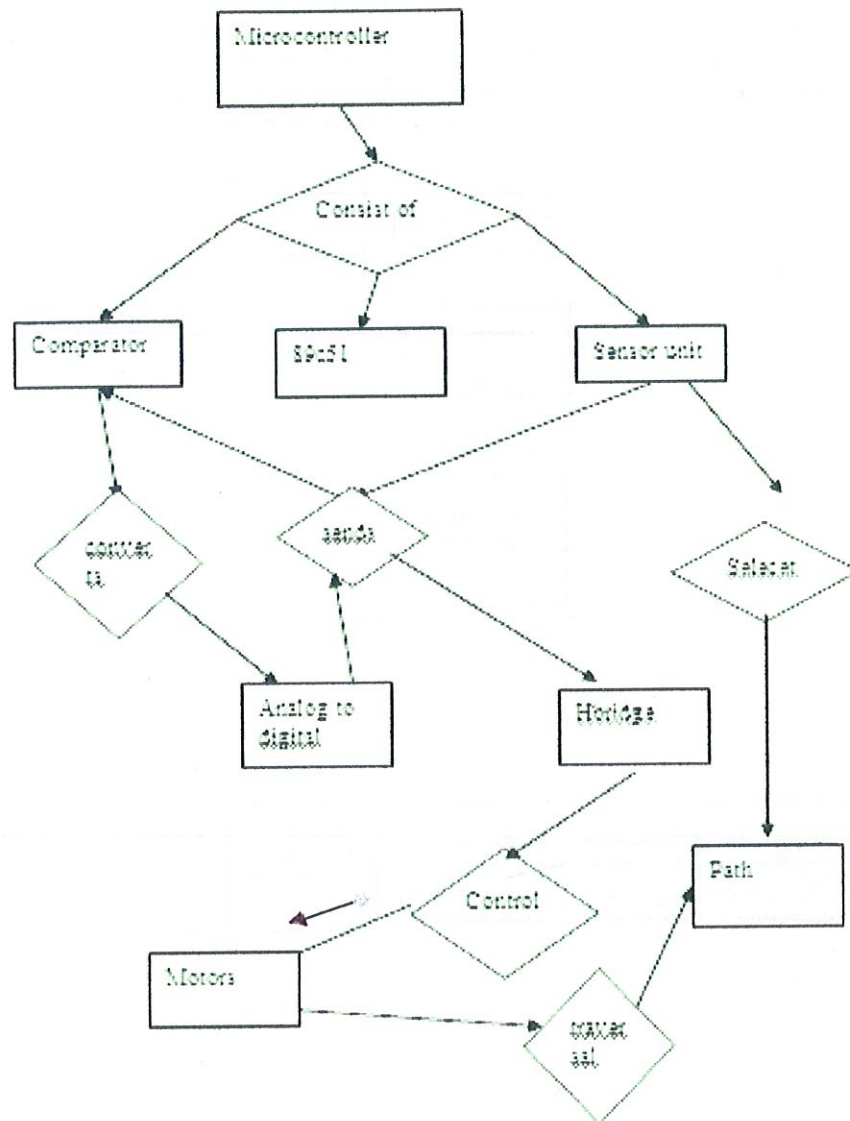
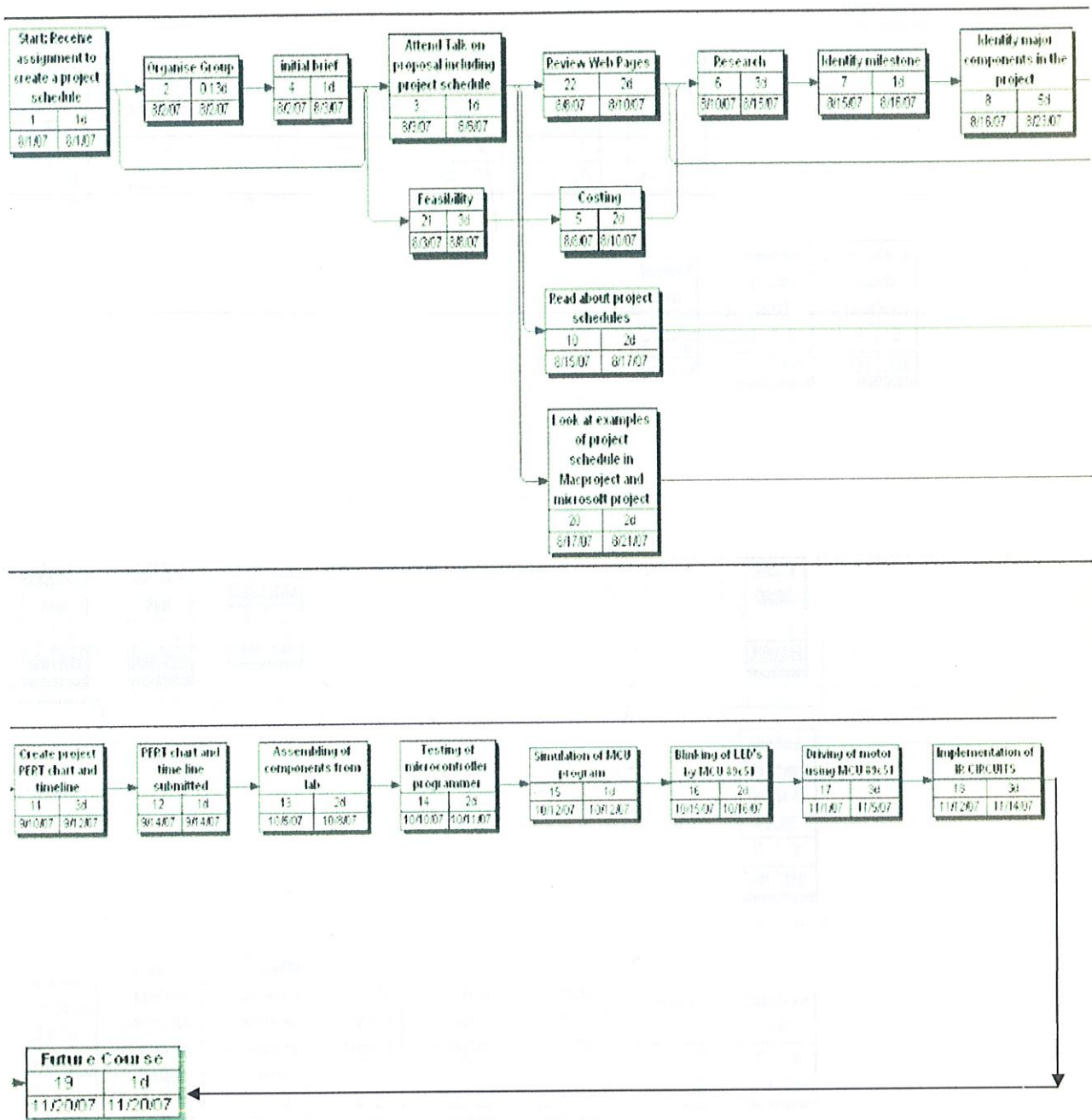
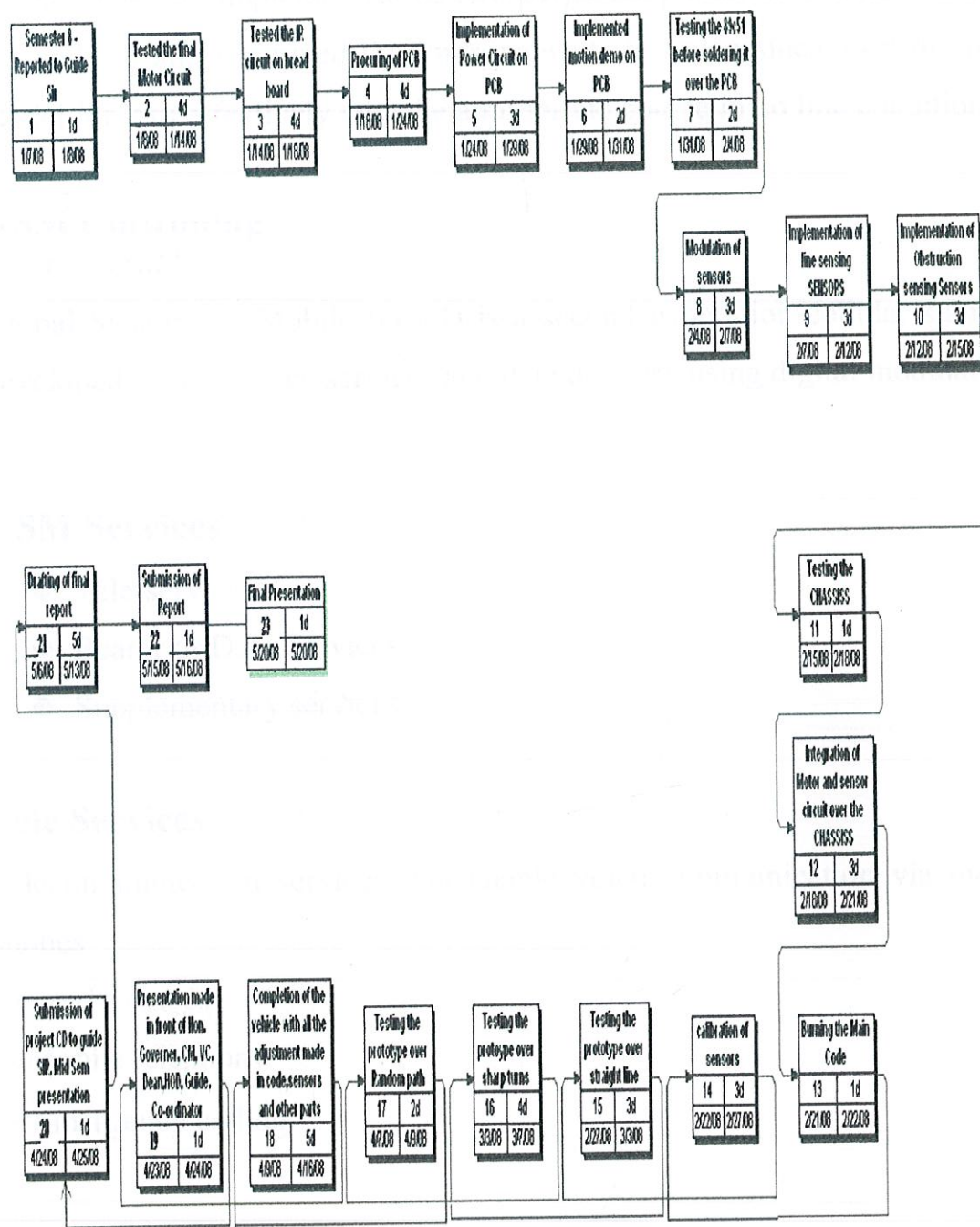


Figure 6.3



## PERT CHART

Figure 6.4



**Figure 6.5**



## 8. CONCLUSION AND FUTURE IMPROVEMENTS

We successively implemented the two purported phases described in section 3.4 and 3.5 and examined the improvements to be conducted on the basic model for it to effectively take the appropriate course in no line condition.

### **GSM controlling**

*What is GSM?*

Global System for Mobile (GSM) is a second generation cellular standard developed to cater voice services and data delivery using digital modulation

### **GSM Services**

- Tele-services
- Bearer or Data Services
- Supplementary services

### **Tele Services**

Telecommunication services that enable voice communication via mobile phones

*Offered services*

- Mobile telephony
- Emergency calling

### **Bearer Services**

- Include various data services for information transfer between GSM and other networks like PSTN, ISDN etc at rates from 300 to 9600 bps



- Short Message Service (SMS)
  - up to 160 character alphanumeric data transmission to/from the mobile terminal
- Unified Messaging Services(UMS)
- Group 3 fax
- Voice mailbox
- Electronic mail

## **Supplementary Services**

Call related services :

- Call Waiting- Notification of an incoming call while on the handset
- Call Hold- Put a caller on hold to take another call
- Call Barring- All calls, outgoing calls, or incoming calls
- Call Forwarding- Calls can be sent to various numbers defined by the user
- Multi Party Call Conferencing - Link multiple calls together
- CLIP – Caller line identification presentation
- CLIR – Caller line identification restriction
- CUG – Closed user group

## **USING OF DTMF IN LINE FOLLOWER**

Dual-tone-multi-frequency (DTMF, also known as touch-tone) are the audible sounds you hear when you press keys on your phone.

## Theory of Operation

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.

Your phone only uses 12 of the possible 16 tones. If you look at your phone, there are only 4 rows (R1, R2, R3 and R4) and 3 columns (C1, C2 and C3). The rows and columns select frequencies from the low and high frequency group respectively. The exact value of the frequencies are listed in Table 3 below:

### LOW-FREQUENCIES

ROW #	FREQUENCY (HZ)
-------	----------------

R1: ROW 0	697
-----------	-----

R2: ROW 1	770
-----------	-----

R3: ROW 2	852
-----------	-----

R4: ROW 3	941
-----------	-----

### HIGH-FREQUENCIES

COL #	FREQUENCY (HZ)
-------	----------------

C1: COL 0	1209
-----------	------

C2: COL 1	1336
-----------	------

C3: COL 2	1477
-----------	------

C4: COL 3	1633
-----------	------

C4 not used in phones

Thus to decipher what tone frequency is associated with a particular key, look at your phone again. Each key is specified by its row and column locations. For example the "2" key is row 0 (R1) and column 1 (C2). Thus

using the above table, "2" has a frequency of  $770 + 1336 = 2106$  Hz The "9" is row 2 (R3) and column 2 (C3) and has a frequency of  $852 + 1477 = 2329$  Hz

A microphone picks up the tones, a preamplifier boosts the signals, an DTMF chip decodes the tones and converts it into BCD signal and thus line follower can respond to it accordingly.

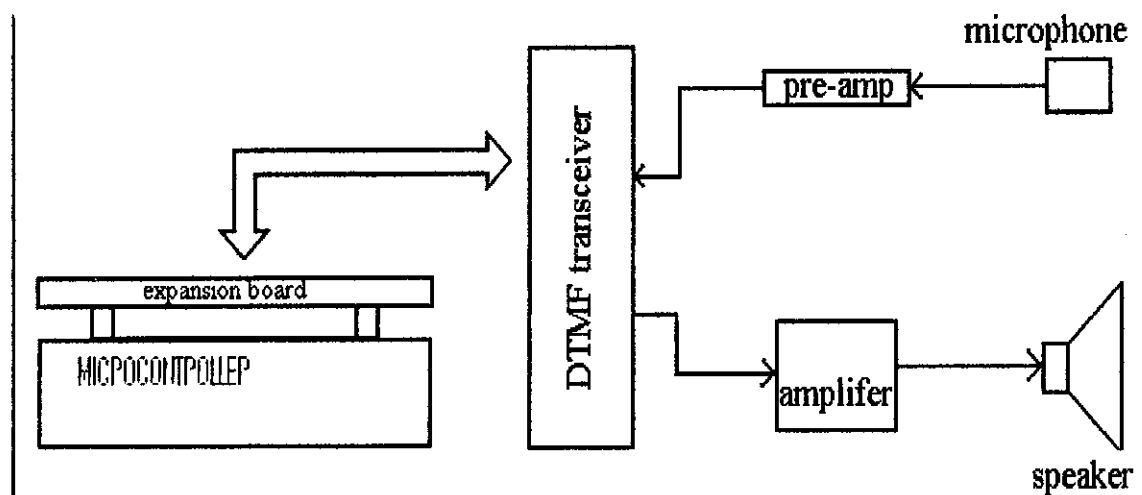


Figure 7.1

## **DTMF (ANALOG) TO BCD (DIGITAL) CONVERSION**

### **IC 8870 DTMF-BCD CONVERTER**

IC 8870 is used in our model for mobile interface. By using this IC we decode the DTMF pulses from the mobile phone through hands-free port and by using this IC we convert these pulses into BCD signal.

### **IC 74154 DEMULTIPLEXER**

A demultiplexer (DEMUX) basically reverses the multiplexing function. It takes digital information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as a data distributor. Decoders can also be used as demultiplexers. A  $2^n$  bit demultiplexer will have 'n' number of select lines. There may be special ports to enable and disable the demultiplexer. In this model, IC 74154, a 16-pin demultiplexer, is used with 4 select lines.

IC 74154 basically converts the BCD signal into decimal signal for further use with the microcontroller. It converts the 4-bit data into 16-bit output. Here in this model there is no use of entire 16 output pins, so we use only 4 output.

## **PROBLEMS in DTMF**

**Amplification Failure** – As the input signal is received by the microphone, it is amplified by a simple circuit before being read by the DTMF receiver. This means that the receiver can only be in an optimal distance away from the sender in order to effectively receive the signal and send to the decoder.

**Microphone's Inefficiency** – The second problem that arose is that the microphone seems to be inefficient in picking up asymmetric/low-amplitude/high-frequency signals. More specifically, the microphone performs well when it is near the source of the signal. However, as the distance is increased, the signal is clipped.

**CONCLUSION** - The extensive research and results have lead to the recommendation of not using DTMF or acoustical communication as a method for mobile robot information exchange. Many factors contribute to the shortfall of this idea. The main factors are, but not limited to, the nature of sound generation, transmission, degradation through air, amplification technique, and receiver technology. These limitations are further aggravated by the mobility of the robots.

Even though DTMF communication is deemed unsuitable as a communication method for mobile robots, there exist other areas where it should be more applicable. We propose using DTMF technology in applications where robots are not mobile but rather a mobile robot communicating with other stationary devices. For example, in healthcare (hospital and home environments), a robot that is capable of sending acoustic commands to turn on/off devices such as light switch or closing door while letting the user know that the process is taking place will be very helpful in allowing the user to feel more comfortable around robots.



## **GSM Applications**

- Mobile telephony
- GSM-R
- Telemetry System
- Fleet management
- Automatic meter reading
- Toll Collection
- Remote control and fault reporting of DG sets

## **GPS Controlling**

### ***What is GPS?***

The Global Positioning System (GPS) is a U.S. space-based radio navigation system that provides reliable positioning, navigation, and timing services to civilian users on a continuous worldwide basis. For anyone with a GPS receiver, the system will provide location and time. GPS provides accurate location and time information for an unlimited number of people in all weather, day and night, anywhere in the world.

The GPS is made up of three parts: satellites orbiting the Earth; control and monitoring stations on Earth; and the GPS receivers owned by users. GPS satellites broadcast signals from space that are picked up and identified by GPS receivers. Each GPS receiver then provides three-dimensional location (latitude, longitude, and altitude) plus the time.

## **Geopositioning -- Basic Concepts**

By positioning we understand the determination of stationary or moving objects. These can be determined as follows:

1. In relation to a well-defined coordinate system, usually by three coordinate values and
2. In relation to other point, taking one point as the origin of a local coordinate system.

The first mode of positioning is known as point positioning, the second as relative positioning. If the object to be positioned is stationary, we term it as static positioning. When the object is moving, we call it kinematics positioning. Usually, the static positioning is used in surveying and the kinematics position in navigation.

## **GPS - Components and Basic Facts**

The GPS uses satellites and computers to compute positions anywhere on earth. The GPS is based on satellite ranging. That means the position on the earth is determined by measuring the distance from a group of satellites in space. The basic principle behind GPS are really simple, even though the system employs some of the most high-tech equipment ever developed. In order to understand GPS basics, the system can be categorised into

### **FIVE logical Steps**

1. Triangulation from the satellite is the basis of the system.

2. To triangulate, the GPS measures the distance using the travel time of the radio message.
3. To measure travel time, the GPS need a very accurate clock.
4. Once the distance to a satellite is known, then we need to know where the satellite is in space.
5. As the GPS signal travels through the ionosphere and the earth's atmosphere, the signal is delayed.

To compute a positions in three dimensions. We need to have four satellite measurements. The GPS uses a trigonometric approach to calculate the positions, The GPS satellites are so high up that their orbits are very predictable and each of the satellites is equipped with a very accurate atomic clock.

## **Components of a GPS**

The GPS is divided into three major components

- The Control Segment
- The Space Segments
- The User Segment

### **The Control Segment**

The Control Segment consists of five monitoring stations (Colorado Springs, Ascension Island, Diego Garcia, Hawaii, and Kwajalein Island). Three of the stations (Ascension, Diego Garcia, and Kwajalein) serve as

uplink installations, capable of transmitting data to the satellites, including new ephemerides (satellite positions as a function of time), clock corrections, and other broadcast message data, while Colorado Springs serves as the master control station. The Control Segment is the sole responsibility of the DoD who undertakes construction, launching, maintenance, and virtually constant performance monitoring of all GPS satellites.

## **The Space Segment**

The DOD monitoring stations track all GPS signals for use in controlling the satellites and predicting their orbits. Meteorological data also are collected at the monitoring stations, permitting the most accurate evaluation of tropospheric delays of GPS signals. Satellite tracking data from the monitoring stations are transmitted to the master control station for processing. This processing involves the computation of satellite ephemerides and satellite clock corrections. The master station controls orbital corrections, when any satellite strays too far from its assigned position, and necessary repositioning to compensate for unhealthy (not fully functioning) satellites.

## **The User Segment**

The user segment is a total user and supplier community, both civilian and military. The User Segment consists of all earth-based GPS receivers. Receivers vary greatly in size and complexity, though the basic design is

rather simple. The typical receiver is composed of an antenna and preamplifier, radio signal microprocessor, control and display device, data recording unit, and power supply. The GPS receiver decodes the timing signals from the 'visible' satellites (four or more) and, having calculated their distances, computes its own latitude, longitude, elevation, and time. This is a continuous process and generally the position is updated on a second-by-second basis, output to the receiver display device and, if the receiver display device and, if the receiver provides data capture capabilities, stored by the receiver-logging unit

### **GSM**

Global System for Mobile communications (GSM: originally from *Group Special Mobile*) is the most popular standard for mobile phones in the world. Its promoter, the GSM promoters estimates that 82% of the global mobile market uses the standard. GSM is used by over 2 billion people across more than 212 countries and territories. Its ubiquity makes international roaming very common between mobile phone operators, enabling subscribers to use their phones in many parts of the world. GSM differs from its predecessors in that both signaling and speech channels are digital call quality, and thus is considered a *second generation (2G)* mobile phone system. This has also meant that data communication was easy to build into the system.

The ubiquity of the GSM standard has been advantageous to both consumers (who benefit from the ability to roam and switch carriers without switching phones) and also to network operators (who can choose equipment from any of the many vendors implementing GSM). GSM also pioneered a low-cost alternative to voice calls, the short message service (SMS, also called "text messaging"), which is now supported on other mobile standards as well.



### **GSM Technical details**

GSM is a cellular network, which means that mobile phones connect to it by searching for cells in the immediate vicinity. GSM networks operate in four different frequency ranges. Most GSM networks operate in the 900 MHz or 1800 MHz bands. Some countries in the Americas (including Canada and the United States) use the 850 MHz and 1900 MHz bands because the 900 and 1800 MHz frequency bands were already allocated.

The rarer 400 and 450 MHz frequency bands are assigned in some countries, notably Scandinavia, where these frequencies were previously used for first-generation systems.

In the 900 MHz band the uplink frequency band is 890–915 MHz, and the downlink frequency band is 935–960 MHz this 25 MHz bandwidth is subdivided into 124 carrier frequency channels, each spaced 200 kHz apart. Time Division Multiplexing is used to allow eight full-rate or sixteen half-rate speech channels per radio frequency channel. There are eight radio timeslots (giving eight burst periods) grouped into what is called a TDMA frame. Half rate channels use alternate frames in the same timeslot. The channel data rate is 270.833 Kbit/s, and the frame duration is 4.615 ms.

The transmission power in the handset is limited to a maximum of 2 watts in GSM850/900 and 1 watt in GSM1800/1900.

GSM has used a variety of voice codecs to squeeze 3.1 kHz audio into between 5.6 and 13 kbit/s. Originally, two codecs, named after the types of data channel they were allocated, were used, called half rate (5.6 kbit/s) and full rate (13 kbit/s). These used a system based upon linear predictive coding (LPC). In addition to being efficient with bitrates, these codecs also made it



easier to identify more important parts of the audio, allowing the air interface layer to prioritize and better protect these parts of the signal.

GSM was further enhanced in 1997 with the enhanced full rate (EFR) codec, a 12.2 kbit/s codec that uses a full rate channel. Finally, with the development of UMTS, EFR was reformed into a variable-rate codec called AMR- Narrowband, which is high quality and robust against interference when used on full rate channels, and less robust but still relatively high quality when used in good radio conditions on half-rate channels.

There are five different cell sizes in a GSM network—macro, micro, pico, femto and umbrella cells. The coverage area of each cell varies according to the implementation environment. Macro cells can be regarded as cells where the base station antenna is installed on a mast or a building above average roof top level. Micro cells are cells whose antenna height is under average roof top level; they are typically used in urban areas. Picocells are small cells whose coverage diameter is a few dozen meters; they are mainly used indoors. Femtocells are cells designed for use in residential or small business environments and connect to the service provider's network via a broadband internet connection. Umbrella cells are used to cover shadowed regions of smaller cells and fill in gaps in coverage between those cells.

Cell horizontal radius varies depending on antenna height, antenna gain and propagation conditions from a couple of hundred meters to several tens of kilometers. The longest distance the GSM specification supports in practical use is 35 kms (22 mi). There are also several implementations of the concept of an extended cell, where the cell radius could be double or even more, depending on the antenna system, the type of terrain and the timing advance.

Indoor coverage is also supported by GSM and may be achieved by using an indoor picocell base station, or an indoor repeater with distributed indoor antennas fed through power splitters, to deliver the radio signals from an antenna outdoors to the separate indoor distributed antenna system. These are typically deployed when a lot of call capacity is needed indoors, for example in shopping centers or airports. However, this is not a prerequisite, since indoor coverage is also provided by in-building penetration of the radio signals from nearby cells.

The modulation used in GSM is Gaussian minimum shift-keying (GMSK), a kind of continuous-phase frequency shift keying. In GMSK, the signal to be modulated onto the carrier is first smoothed with a Gaussian low-pass filter prior to being fed to a frequency modulator, which greatly reduces the interference to neighboring channels (adjacent channel interference).

### **Network structure**

GSM system seen by the customer is large and complicated in order to provide all of the services which are required. It is divided into a number of sections and these are each covered in separate articles.

- The Base station subsystem (the base stations and their controllers).
- The Network and Switching Subsystems (the part of the network most similar to a fixed network). This is sometimes also just called the core network.
- The GPRS Core Network (the optional part which allows packet based Internet connections).

- All of the elements in the system combine to produce many GSM services such as voice calls and SMS.

### **GSM security**

GSM was designed with a moderate level of security. The system was designed to authenticate the subscriber using a pre-shared key and challenge response. Communications between the subscriber and the base station can be encrypted. The development of UMTS introduces an optional USIM, that uses a longer authentication key to give greater security, as well as mutually authenticating the network and the user - whereas GSM only authenticated the user to the network (and not vice versa). The security model therefore offers confidentiality and authentication, but limited authorization capabilities, and no non-repudiation.

GSM uses several cryptographic algorithms for security. The A5/1 and A5/2 stream ciphers are used for ensuring over-the-air voice privacy. A5/1 was developed first and is a stronger algorithm used within Europe and the United States; A5/2 is weaker and used in other countries. Serious weaknesses have been found in both algorithms: it is possible to break A5/2 in real-time with a cipher text-only attack, and in February 2008, Pico Consulting, Inc revealed its ability and plans to commercialize FPGAs that

allow A5/1 to be broken with a rainbow table attack. The system supports multiple algorithms so operators may replace that cipher with a stronger one.

## **How GPS Receivers Work**

- When people talk about "a GPS," they usually mean a GPS receiver. The Global Positioning System (GPS) is actually a constellation of 27 Earth-orbiting satellites (24 in operation and three extras in case one fails). The U.S. military developed and implemented this satellite network as a military navigation system, but soon opened it up to everybody else.
- Each of these 3,000- to 4,000-pound solar-powered satellites circles the globe at about 12,000 miles (19,300 km), making two complete rotations every day. The orbits are arranged so that at any time, anywhere on Earth, there are at least four satellites "visible" in the sky.
- A GPS receiver's job is to locate four or more of these satellites, figure out the distance to each, and use this information to deduce its own location. This operation is based on a simple mathematical principle called trilateration.



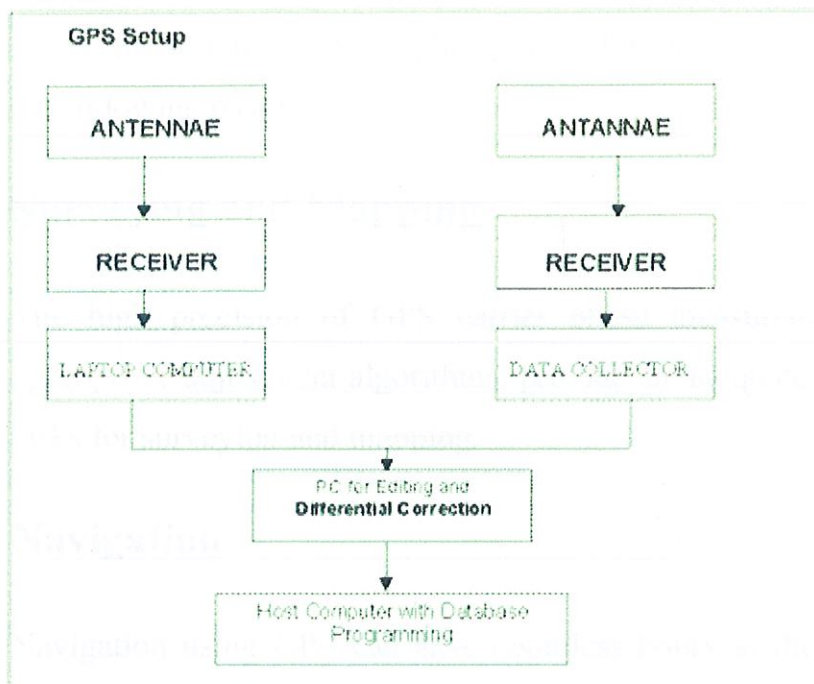


Figure 7.2

## **Grounding**

The radio antenna may be a target for lightning. To avoid damage, you may wish to ground your reference station antenna.

## **GPS Applications**

One of the most significant and unique features of the Global Positioning Systems is the fact that the positioning signal is available to users in any position worldwide at any time. With a fully operational GPS system, it can be generated to a large community of likely to grow as there are multiple applications, ranging from surveying, mapping, and navigation to GIS data

capture. The GPS will soon be a part of the overall utility of technology.

There are countless GPS applications, a few important ones are covered in the following passage.

## **Surveying and Mapping**

The high precision of GPS carrier phase measurements, together with appropriate adjustment algorithms, provide an adequate tool for a variety of tasks for surveying and mapping.

## **Navigation**

Navigation using GPS can save countless hours in the field. Any feature, even if it is under water, can be located up to one hundred meters simply by scaling coordinates from a map, entering waypoints, and going directly to the site.

## **Remote Sensing and GIS**

It is also possible to integrate GPS positioning into remote-sensing methods such as photogrammetry and aerial scanning, magnetometry, and video technology. The GIS user community benefits from the use of GPS for locational data capture in various GIS applications. The GPS can easily be linked to a laptop computer in the field, and, with appropriate software, users can also have all their data on a common base with every little distortion. Thus GPS can help in several aspects of construction of accurate and timely GIS databases.

## **Military**

The GPS was primarily developed for real time military positioning. Military applications include airborne, marine, and land navigation.

## **Future of GPS Technology**

Barring significant new complications due to S/A (Selective Availability) from DOD, the GPS industry is likely to continue to develop in the civilian community. There are currently more than 50 manufacturers of GPS receivers, with the trend continuing to be towards smaller, less expensive, and more easily operated devices. While highly accurate, portable (hand-held) receivers are already available, current speculation envisions inexpensive and equally accurate 'wristwatch locators' and navigational guidance systems for automobiles. However, there is one future trend that will be very relevant to the GIS user community, namely, community base stations and regional receive networks, as GPS management and technological innovations that will make GPS surveying easier and more accurate.

*GSM module and GPS module work in co-ordination with each other , the GPS receiver tells the location of the robot to the GPS module, this location is transmitted by the GPS transmitter to GSM receiver which sends to the microcontroller 89c51 which in turn decides to move left or right.*

## 9.BIBLIOGRAPHY

### Books:

Programming and Customizing the AVR Microcontroller – Dhananjay V. Gadre

Parallel Port Complete – Jan Axelson

The 8051 Microcontroller and Embedded Systems- Muhammad Ali Mazidi, Janice Gillispie Mazidi

### Links:

Atmel Corp.

Makers of the AVR microcontroller

<http://www.atmel.com>

AVRbeginners.net

<http://www.avrbeginners.net/>

AVR assembler tutorial

Tutorial for learning assembly language for the AVR-Single-Chip-Processors

AT90Sxxxx from ATMEL with practical examples.

<http://www.avr-asm-tutorial.net/>

One of the best sites AVR sites

<http://www.avrfreaks.net>

WinAVR

An open source C compiler for AVR

<http://sourceforge.net/projects/winavr>

PonyProg

A widely used programmer. Support for newer chips is added periodically. Can also program PICs and EEPROMS

<http://www.lancos.com/prog.html>

Basic Electronics

<http://www.kpsec.freeuk.com/>

Williamson Labs

Nice animated tutorials, articles and project ideas.

<http://www.williamson-labs.com/home.htm>

#### Small Robot Sensors

[http://www.andrew.cmu.edu/user/tjg/websensors/robot\\_sensors2.html](http://www.andrew.cmu.edu/user/tjg/websensors/robot_sensors2.html)

#### Robotics India

An Indian site devoted to robotics. Must see

<http://www.roboticsindia.com/>

#### Seattle Robotics Society

<http://www.seattlerobotics.org/>

F.D.Makaya, D.Chatelain and LW Snyman, "Design and performance assessment of a prototype wireless controlled robot", *IEEE – EDMO*, pp. 115-118 , November 2004.