GSM GUARD SYSTEM FOR VEHICLES

A PROJECT REPORT

Submitted by

KOMALAPRIYA.O.S	**71203106025**
KM.KUMKUM SINGH	**71203106026**
LAVANYA.B	**71203106027**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2007

ANNA UNIVERSITY: CHENNAI 600 025

CERTIFICATE

BONAFIDE CERTIFICATE

Certified that this project report **“GSM GUARD SYSTEM FOR VEHICLES”** is the bonafide work of **“KOMALAPRIYA.O.S,KM.KUMKUM SINGH,LAVANYA.B”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.M.Rajeshwari	Mr.G.C.Thiyagarajan
HEAD OF DEPARTMENT	**SUPERVISOR**
Electronics & Communication Engineering,	Electronics & Communication Engineering,
Kumaraguru College of Technology,	Kumaraguru College of Technology,
Coimbatore - 641006	Coimbatore - 641006

The candidates with University Register Nos. 71203106025, 71203106026, 71203106027 were examined by us in the project viva-voce examination held on __________.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We are grateful to our principal **Dr. Joseph V. Thanikal, Ph.D. Kumaraguru College of Technology, Coimbatore,** for his patronage.

We profusely thank our professor and the head of the department **Dr.Rajeshwari Mariappan , Ph.D,** for her encouragement and support.

We are highly indebted and grateful to our guide **Mr.G.C.Thiyagarajan, M.E., Senior lecturer, Department of Electronics and Communication Engineering,** for his invaluable suggestions and constant guidance.

We are grateful to our project coordinator **Prof. Ramprakash** for his Support and guidance in completing the project in time.

We express our sincere gratitude to **Mr. Venkatesh**, **B.E**, **AGT Electronics, Coimbatore** for his constant motivation and assistance.

We thank our beloved friends, who stood as moral support for us throughout the project proceedings. Last but not the least , we are thankful to our **parents** who helped us in all aspects.

ABSTRACT

ABSTRACT

In the existing systems which monitor the vehicle condition, the parameters of the vehicle are made known to the owner only through the driver. But in our GSM guard system, the parameters of the vehicle are made known to the owner automatically without the knowledge of the driver. This provides security and accuracy of the vehicle condition.

The parameters of the vehicle monitored are alcohol usage of the driver, speed, humidity level of the air conditioner, door locking and security system. They are monitored by the Microcontroller (Atmel 89C52) which receives inputs from Gas sensor, C type sensor, Temperature sensor.

The output ports of the microcontroller are connected to the computer. The computer will give the information in the text format to the mobile within the guard system.

Our guard system will send the alcohol level used by the driver, speed of the vehicle, humidity level of the air conditioner, door locking condition, security system to the mobile of the owner at regular intervals.

The mobile in the guard system automatically sends this message to the owner mobile. The owner checks whether these parameters fall within the acceptable level, if not he can send the reply message and control the vehicle.

CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES

INTRODUCTION

INTRODUCTION

In today’s world vehicles come with different security mechanism . But they are mostly mechanical and costly . But here we are going to give insight into a GSM security system which is purely electronic and more secure than the conventional security systems available . Here we have designed an GSM guard system which will have microcontroller , computer , mobile within it which is placed within the vehicle.

We have used microcontroller which monitors the various parameters of vehicle like the alcohol level used by the driver, speed of the vehicle, humidity level of the air conditioner, door locking condition and give it to computer. Then the computer transfers various parameters in the message format to the mobile in GSM guard system.

Our guard system provides two way communication facility . The mobile in the guard system transfers this information’s to owner’s mobile automatically and the owner can control the vehicle by sending message to the mobile in the GSM guard system and vehicle performs according to message passed by the owner . Thus the main aim of our project is to prevent accidents

BLOCK DIAGRAM

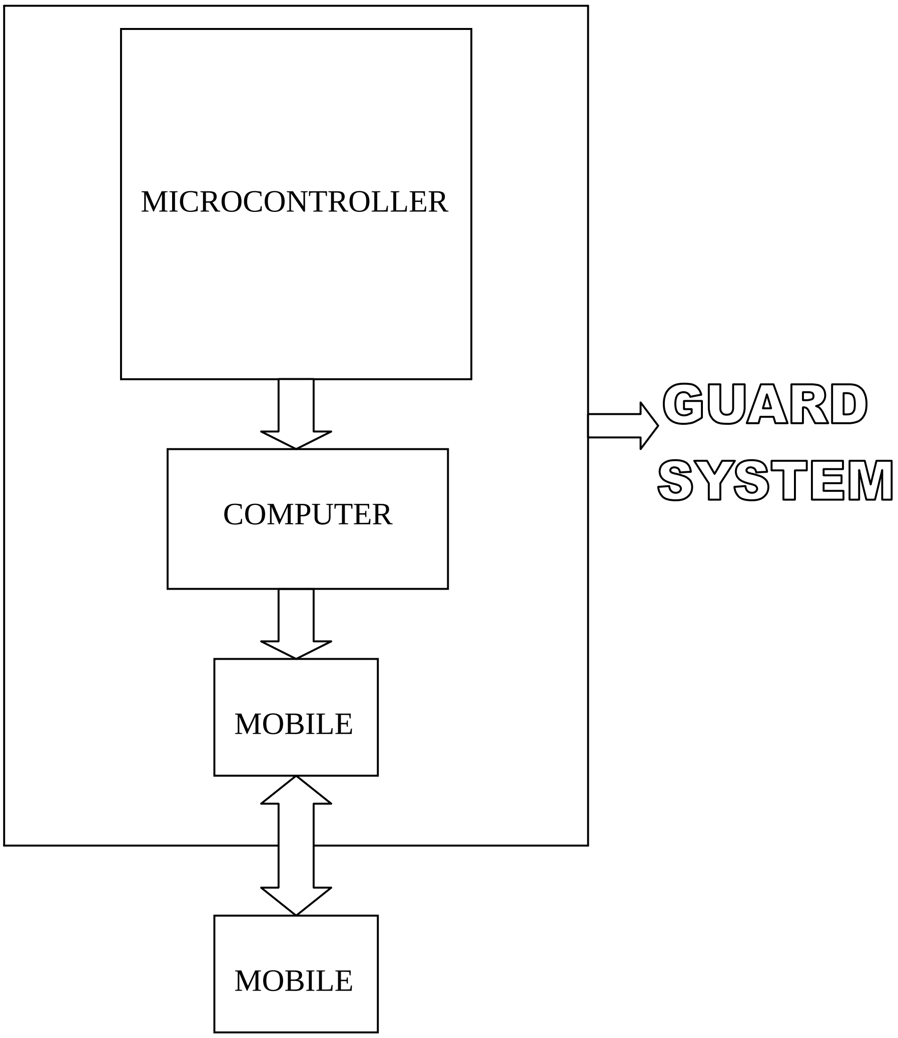


Fig. 1.1 BLOCK DIAGRAM OF GSM GUARD SYSTEM

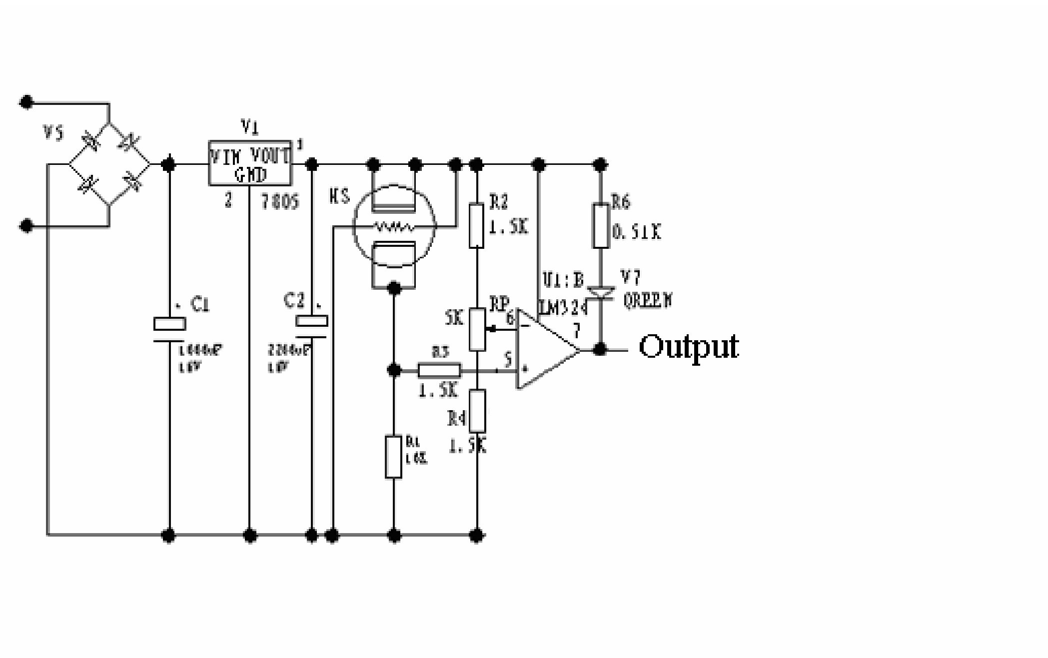
PROJECT DETAILS

ALCOHOL DETECTOR

ALCOHOL DETECTOR

The alcohol detector has a gas sensor which detects the alcohol level used by the driver. This information is given to the microcontroller(89C52),computer and the mobile of the guard system. This information is given to the owner mobile and the owner can in turn control the motor of the vehicle by sending a text message.

CIRCUIT DIAGRAM :



.

Fig.2.1.1 Gas sensor

The above circuit diagram consists of a bridge circuit, a voltage regulator, a gas sensor, a comparator and a LED.

GAS SENSOR :

The gas sensor used here is the TGS gas sensor.

GENERAL CHARACTERISTICS :

The sensing material in TGS gas sensors is metal oxide, most typically SnO2. When a metal oxide crystal such as SnO2 is heated at a certain high temperature in air, oxygen is adsorbed on the crystal surface with a negative charge. Then donor electrons in the crystal surface are transferred to the adsorbed oxygen, resulting in leaving positive charges in a space charge layer. Thus, surface potential is formed to serve as a potential barrier against electron flow (Figure 1).

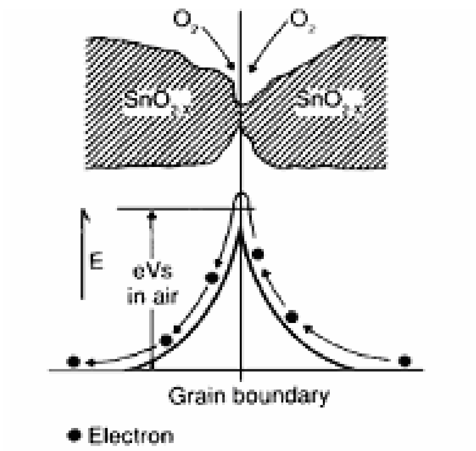


Fig.2.1.2 Model of inter-grain potential barrier (in the absence of gas)

Inside the sensor, electric current flows through the conjunction parts (grain boundary) of SnO2 microcrystals. At grain boundaries, adsorbed oxygen forms a potential barrier which prevents carriers from moving freely. The electrical resistance of the sensor is attributed to this potential barrier. In the presence of a deoxidizing gas, the surface density of the negatively charged oxygen decreases, so the barrier height in the grain boundary is reduced (Figures 3)

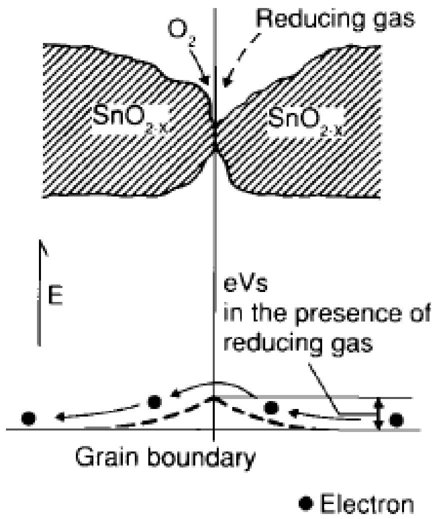


Fig.2.1.3 Model of inter-grain potential barrier

(in the presence of gas)

The reduced barrier height decreases sensor resistance. The relationship between sensor resistance and the concentration of deoxidizing gas can be expressed by the following equation over a certain range of gas concentration:

$$Rs = A[C]^{-a}$$

where: Rs = electrical resistance of the sensor

A = constant

[C] = gas concentration

a = slope of Rs curve

INSTALLATION :

If the detected gas is LPG, Butane and propane which is heavier than normal air, the gas sensor is installed about 1.00 meter above the ground, adversely, For the Natural gas, Methane, coal gas, CO and H2, which is lighter than the normal air, gas sensor is installed about 1 meter below the roof, For both of cases there should be good air circulation.

Detecting Gas	**Calibrating Concentration**
Combustible gas/Smoke	1000ppm±30%
Natural Gas/Methane	5000ppm±30%
Coal Gas/Methane/LPG	2000/5000/2000ppm±30%
LPG	3000ppm±30%
CO	200ppm±30%
Coal Gas	800ppm±30%

COMPARATOR :

The comparator used here is LM324. It consists of four independent, high gain, internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide range of voltages. The operational temperature is 0C to +70C. It has a wide bandwidth of 1MHz.

VOLTAGE REGULATOR :

The voltage regulator used here is 7805. It is a three terminal, positive fixed voltage regulator. It provides an output volage of 5V. It has two capacitors. The input capacitor C1 is usually connected between input terminal and ground to cancel the inductive effects due to long distribution leads. The output capacitor C2 improves the transient response

CIRCUIT OPERATION :

The Transformer is used to convert 230V to 12V. The bridge circuit used here converts the ac voltage to pulsating dc voltage. Then the voltage regulator 7805 provides a constant dc voltage of 5V. The capacitors are used to remove ripples. The comparator LM324 compares the applied voltage with the reference voltage. If the comparator output increases above a specified level, the LED will not glow indicating the alcohol presence.

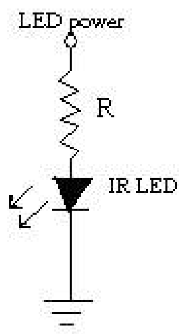
This information is then given to the microcontroller 89C52.

C TYPE SENSOR

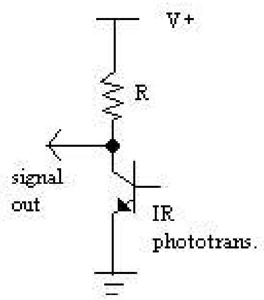
C TYPE SENSOR

Overview

An **infrared emitter** is an LED made from gallium arsenide, which emits near-infrared energy at about 880nm. The infrared phototransistor acts as a transistor with the base voltage determined by the amount of light hitting the transistor. Hence it acts as a variable current source. Greater amount of IR light cause greater currents to flow through the collector-emitter leads. As shown in the diagram below, the phototransistor is wired in a similar configuration to the voltage divider. The variable current traveling through the resistor causes a voltage drop in the pull-up resistor. This voltage is measured as the output of the device.



An IR emitter



An IR phototransistor

Fig.2.2.1 IR emitter and IR phototransistor

Applications

One of the applications of the IR emitter and IR phototransistor is a **photo-reflector.** The photo-reflector is a small rectangular device that contains an phototransistor (sensitive to infrared light) and an infrared emitter. The amount of light reflected from the emitter into the phototransistor yields a measurement of a surface's reflectance.The photoreflector can be used in robot to follow a path (e.g. a white line on the floor).

Infrared Reflectance Sensor

IR reflectance sensors contain a matched infrared transmitter and infrared receiver pair. These devices work by measuring the amount of light that is reflected into the receiver. Because the receiver also responds to ambient light, the device works best when well shielded from abient light, and when the distance between the sensor and the reflective surface is small(less than 5mm). IR reflectance sensors are often used to detect white and black surfaces. White surfaces generally reflect well, while black surfaces reflect poorly. One of such applications is the line follower of a robot.

The diagram on the right shows an example of a infrared reflectance sensor. For this particular example, the IR detector being used can be ordered from Mouser Electronics (Part# 512-QSE113) and the IR emitter can be ordered form Digi-Key (Part# LN175PA-ND). The resistors being used in this example are chosen to match the electrical properties of the IR dectector and emitter.

You might want to choose different resistors if you use different dectectors and emitters. Different resistor values affect the sensitivity of the infrared reflectance sensor.

The following is a test IC program used to illustrate how to use the IR emitter and detector with the Handy Board. The program will switch on a servo motor when the the IR detector receives enough Infra-red light. The program uses the analog input 6 of the Handy Board to read from the IR reflectance sensor. (The lower the value of the variable *num* is, the higher the intensity of the IR light)

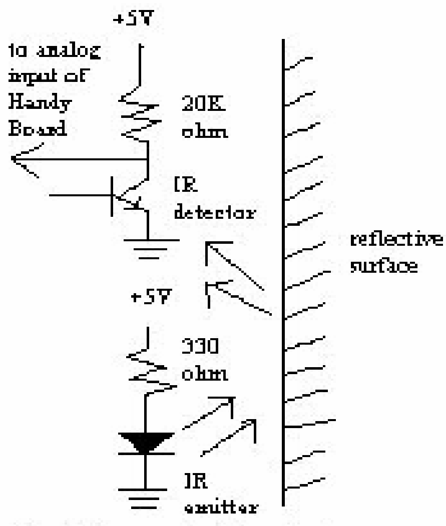


Fig . 2.2.2 Circuit diagram of infrared reflectance sensor

IR slotted optical switch

An infrared slotted optical switch is a device similar to the photo-reflector except that the emitter is pointed directly into the phototransistor. The slotted optical switch can be used to build *shaft encoders*. Shaft encoders can give the robot feedback on how far its wheels have turned or on synchronizing two wheels' velocity. A shaft encoder usually consists of aslotted optical switch and a striped wheel with a palette of radically alternating holes or slots on it. The palette of stripes will alternately reflect or not reflect light to the phototransistor, yielding a pulse-train output. The robot can then tell how far its wheels have rotated by counting the pulses.

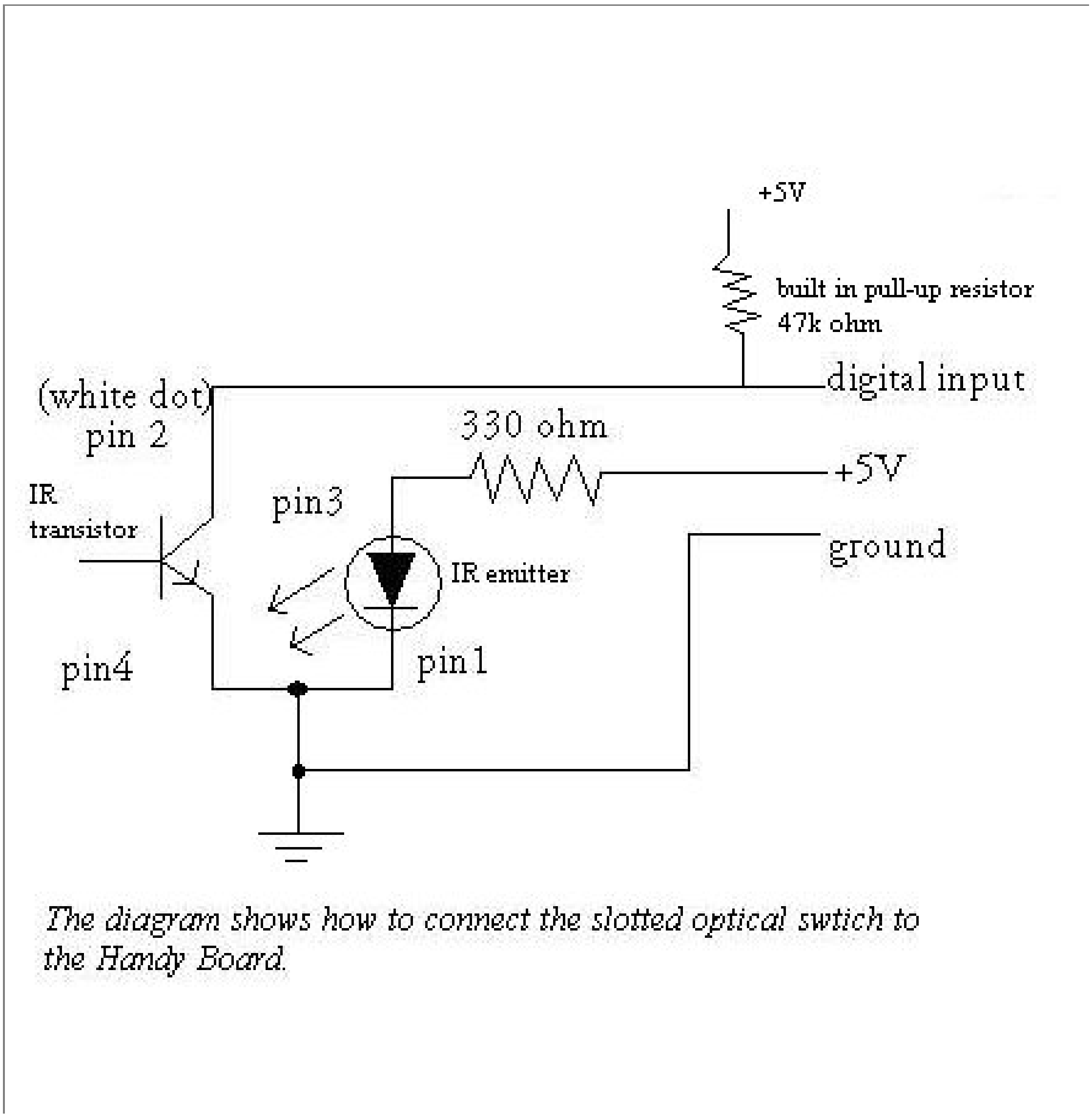


Fig 2.2.3 Slotted optical switch

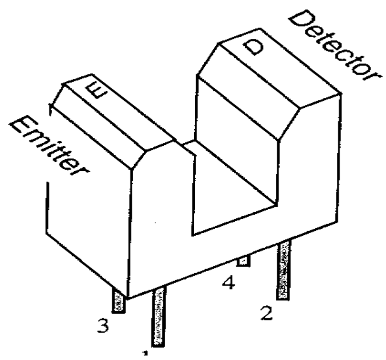


Fig.2.2.4 A typical IR slotted optical switch

The wiring for the slotted optical switch is straightforward. The white dot on the optical switch corresponds to pin 2 in the pinout diagram and you can figure out the pins using the pinout diagram shown above. The emitter LED is powered by te *Handy Board*'s +5V supply, with a 330 ohm resistor in series to limit the current through the LED to an appropriate value. In fact, there is a pull-up resistor of 4.7K ohm built in the *Handy Board* and it is not shown in the diagram. Different varieties of phototransistor, however, may perform better with a smaller resistor value than the on-board 47K resistor. If the sensitivity of the device is poor, you can try connecting the signal line to the power supply through another resistor to determine the best response.

TEMPERATURE SENSOR

LM35- PRECISION CENTIGRADE TEMPERATURE SENSOR

General description:

The LM35 series are precision integrated circuit temperature sensors, whose output voltage are linearly proportional to the Celsius temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in Kelvin as the user is not required to subtract a large constant voltage from its output to obtain a convenient centigrade scaling. The LM35 does not require any external calibration or trimming. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60micro amperes from its supply,it has very low self heating, less than 0.1 degree celcius in still air. The LM35 is available package in hermetic TO-46 transistor packages.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 μA current drain
- Low self-heating, 0.08°C in still air
- Low impedance output, 0.1 Ω for 1 mA load

Connection diagram:

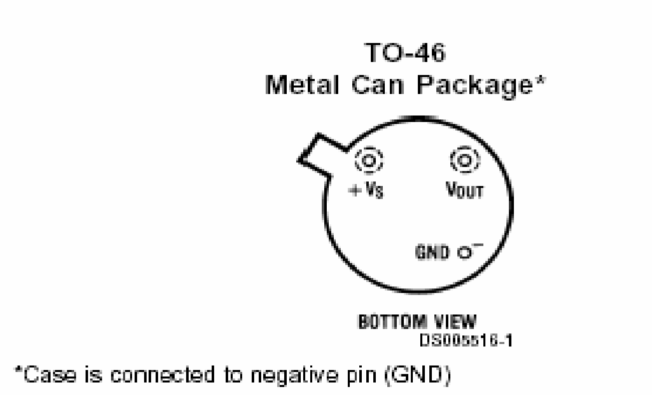
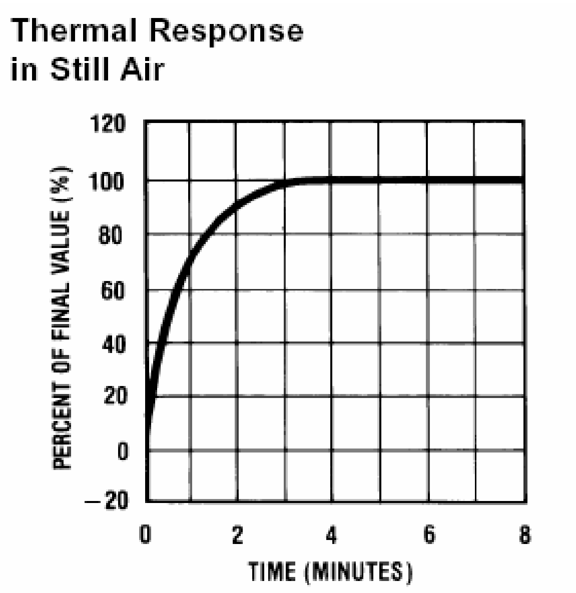
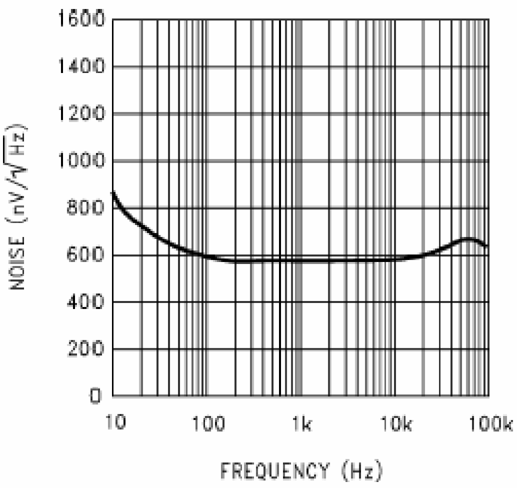


Fig.2.3.1 Connection diagram of c- type sensor

Typical performance characteristics:



Noise Voltage



Start-Up Response

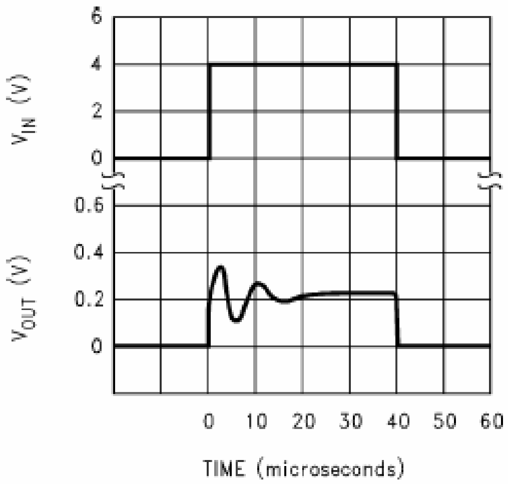


Fig.2.3.2 Typical performance characteristics of c- type sensor

Applications:

The LM35 can be applied easily in the same way as other integrated circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01 degree celsius of the surface temperature.

This presumes that the ambient air temperature is also the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. To minimize this problem be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will ensure that the leads and wires are all at the same temperature as the surface, and that the LM35 die’s temperature will not be affected by air temperature.

The TO-46 can also be soldered to a metal surface or pipe without damage. Of course in that case the V terminal of the circuit will be grounded to the metal. Alternatively the LM35 can be mounted inside a sealed end metal tube, and can be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry to avoid leakage and corrosion. This is

Especially true if the circuit may operate at cold temperatures where condensation can occur.

These devices are sometimes soldered to a small light weight heat fin to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature to digital converter:

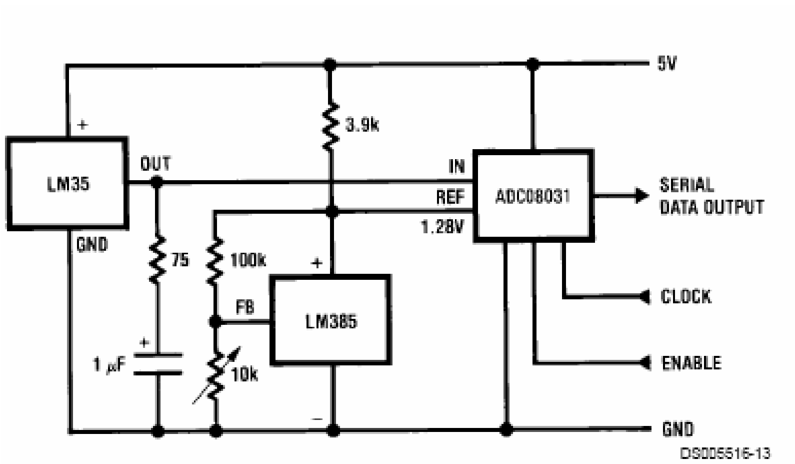


Fig.2.3.3 Temperature to digital converter (serial output)

MICROCONTROLLER
(89C52)

MICROCONTROLLER- AT89C52

The microcontroller used in our project is AT89C52. It has 8Kbytes of flash memory which is not present in AT89C51.

DESCRIPTION:

The AT89C52 is a low power , high performance CMOS 8-bit microcomputer with 8K 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 80C51 and 80C52 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 AT89C52 is a powerful microcomputer which provides a highly-flexible and cost effective solution to many embedded control applications.

BLOCK DIAGRAM:

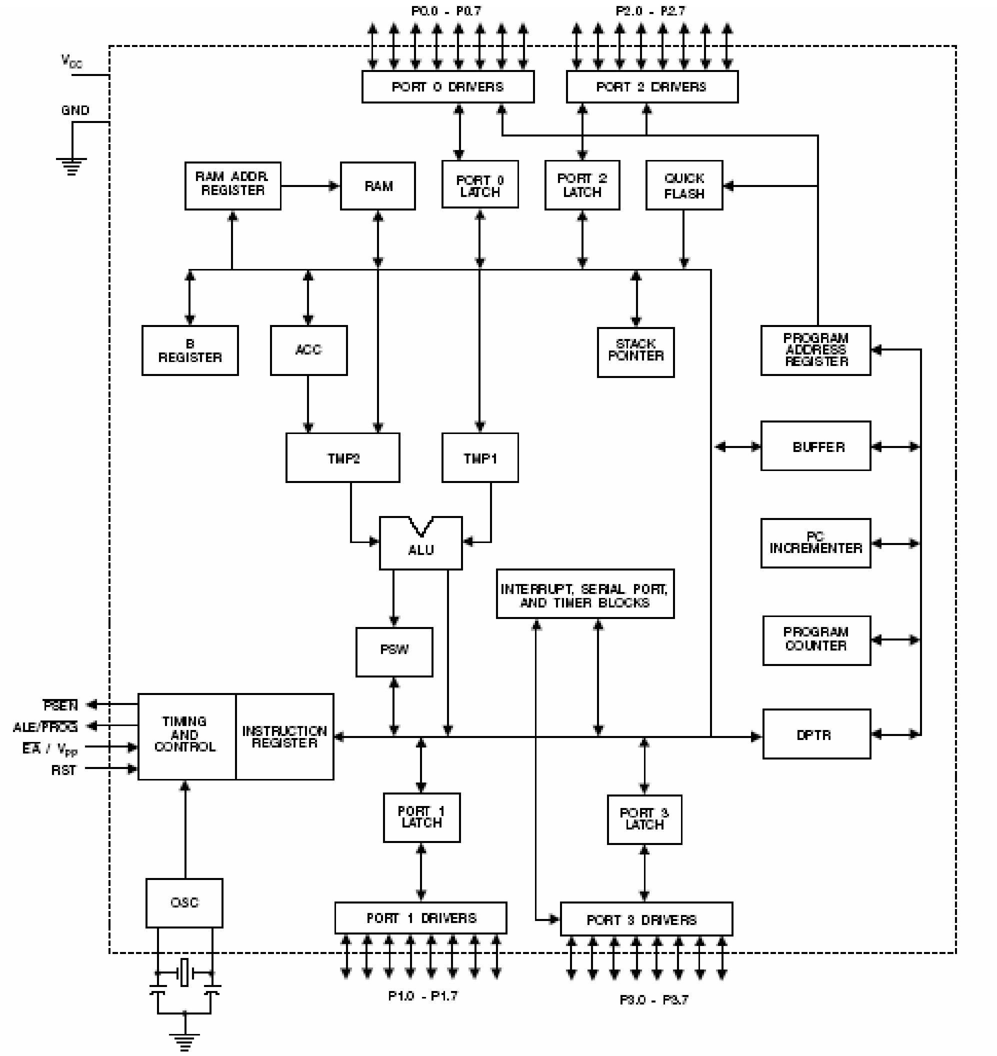


Fig.2.4.1 Block diagram of 89C52

The AT89C52 provides the following standard features:8k bytes of flash 256 bytes of ram, 32 I/O lines ,three 16-bit timers/counters, a six-vector two level interrupt architecture, a full duplex serial port ,on –chip oscillator , and clock circuitry . in addition , the AT89C52 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 , timers/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.

Port 0

Port 0 is an 8-bit open drain bi-directional I/O port . As an output port , each written to port 0 pins , the pins can be used as high- impedance inputs .

Port 0 can 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 external pull-ups.

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 pull-ups. 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 pull-ups and can be used as inputs. As inputs , Port 1 pins that are externally being pulled low will source current (Iil) because of the internal pull-ups .

In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX).

Port 1 also receives the low –order address bytes during flash programming and verification.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)

TABLE 1. Alternate functions of port pin1

Port 2

Port 2 is an 8-bit bi-directional I/O port with internal pull-ups . The Port 2 output buffers can sink/source four TTL inputs . when 1s are written to Port 2 pins ,they are pulled high by internal pull-ups and can be used as inputs . As inputs , Port 2 pins that are externally being pulled low will source current (Iil) because of the internal pull-ups .

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 , Port 2 uses strong internal pull-ups when emitting 1s . During accesses to external data memory that use 8-bit addresses (MOVX @ r1), Port 2 emits the contents of the P2 special function register .

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

Port 3

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

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 pull-ups and can be used as inputs. As inputs , Port 3 pins that are externally being pulled low will source current (Iil) because of the pull-ups .

Port 3 also serves the functions of various special features of the AT89C51.

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

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

TABLE 2. Alternate functions of port pin3

RST

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

ALE/$\overline{PROG}$

Address latch enable is an 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 .

$\overline{PSEN}$

Program store enable is the read strobe to external program memory . When the AT89C52 is executing code from external program memory $\overline{PSEN}$ is activated twice each machine cycle , except that $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{EA}$/VPP

External access enable . $\overline{EA}$ 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 , $\overline{EA}$ will be internally latched on reset .EA should be strapped to Vcc for internal program executions . This pin also receives the 12- volt programming enable voltage (Vpp) during flash programming when 12-volt programming is selected .

XTAL1

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

XTAL2

Output from the inverting oscillator amplifier .

SPECIAL FUNCTION REGISTERS:

A map of the on chip memory area called the special function register (SFR) is shown in table . note that not all the addresses are occupied addresses may not be implemented on the chip . Read accesses to these addresses will in general return random data , and write accesses will have an indeterminate effect .

User software should not write 1s to these locations , since they may be used in future products to invoke new features .In that case , the reset or inactive values of the new bits will always be 0.

AT89C52 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								0AFH
0A0H	P2 11111111								0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000				PCON 0XXX0000	87H

Fig2.4.2 SFR register

Timer 2 Registers Control and status bits are contained in registers T2CON and T2MOD for Timer 2 . The register pair (RCAP2H,RCAP2L) are the capture /reload registers for timer 2 in 16-bit capture mode or 16-bit auto –reload mode.

Interrupt Registers The individual interrupt enable bits are in the IE register . Two priorities can be set for each of six interrupt sources in the IP register .

T2CON – Timer/Counter 2 Control Register

T2CON Address = 0C8H Reset Value = 0000 0000B

Bit Addressable

Bit	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
	7	6	5	4	3	2	1	0

Symbol	Function
TF2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
RCLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
C/$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
CP/$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

TABLE 3. Timer /counter 2 control register

DATA MEMORY

The AT89C52 implements 256 bytes of on chip RAM . The upper 128 bytes occupy a parallel address space to the special function registers. That means the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space . When an instruction accesses an internal location above address 7FH, the address mode used in the instruction specifies whether the CPU accesses the upper 128 bytes of RAM or the SFR space . Instructions that use direct addressing access SFR space .

For example , the following direct addressing instruction accesses the SFR at location 0A0H(which is P2).

MOV 0A0H, #data

Instructions that use indirect addressing access the upper 128 bytes of RAM. For example , the following indirect addressing instruction , where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2(whose address is 0A0H).

MOV @R0, #data

Note that stack operations are examples of the indirect addressing , so the upper 128 bytes of data RAM are available as stack space.

TIMER 0 and 1:

Timer 0 and Timer 1 in the AT89C52 operate the same way as Timer 0 and Timer 1 in the AT89C51.

TIMER 2:

Timer 2 is a 16-bit Timer / counter that can operate as either a timer or an event counter . The type of operation is selected by bit **C/T2** in the SFR T2CON .

Timer 2 has three operating modes :

1.capture

2. Auto-reload(up or down counting)

3.Baud rate generator.

The modes are selected by bits in T2CON . Timer 2 consists of two 8-bit registers , TH2 and TL2 . In the timer function ,the TL2 register is incremented every machine cycle . Since a machine cycle consists of 12 oscillator periods , the count rate is 1/12 of the oscillator frequency .

UART:

The UART in the AT89C52 operates the same way as the UART in the AT89C51.

INTERRUPTS:

The AT89C52 has total of six interrupt vectors : two external interrupts (INT0 and INT1), three timer interrupts (Timers 0,1, and 2), and the serial port interrupt . These interrupts are shown in fig .

Each of these interrupt sources can be individually enabled or disabled by setting or cleaning a bit in special function register IE. IE also contains a global disable bit ,EA , which disables all interrupts at once.

Note that table shows that bit position IE.6 is unimplemented . In the AT89C51, bit position IE.5 is also unimplemented . User software should not write 1s to these bit positions , since they may be used in future AT89 products.

Timer 2 interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact , the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt , and that bit will have to be cleared in software .

The Timer 0 and Timer 1 flags , TF0 and TF1 , are set at S5P2 of the cycle in which the timers overflow .The values are then polled by the circuitry in the next cycle .However the Timer 2 flag , TF2, is set at S2P2 and is polled in the same cycle in which the timer overflows.

Interrupt Enable (IE) Register

(MSB) (LSB)

EA	–	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	Serial Port interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.
User software should never write 1s to unimplemented bits, because they may be used in future AT89 products.		

TABLE 4. Interrupt enable register

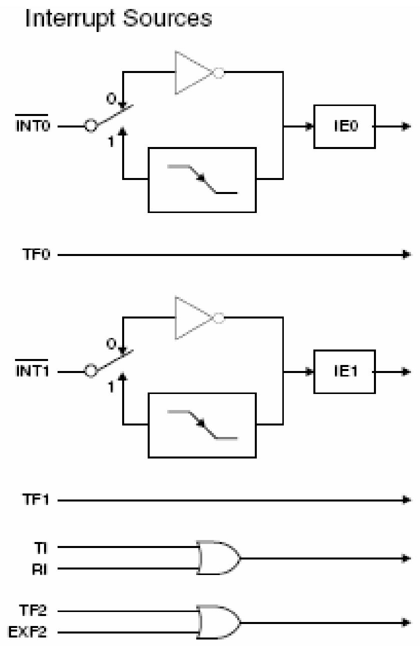


Fig2.4.3 Interrupt sources

OSCILLATOR CHARACTERISTICS:

XTAL1 and XTAL2 are the input and output , respectively , of an inverting amplifier that can be configured for use as on –chip oscillator as shown in fig.

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.

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 maximum and minimum voltage high and low time specifications must be observed.

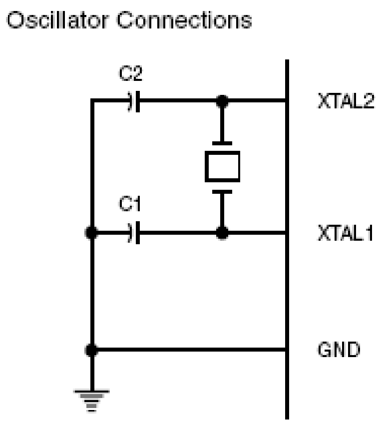
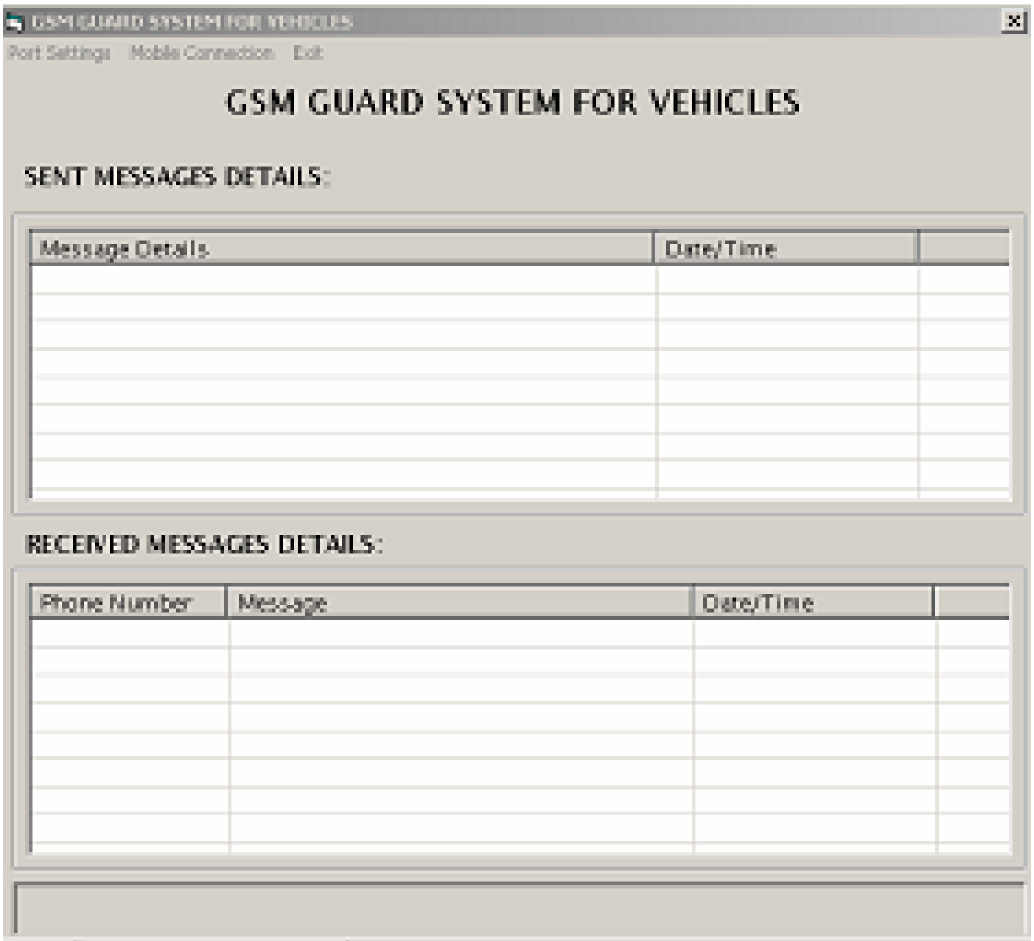
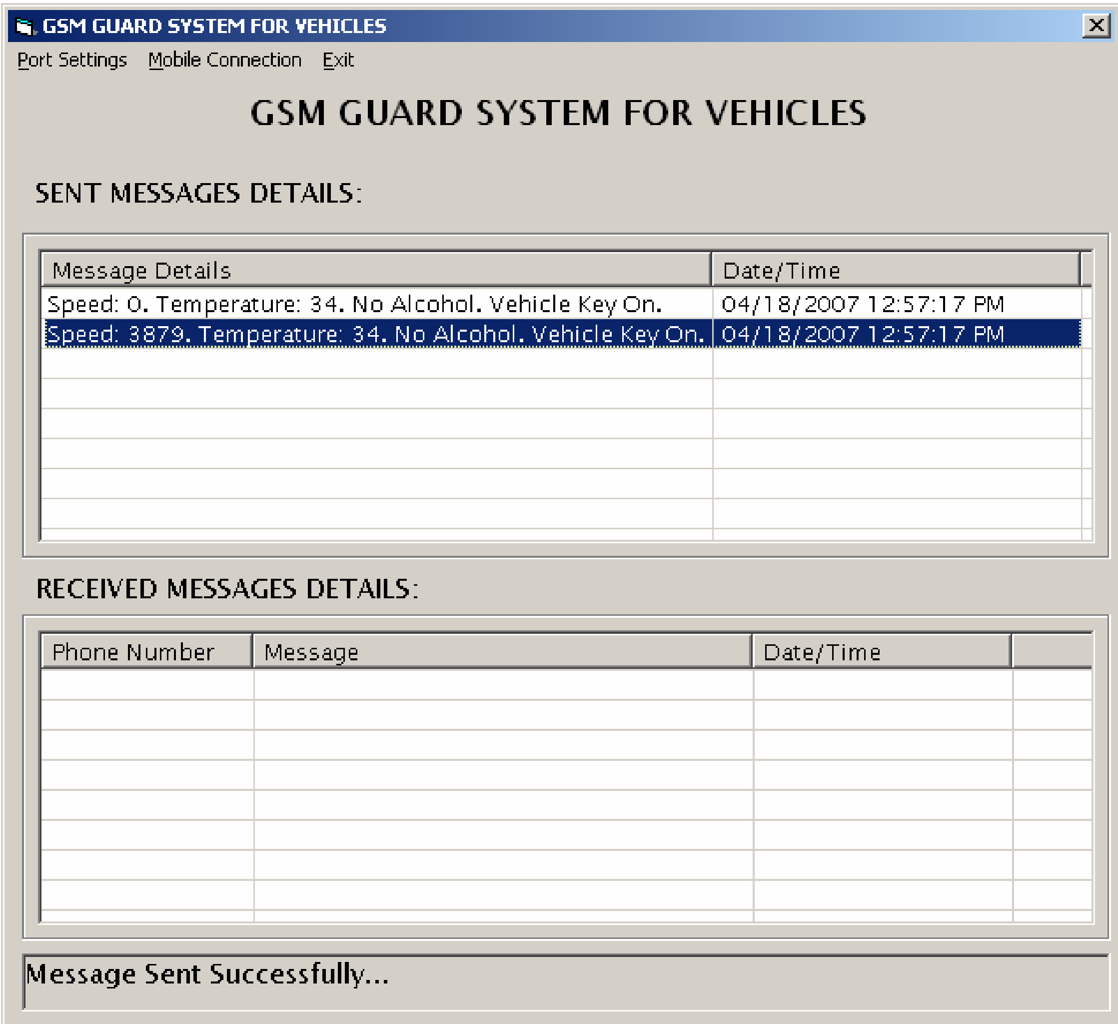


Fig.2.4.4 Oscillator connection

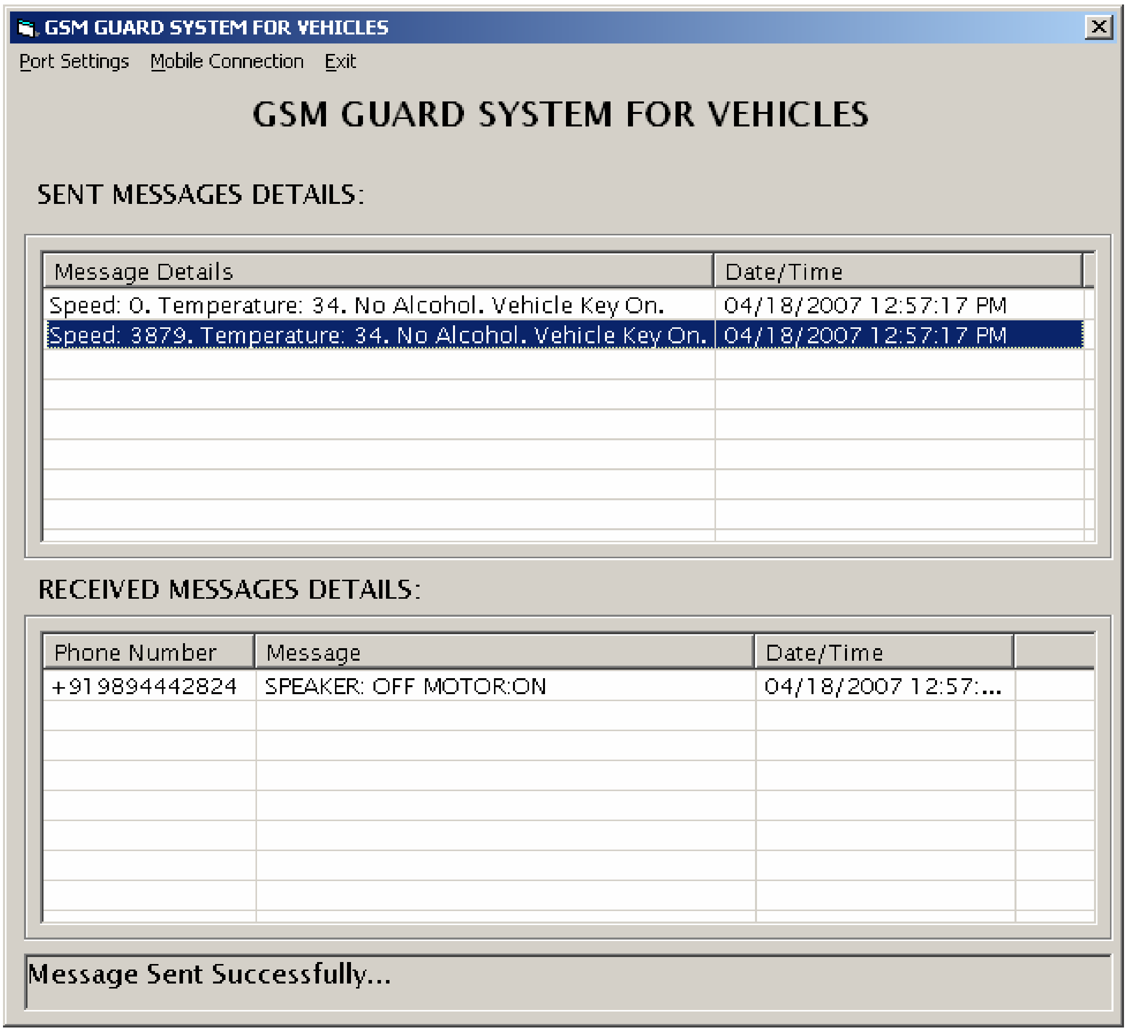
GRAPHICAL USER INTERFACE



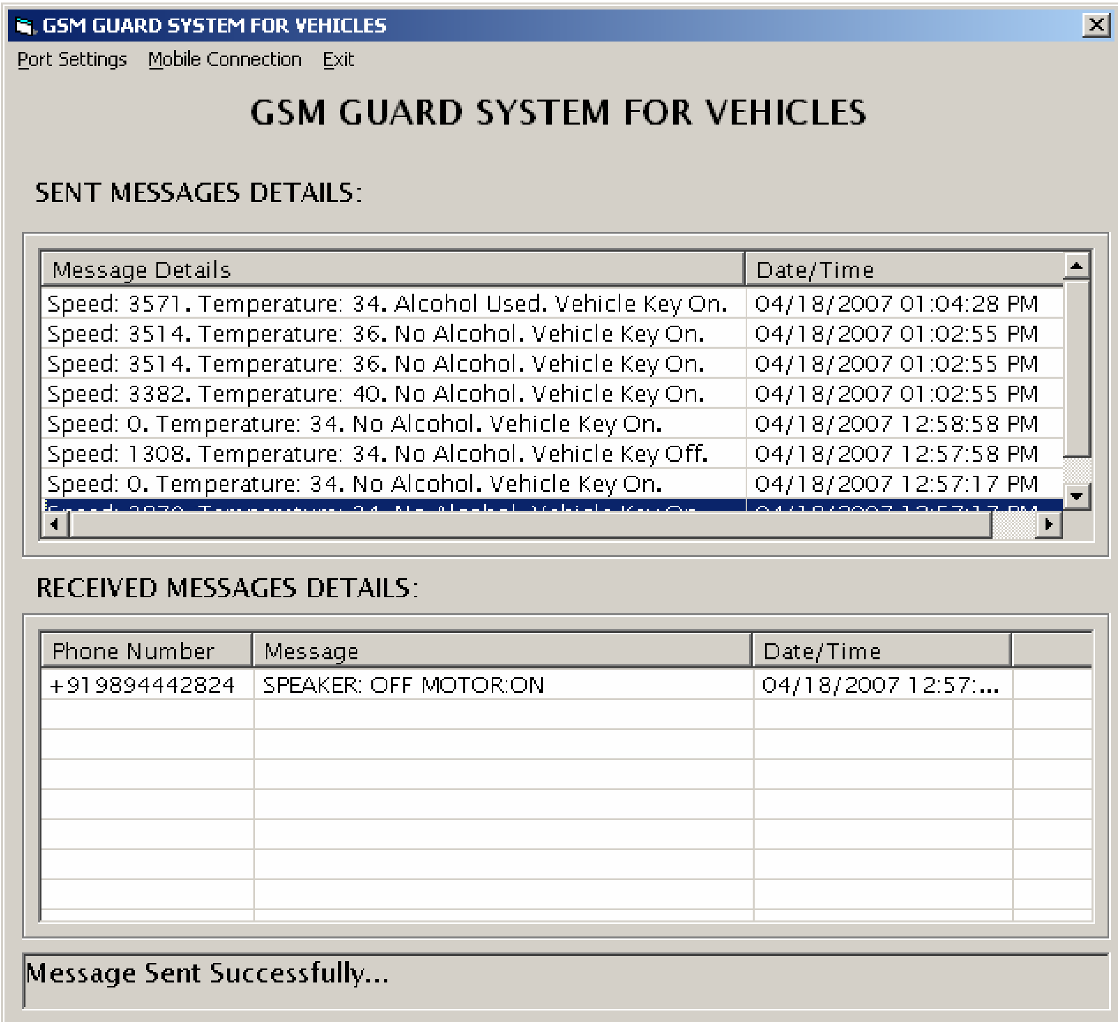
Screen before process



when speed exceeds 2000rpm,motor of the vehicle gets off automatically



Reply message to turn on the motor vehicle



Message sent from the guard system when alcohol is detected

CONCLUSION

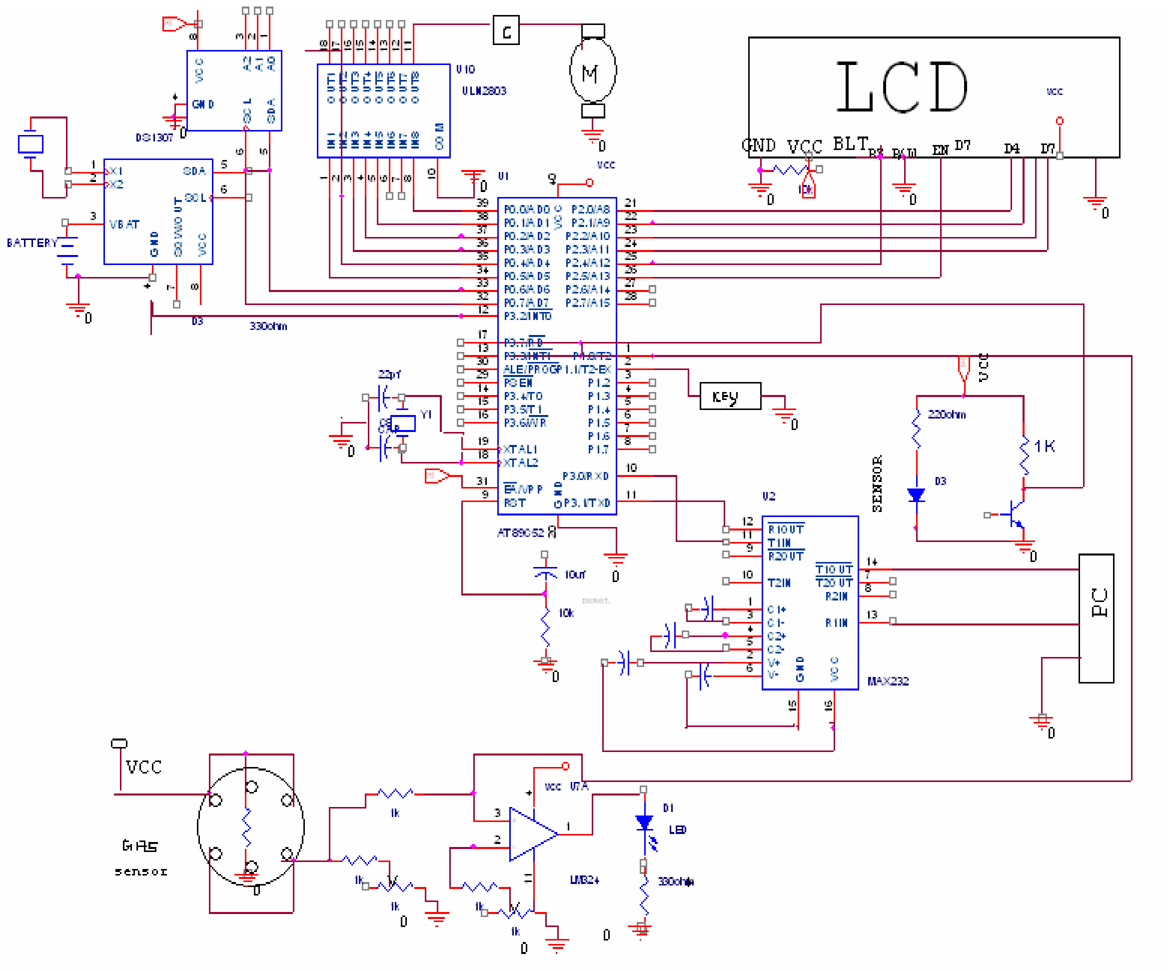
CONCLUSION

The problems that we are facing and the results of these problems . The problems that we are facing due to the over speed and driving after having alcohol are increasing desperately. In order to prevent these type of problems our GSM guard system are very much useful in avoiding the unnecessary vehicle accidents and it will safeguard the precious human life.

At present in call taxi’s wacky talkies are used for knowing the status of the vehicle. In future our project can be implemented in all types of four wheelers for continuously monitoring the status of the vehicle and thus preventing accidents.

APPENDIX

APPENDIX(1)-SCHEMATIC DIAGRAM OF THE PROJECT



APPENDIX(2)-CODING

****

```
#include<stdio.h>
#include<reg52.h>
```

```
#define ON 1
#define OFF 0
#define HIGH 1
#define LOW 0
#define LCD_DATA P2
sbit LCD_RS = P2^4;
sbit LCD_EN = P2^5;
#define uchar unsigned char
 #define ACK 0
#define NACK 1
#define ADDRTC 0xd0 /* I2C slave address */
#define DS1307 /* compile directive, modify as required */

bit once_speed,once_liquor,once_key,once_key_off,once_temp;
unsigned char  drunken='0',keystatus='0';
 //----------------------------------------


 //rx 3^0
 //tx 3^1                  //interrupts int0
//speed 3^3   //int1
```



```
sbit GO = P3^3;
sbit WRIT = P3^4;
          //interrupts int0
sbit alcohol_sensor= P3^6;
sbit recording1_1   =P3^7;
sbit car_stop       =P0^0;
sbit buzzar         =P0^1;
```

```
sbit car_key       =P0^2;
sbit press_key1     =P0^3;
sbit sendsms    =P0^4;



void readadc();
 bit chkbit;
 unsigned char tt;
 unsigned int sdelay;
 unsigned int longcut,store;
sbit scl = P0^6; /* I2C pin definitions */
sbit sda = P0^7;

void delayms(unsigned char dly);
void I2C_start();
void I2C_stop();
void I2C_write(unsigned char d);
unsigned char I2C_read(unsigned char);
void initialize();
void disp_clk_regs(void);
```



```
void I2C_start() /* ---------------------------------------------- */
{
sda = 1; scl = 1; /* Initiate start condition */
sda = 0;
}
void I2C_stop() /* ---------------------------------------------- */
```

```
{
sda = 0; sda = 0; sda = 0; sda = 0; /* Initiate stop condition */
scl = 1; scl = 1; sda = 1;
}
void I2C_write(uchar d) /* ----------------------------- */
{
uchar i;
scl = 0;
for (i = 1; i <= 8; i++)
{
sda = (d >> 7);
scl = 1;
d = d << 1; /* increase scl high time */
scl = 0;
}
sda = 1; /* Release the sda line */
scl = 0;
scl = 1;
if(sda) printf("Ack bit missing %02X\n",(unsigned int)d);
```



```
scl = 0;
}
uchar I2C_read(uchar b) /* ----------------------------------- */
{
uchar d, i;
sda = 1; /* Let go of sda line */
scl = 0;
```

```
for (i = 1; i <= 8; i++) /* read the msb first */
{
scl = 1;
d = d << 1;
d = d | (unsigned char)sda;
scl = 0;
}
sda = b; /* Hold sda low for acknowledge */
scl = 0;
scl = 1;
if(b == NACK) sda = 1; /* sda = 1 if next cycle is reset */
scl = 0;
sda = 1; /* Release the sda line */
return d;
}

void initialize() /* -- initialize the time and date using entries from stdin --
*/
/* Note: NO error checking is done on the user entries! */
{
```



```
uchar yr, mn, dt, dy, hr, min, sec;//, day;
I2C_start(); /* The following Enables the Oscillator */
I2C_write(ADDRTC); /* address the part to write */
I2C_write(0x00); /* position the address pointer to 0 */
I2C_write(0x00); /* write 0 to the seconds register, clear the CH bit */
I2C_stop();
printf("Enter the year (0-99): \n");
scanf("%bx", &yr);
```

```
printf("Enter the month (1-12): \n");
scanf("%bx", &mn);
printf("Enter the date (1-31): \n");
scanf("%bx", &dt);
printf("Enter the day (1-7): \n");
scanf("%bx", &dy);
printf("Enter the hour (1-23): \n");
scanf("%bx", &hr);
hr = hr & 0x3f; /* force clock to 24 hour mode */
printf("Enter the minute (0-59): \n");
scanf("%bx", &min);
printf("Enter the second (0-59): \n");
scanf("%bx", &sec);
I2C_start();
I2C_write(ADDRTC); /* write slave address + write */
I2C_write(0x00); /* write register address, 1st clock register */
I2C_write(sec);
I2C_write(min);
I2C_write(hr);
```



```
I2C_write(dy);
I2C_write(dt);
I2C_write(mn);
I2C_write(yr);
#if defined DS1307 || defined DS1338
{
I2C_write(0x10); /* enable sqwe, 1Hz output */
}
```

```
#endif
I2C_stop();
}



uchar Sec, Min, Hrs, Dte, Mon, Day, Yr, mil, pm;
void disp_clk_regs() /* ---------------------------------------- */
{
//printf("Yr Mn Dt Dy Hr:Mn:Sc\n");
if(1) /* Read & Display Clock Registers */
	{
	I2C_start();
	I2C_write(ADDRTC); /* write slave address + write */
	I2C_write(0x00); /* write register address, 1st clock register */
	I2C_start();
	I2C_write(ADDRTC | 1); /* write slave address + read */
	Sec = I2C_read(ACK); /* starts w/last address stored in register
	                                            pointer */
```



```
]Min = I2C_read(ACK);
	Hrs = I2C_read(ACK);
	Day = I2C_read(ACK);
	Dte = I2C_read(ACK);
	Mon = I2C_read(ACK);
	Yr = I2C_read(NACK);
	I2C_stop();
	if(Hrs & 0x40)
```

```
}

        mil = 0;
        else
        mil = 1;
        if(1) /* display every time seconds change */
        {
                if(mil)
                {
//              printf("%02bX/%02bX/%02bX %2bX ", Yr, Mon, Dte, Day);
                printf("<%02bX:%02bX-%02bX>", Hrs, Min, Sec);
                }
        else
        {
        if(Hrs & 0x20)
        pm = 'A';
        else
        pm = 'P';
        Hrs &= 0x1f; /* strip mode and am/pm bits */
//      printf("%02bx/%02bx/%02bx %02bx", Yr, (Mon & 0x1f), Dte, Day);
```



```
printf(" %<02bx:%02bx-%02bx> %cM", Hrs, Min, Sec, pm);
        }
        }
}
}
```

```
 void sleep()
{
        delayms(250);
        delayms(250);
        delayms(250);
        delayms(250);
}
void lcdinit();
void Lcd_Clear();
void Lcd_WriteChar(unsigned char b);
void Lcd_WriteString(unsigned char *);
void LcdAddress_Position(unsigned char,unsigned char);
void Lcd_DisplayNumber(unsigned int a, unsigned char nodig);
void ext0intr(void);

code const unsigned char Data1[]= {" Automatic Car  "};
code const unsigned char Data2[]= {" Monitor&Control"};
code const unsigned char Data3[]= {"    K.C.T       "};
code const unsigned char Data7[]= {"alcohol detected"};
```



```
code const unsigned char Data8[]= {"SPEED detected  "};
code const unsigned char Datax[]= {"1234567890abcdef"};
code const unsigned char Data9[]= {"rpm:       S:   "};
code const unsigned char Datab[]= {"alcohol detected"};
code const unsigned char Datac[]= {"Car Monitor T:  "};
code const unsigned char Datas[]= {"Car started:    "};
code const unsigned char Datat[]= {"Car stopped:    "};
code const unsigned char Datate[]= {"HIGH TEMPERATURE"};
```

```
code const unsigned char Datad[]= {" GUIDED BY   "};
code const unsigned char Datae[]= {"Mr.VENKATESH B.E"};
unsigned char ddata[4]={"    "};
unsigned char rx_data[4]={"    "};
unsigned char i;

 unsigned int alcohol,rain;
 unsigned char adc,adc2=0,gasleaked=0;
 unsigned int adcount=0;

void main(){
P3=0XFF;
buzzar=1;
car_stop=0;
 once_speed=0;
 once_liquor=0;once_key=0;once_key_off=0;
 once_temp=0;
```



```
    SCON = 0x50;    /* SCON */              /* setup serial port control */
    TMOD = 0x21;    /* TMOD */      /* hardware (2400 BR@12MHZ)*/
    TCON = 0x69;    /* TCON */
    TH1 =  0xE8;    /* TH1 */
     ES=1;
     ES=0;
     TI=1;
      printf ("\n\nC CLUSTER program\n\n");
```

```
ES=1;

  EX0 = 1;     // Enable External Interrupt 0
  IT0 = 1;     // Set Falling Edge for EX0
  EA = 1;
//  EX1 = 1;   // Enable External Interrupt 0
//  IT1 = 1;   // Set Falling Edge for EX0
  lcdinit();
  Lcd_Clear();
  LcdAddress_Position(1,0);
  Lcd_WriteString(Data1);
  sleep();
  LcdAddress_Position(2,0);
  Lcd_WriteString(Data2);
  sleep();
  LcdAddress_Position(2,0);
  Lcd_WriteString(Data3);   //sec
  sleep();
  Lcd_Clear();
```



```
LcdAddress_Position(1,0);
  Lcd_WriteString(Datad);   //sec
  LcdAddress_Position(2,0);
  Lcd_WriteString(Datae);   //sec
  sleep();
  Lcd_Clear();
  LcdAddress_Position(1,0);
  Lcd_WriteString(Data1);
```

```
LcdAddress_Position(2,0);
Lcd_WriteString(Data9);
LcdAddress_Position(1,0);
Lcd_WriteString(Datac);
 ES=0;
 TI=1;
//      initialize();
  ES=1;
  TR0=1;
  ET0=1;
  adcount=0;
  buzzar=1;
while(1)
   {
   adcount++;
   if (adcount==20)
      {
         adcount=0;
         EA=0;
```



```
         readadc();
         EA=1;
      }

        LcdAddress_Position(1,12);
        Lcd_DisplayNumber(adc2=adc*2,3);
   //--------------------------
        LcdAddress_Position(2,13);
```

```
Lcd_DisplayNumber(longcut,3);

if(chkbit==1)
    {

        LcdAddress_Position(2,4);
        Lcd_DisplayNumber(store,4);
        chkbit=0;
        if((store>2000)&&(once_speed==0))
        {

            ES=0;
            TI=1;
            printf("$%d&%d@%c%c#",
            store,(unsigned int) adc2,drunken,keystatus);
             disp_clk_regs();
             ES=1;
             once_speed= 1;
            LcdAddress_Position(1,0);
```



```
Lcd_WriteString(Data8);
            buzzar=0;
            car_stop=0;
            sleep();
            sleep();
            sleep();
            sleep();
            buzzar=1;
```

```
        }
    }

  if((car_key==0)&&(once_key==0))
    {

          keystatus='1';
          once_key=1;
          once_key_off=0;
          car_stop=1;

          LcdAddress_Position(1,0);
          Lcd_WriteString(Datas);

          ES=0 ;
          TI=1;
         printf("$%d&%d@%c%c#",
         store,(unsigned int)adc2,drunken,keystatus);
```



```
         disp_clk_regs();
         ES=1;
     }

        if((car_key==1)&&(once_key_off==0))
          {
                keystatus='0';
```

```
                once_key_off=1;
                once_key=0;
                car_stop=0;
                LcdAddress_Position(1,0);
                Lcd_WriteString(Data);
                ES=0;
                TI=1;
                printf("$%d&%d@%c%c#",
                store,(unsigned int)adc2,drunken,keystatus);
                disp_clk_regs();
                ES=1;
            }

        if((alcohol_sensor==0)&&(once_liquor==0))
          {
                drunken='1';
                once_liquor=1;
                buzzar=0;
                car_stop=0;
                LcdAddress_Position(1,0);
```



```
                Lcd_WriteString(Data7);
                ES=0;
                TI=1;
                printf("$%d&%d@%c%c#",
                store,(unsigned int)adc2,drunken,keystatus);
                disp_clk_regs();
                ES=1;
                sleep();
```

```
                sleep();
                sleep();
                sleep();
               buzzar=1;
            }

if((adc2>50)&&(once_temp==0))
 {
        once_temp=1;
        buzzar=0;
        car_stop=0;
        LcdAddress_Position(1,0);
        Lcd_WriteString(Datate);
        ES=0;
        TI=1;
        printf("$%d&%d@%c%c#",
        store,(unsigned int)adc2,drunken,keystatus);
        disp_clk_regs();
        ES=1;
```



```
        sleep();
        sleep();
        sleep();
        sleep();
        buzzar=1;
       }

if( press_key1==0)
```

```
      {
                  once_speed=0;
                  once_liquor=0;
                  once_key=0;
                  once_key_off=0;
                  once_temp=0;
                  sleep();
                  sleep();
                  sleep();
                  sleep();
                  sleep();
                  sleep();
                  sleep();
   }

if( sendsms==0)
      {
                  ES=0;
                  TI=1;
                  printf("$%d&%d@%c%c#",
```

```
                  store,(unsigned int)adc2,drunken,keystatus);
                  disp_clk_regs();
                  ES=1;
                  sleep();
                  sleep();
                  sleep();
                  sleep();
                  sleep();
```

```
                    sleep();
                    sleep();
                    sleep();
                }
              }
          }
 void ext1intr(void) interrupt 2 using 1
{

}
void ext0intr(void) interrupt 0
{
longcut++;
}
void tm0intr(void) interrupt 1
{
   sdelay++;
   if(sdelay==100)
    {
      sdelay=0;
```



```
       store=longcut;
       longcut=0;
         chkbit=1;
      }
    }

 void serial_isr() interrupt 4
```

```
{       TI=0;
        if(RI)
          {
                RI = 0;
                rx_data[i] = SBUF;
                if(rx_data[0] == '!')
                  {
                   i++;

                        if(i==4)
                          {
                                i=0;
                          rx_data[0]==0;
                          if (rx_data[3]=='*')
                                  {
                                        if(rx_data[1]=='0')
                                          buzzar  =1;
                                         else buzzar  =0;
                                        if(rx_data[2]=='0')
                                          car_stop =0;
```



```
                                        else car_stop =1;
                                         rx_data[0]==0;

                                        }
                               }
                          }
                  }
}
```

```
void lcdinit()
{
	LCD_RS = 0;			// write control bytes
	delayms(15);			// power on delay
	Lcd_WriteChar(0x02);	//PORTD = 0x02;  // attention //4 bit    mode
	delayms(5);
	Lcd_WriteChar(0x02);
	delayms(100);
	Lcd_WriteChar(0x02);
	delayms(5);
	Lcd_WriteChar(0x28);	// 4 bit mode,5*7 matrix,2 line mode
	Lcd_WriteChar(0x08);	// display off,underline off, blink off
	Lcd_WriteChar(0x0F);	// display on, blink curson on, underline on
	Lcd_WriteChar(0x06);	// display shift off,increment
}

void Lcd_WriteChar(unsigned char b)
{
```



```
	LCD_DATA = (LCD_DATA & 0xF0) |  (b >> 4);
	LCD_EN = 1;
	delayms(1);
	LCD_EN = 0;

	LCD_DATA = (LCD_DATA & 0xF0) | (b & 0x0F);
	LCD_EN = 1;
	delayms(1);
```

```
        LCD_EN = 0;
}

void LcdAddress_Position(unsigned char LineNo, unsigned char Position)
{
        LCD_RS = 0;
        if(LineNo == 1)
                {
                Lcd_WriteChar(0x80 + Position);
                }
        if(LineNo == 2)
                {
                Lcd_WriteChar(0xC0 + Position);
                }
        delayms(1);
}

void Lcd_Clear()
{
        LCD_RS = 0;
```



```
        Lcd_WriteChar(0x01);
        delayms(5);
}

void Lcd_WriteString(unsigned char *a)
{
   LCD_RS=1;
   while(*a)
```

```
   Lcd_WriteChar(*a++);
}

void Lcd_DisplayNumber(unsigned int a, unsigned char nodig)
{
unsigned char temp;
        if (a!=0)
        {
                ddata[0]=((a%10)+0x30);
                a=a/10;
                ddata[1]=((a%10)+0x30);
                a=a/10;
                ddata[2]=((a%10)+0x30);
                a=a/10;
                ddata[3]=((a%10)+0x30);
                a=a/10;
        }
        else
        {
```



```
                 ddata[0]=0x30;
                ddata[1]=0x30;
                ddata[2]=0x30;
                ddata[3]=0x30;
        }

        LCD_RS=1;
```

```
        for (temp=1;temp<=nodig;temp++)
        {
                Lcd_WriteChar(ddata[nodig-temp]);
        }
}



void readadc()
    {

        WRIT=0;
        WRIT=1;
                WRIT=1;
                adc=0;
        while(GO==1);
                WRIT=0;
                adc= P1;
                WRIT=1;
         }
```



```
void delayms(unsigned char dly)
{
        while(dly--)
        {
                B = 125;
                while(B--);
        }
```

```
}
```

APPENDIX(3)-CODING – VISUAL BASICS

```
Dim strData As String

Private Sub Command1_Click()
    Dim msg As String
    msg = "M1:OFF-M2:OFF"
    Call ReceiveMessage(122, msg)
End Sub
Private Function ReceiveMessage(ph As String, str As String)
    Dim Dev1 As Integer
    Dim Dev2 As Integer
    Dim Dev1Status As String
```



```
    Dim Dev2Status As String
    Dim MsgFormat As String
    Dim DIDX As Integer
    If Len(Trim(str)) < 15 And InStr(1, Trim(str), "-", vbTextCompare) > 0
Then
        DIDX = InStr(1, str, "-", vbTextCompare)
        If Mid(str, 1, 2) = "M1" Then
            If Mid(str, 4, (DIDX) - 4) = "ON" Then
```

```
            Dev1Status = "ON"
            Dev1 = 1
        ElseIf Mid(str, 4, (DIDX) - 4) = "OFF" Then
            Dev1Status = "OFF"
            Dev1 = 0
        End If
    End If
    If Mid(str, DIDX + 1, 2) = "M2" Then
        If Mid(str, DIDX + 4, (Len(str) - (DIDX + 4)) + 1) = "ON" Then
            Dev2Status = "ON"
            Dev2 = 1
        ElseIf Mid(str, (DIDX + 4), (Len(str) - (DIDX + 4)) + 1) = "OFF"
Then
            Dev2Status = "OFF"
            Dev2 = 0
        End If
    End If
    MsgFormat = "SPEAKER: " & Dev1Status & " MOTOR:" &
Dev2Status
    Call StoreReceivedMessage(ph, MsgFormat)
```



```
    Dim DS As String
    DS = "!" & CStr(Dev1) & CStr(Dev2) & "*"
    MSComm1.Output = DS
  Else
    lblStatus.Caption = "Received Message Format is Not Clear."
  End If
End Function
```

```
Private Sub Form_Load()
    portnumber = 2
    MobilePort = 1
    phNumber = "9994416670"

    Call InitializePort

    lvMessages.ListItems.Clear
    lvMessages.ColumnHeaders.Clear
    lvMessages.View = lvwReport
    lvMessages.ColumnHeaders.Add 1, , "Message Details", 5700
    lvMessages.ColumnHeaders.Add 2, , "Date/Time", 2400

    lvReceivedMsg.ListItems.Clear
    lvReceivedMsg.ColumnHeaders.Clear
    lvReceivedMsg.View = lvwReport
    lvReceivedMsg.ColumnHeaders.Add 1, , "Phone Number", 1800
    lvReceivedMsg.ColumnHeaders.Add 2, , "Message", 4250
    lvReceivedMsg.ColumnHeaders.Add 3, , "Date/Time", 2200
End Sub
```



```
Private Sub InitializePort()
    With MSComm1
        If .PortOpen = True Then .PortOpen = False
        .CommPort = portnumber
        .Settings = "1200,N,8,1"
        .InputLen = 0
        .RThreshold = 1
        .SThreshold = 1
```

```
        .InputMode = comInputModeText
        .InBufferSize = 256
        .OutBufferSize = 256
        .Handshaking = comNone
        .PortOpen = True
    End With
End Sub

Private Sub Form_Unload(Cancel As Integer)
 If Mobile1.State <> 0 Then
    Mobile1.Close
  End If
End Sub

Private Sub mnuConnect_Click()
    If ConnectCellPhone = True Then
        lblStatus.Caption = "Cell Phone Connected..."
    Else
        lblStatus.Caption = "Cell Phone not Connected..."
    End If
```



```
End Sub

Private Sub mnuDisconnect_Click()
If Mobile1.State <> 0 Then
    Mobile1.Close
    lblStatus.Caption = "Mobile Disconnected..."
End If
End Sub
```

```
Private Sub mnuExit_Click()
   End
End Sub


Private Sub mnuPort_Click()
   Form3.Show vbModal
   Call InitializePort
End Sub
Private Function ConnectCellPhone() As Boolean
Mobile1.Close
Mobile1.ComNumber = MobilePort
Mobile1.ConnectionMode = 0 'DAU-9P
If Mobile1.Open = True Then ConnectCellPhone = True Else
ConnectCellPhone = False
End Function
```



```
Private Function SendMessage(sMsg As String) As Boolean
Dim SendFlag As Boolean
Mobile1.SMSCenterNumber = Mobile1.GetDefaultSMSCenterNumber
SendFlag = Mobile1.SendSMSMessage(phNumber, sMsg, 167, False, False,
"")
If SendFlag Then
  SendMessage = True
Else
```

```
  SendMessage = False
End If
End Function
Private Function StoreMessage(msg As String)
   lvMessages.ListItems.Add 1, , msg
   lvMessages.ListItems(1).ListSubItems.Add 1, , Now
End Function
Private Function StoreReceivedMessage(phNumber As String, msg As
String)
   lvReceivedMsg.ListItems.Add 1, , phNumber
   lvReceivedMsg.ListItems(1).ListSubItems.Add 1, , msg
   lvReceivedMsg.ListItems(1).ListSubItems.Add 2, , Now
End Function

Private Sub Mobile1_OnSMSMessageReceived(ByVal Index As Long,
ByVal Time As Double, ByVal Text As String, ByVal PhoneNumber As
String, ByVal HasPicture As Boolean)
   Call ReceiveMessage(PhoneNumber, Trim(Text))
End Sub
```



```
Private Sub MSComm1_OnComm()
   Select Case MSComm1.CommEvent
      Case comEvReceive
         Do
            strData = strData & MSComm1.Input
         Loop Until Mid(strData, Len(strData), 1) = "#"
   End Select
```

```
    If InStr(1, strData, "$", vbTextCompare) > 0 And InStr(1, strData, "#",
vbTextCompare) > 0 Then
        Debug.Print strData
        MessageDetails (strData)
    Else
        Debug.Print strData
        strData = vbNullString
        Exit Sub
    End If
End Sub

Private Function MessageDetails(sData As String) As Boolean
strData = ""

Dim Speed As String
Dim Temp As String
Dim Alcohol As String
Dim KeyPosition As String
Dim IDX As Integer
Dim IDX1 As Integer
```



```
Dim Message As String
If InStr(1, sData, "&", vbTextCompare) > 0 And InStr(1, sData, "@",
vbTextCompare) > 0 Then
    IDX = InStr(1, sData, "&", vbTextCompare)
    IDX1 = InStr(1, sData, "@", vbTextCompare)
    Speed = Mid(sData, 2, IDX - 2)
    Temp = Mid(sData, IDX + 1, (IDX1 - IDX) - 1)
```

```
    If Mid(sData, IDX1 + 1, 1) = "0" Then Alcohol = "No Alcohol." Else:
Alcohol = "Alcohol Used."
    If Mid(sData, IDX1 + 2, 1) = "0" Then KeyPosition = "Vehicle Key Off."
Else: KeyPosition = "Vehicle Key On."
Else
    MsgBox "Message Not Properly Received. Please send it Again"
End If

Message = "Speed: " & Speed & ". Temperature: " & Temp & ". " &
Alcohol & " " & KeyPosition



If SendMessage(Message) = True Then
    Call StoreMessage(Message)
    lblStatus.Caption = "Message Sent Successfully..."
Else
    lblStatus.Caption = "Message Sent Failed..."
End If

End Function
```


Port Settings:

```
Private Sub cmdSavePort_Click()
If cboPort.Text <> vbNullString And cboMobilePort.Text <> vbNullString
And txtPh.Text <> vbNullString Then
    portnumber = Val(cboPort.Text)
    MobilePort = Val(cboMobilePort.Text)
    phNumber = txtPh.Text
```

```
    Unload Me
Else
    MsgBox "Please fill all the details."
    Exit Sub
End If
End Sub

Private Sub Form_Load()
Dim i As Integer
cboPort.Clear
txtPh.Text = "9994416670"
For i = 1 To 4
    cboPort.AddItem (i)
    cboMobilePort.AddItem (i)
Next
End S
```

Module

```
Public portnumber As Integer
Public MobilePort As Integer
Public phNumber As String
```


APPENDIX(4)-DATA SHEETS

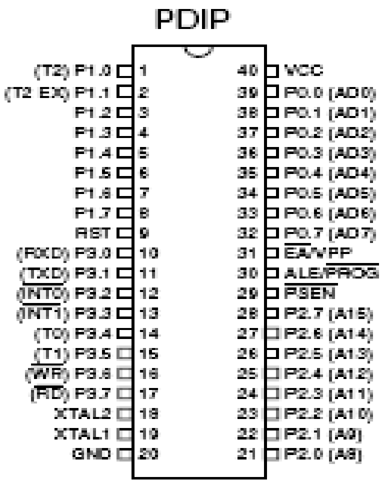
AT89C52

The microcontroller used in our project is 89C52 because it has 8Kbytes of flash memory compared to 89C51.The DC and AC characteristics of 89C52 are given below:

FEATURE:

1. Compatible with Mcs-51 products.
2. 8k Bytes of in system reprogrammable flash memory.
3. Endurance:1,000 write/Erase Cycles.
4. Fully static operation: 0 Hz to 24 MHz.
5. Three- level program memory lock.
6. 256* 8- bit internal RAM.
7. 32 programmable I/O lines.
8. Eight interrupt sources.
9. Three 16-bit timers/counters.
10. Programmable serial channel.
11. low power idle and power down modes.

PIN CONFIGURATION:



AT89C52

Atmel's Flash Microcontrollers

Device Name	Program Memory	Data Memory Bytes	16-bit Timers	Technology
AT89C1051	1K Flash	64 RAM	1	CMOS
AT89C2051	2K Flash	128 RAM	2	CMOS
AT89C51	4K Flash	128 RAM	2	CMOS
AT89C52	8K Flash	256 RAM	3	CMOS
AT89C55	20K Flash	256 RAM	3	CMOS
AT89S8252	8K Flash	256 RAM 2K EEPROM	3	CMOS
AT89S53	12K Flash	256 RAM	3	CMOS

DC Characteristics

The values shown in this table are valid for T_A = -40°C to 85°C and V_{CC} = 5.0V ± 20%, unless otherwise noted.

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC}-0.1	V
V_{IL1}	Input Low-voltage ($\overline{EA}$)		-0.5	0.2 V_{CC}-0.3	V
V_{IH}	Input High-voltage	(Except XTAL1, RST)	0.2 V_{CC}+0.9	V_{CC}+0.5	V
V_{IH1}	Input High-voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC}+0.5	V
V_{OL}	Output Low-voltage(1) (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
V_{OL1}	Output Low-voltage(1) (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
V_{OH}	Output High-voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 μA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 μA	0.75 V_{CC}		V
		I_{OH} = -10 μA	0.9 V_{CC}		V
V_{OH1}	Output High-voltage (Port 0 in External Bus Mode)	I_{OH} = -800 μA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 μA	0.75 V_{CC}		V
		I_{OH} = -80 μA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	μA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-650	μA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	μA
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode(1)	V_{CC} = 6V		100	μA
		V_{CC} = 3V		40	μA

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

AT89C52

AC Characteristics

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

External Program and Data Memory Characteristics

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

2. Sensor Characteristics

2-1 *Dependency on partial pressure of oxygen*

Figure 4 illustrates the relationship between oxygen pressure in the atmosphere (P_{O_2}) and the resistance of a typical TGS sensor in clean air. Note that reduced oxygen pressure will decrease the sensor's resistance.

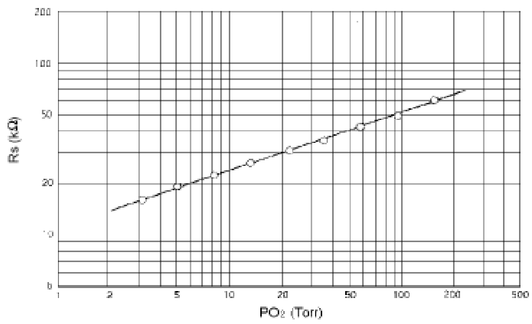


Fig. 4 - Typical dependency on P_{O_2}

2-2 *Sensitivity to gas*

According to the formula in *Section 1*, the relationship of sensor resistance to gas concentration is linear on a logarithmic scale within a practical range of gas concentration (from several ppm to several thousand ppm). Figure 5 shows a typical example of the relationship between sensor resistance and gas concentration. The sensor will show sensitivity to a variety of deoxidizing gases, with relative sensitivity to certain gases optimized by the formulation of sensing materials and operating temperature. Since actual sensor resistance values vary from sensor to sensor, typical sensitivity characteristics are expressed as a ratio of sensor resistance in various concentrations of gases (Rs) over resistance in a certain concentration of a target gas (Ro).

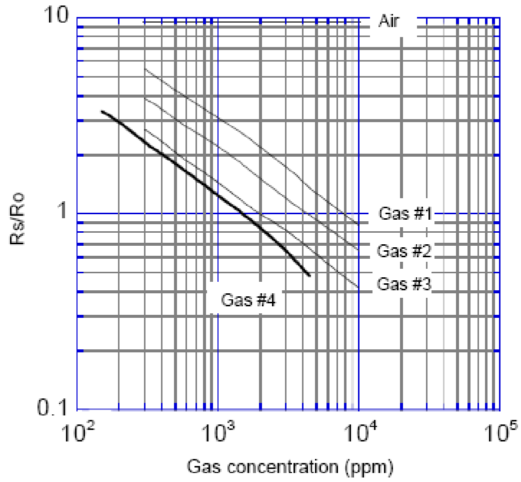


Fig. 5 - Typical sensitivity characteristics

2-3 *Sensor response*

Figure 6 demonstrates typical behavior when the sensor is exposed to and then removed from a deoxidizing gas. Sensor resistance will drop very quickly when exposed to gas, and when removed from gas its resistance will recover to its original value after a short time. The speed of response and reversibility will vary according to the model of sensor and the gas involved.

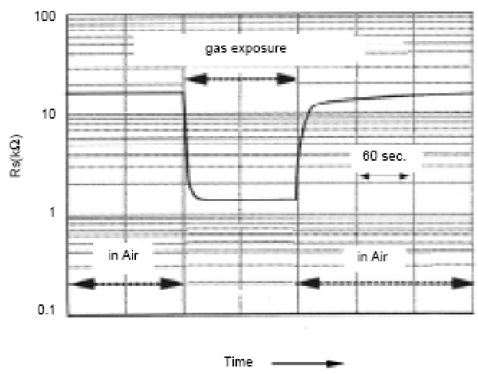


Fig. 6 - Typical sensor response

2-4 *Initial action*

As shown in Figure 7, all sensors exhibit a transient behavior referred to as "Initial Action" when stored unenergized and later energized in air. The Rs drops sharply for the first few seconds after energizing, regardless of the presence of gases, and then reaches a stable level according to the ambient atmosphere. The length of initial action depends on the atmospheric conditions during storage and length of storage and varies by sensor model. This behavior should be considered when designing a circuit since it may cause activation of an alarm during the first few moments of powering (*refer to Section 4-6*).

2-5 *Dependency on temperature and humidity*

The detection principle of TGS sensors is based on chemical adsorption and desorption of gases on the sensor's surface. As a result, ambient temperature will affect sensitivity characteristics by changing the rate of chemical reaction. In addition, humidity causes a decrease in Rs as water vapor adsorbs on the sensor's surface. Figure 8 shows a typical example of these dependencies. A compensation circuit for temperature dependency should be considered when using TGS sensors (*refer to Section 4-3*).

2-6 *Long term stability*

Figure 9 shows typical data of long term stability for TGS series sensors. Generally, TGS sensors show stable characteristics over time, making them suitable for maintenance-free operation.

2-7 *Heater voltage dependency*

TGS sensors are designed to show optimum senitivity characteristics under a certain constant heater voltage. Figure 10 shows a typical example of how gas sensitivity varies depending on heater voltage. Since the sensor has a heater voltage dependency, a constant regulated heater voltage must be supplied to the sensor according to specifications.

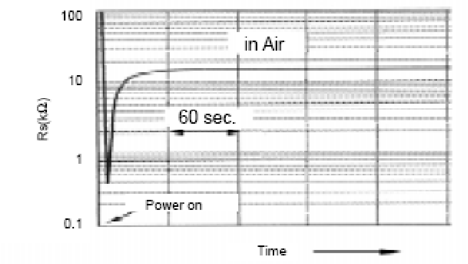


Fig. 7 - Typical initial action

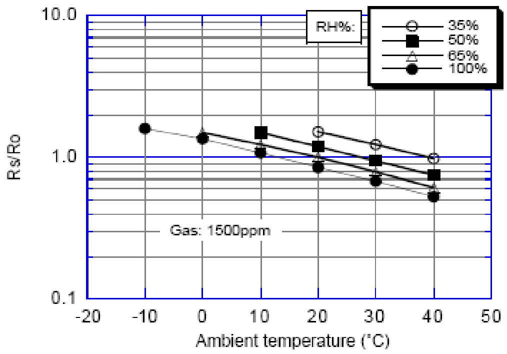


Fig. 8 - Typical temperature and humidity dependency

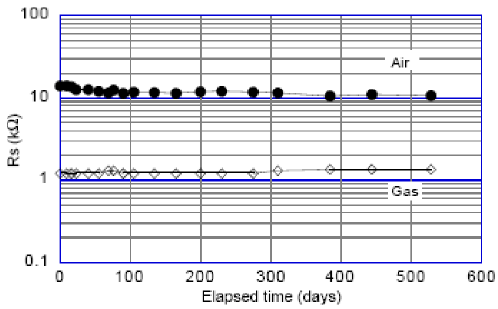


Fig. 9 - Typical long term stability

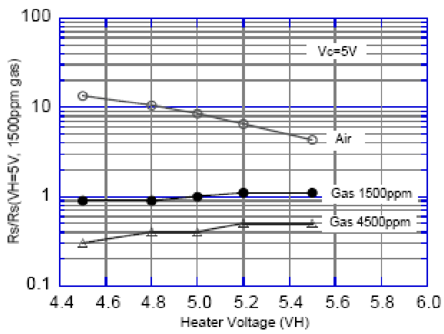


Fig. 10 - Typical heater voltage dependency

4. Circuit Design

4-1 *Load resistor (RL)*

Signal output is obtained through the RL which also acts as a sensor protector by regulating sensor power consumption (Ps) below the rated value for the sensor. Proper selection of the RL for an individual sensor enables the sensor to provide uniform characteristics so that users can apply the sensor under the best characteristics.

Figure 11 shows typical sensitivity characteristics of a sensor. Figure 12 shows gas concentration vs. output voltage (VRL) when the sensor is used in a circuit (such as that shown in Figure 14) along with various RL values (5kΩ, 2.5kΩ, 1kΩ).

Figure 13 shows the relationship between Rs/RL and VRL/Vc. At the point where Rs/RL equals 1.0, the slope of VRL/Vc reaches its maximum. At this point, the optimal resolution of signal at alarm concentration can be obtained. As a result, it is recommended to use an RL whose Rs/RL value is equal to 1.0 at the concentration to be detected. A variable resistor (RL) is recommended for optimal results.

4-2 *Signal processing*

The conventional method to process signal output is to use a comparator as shown in Figure 14. When the VRL exceeds a preset value (Vref), the comparator signal activates external equipment such as a buzzer or LED lamp.

Usage of a microprocessor is becoming more popular for signal processing. Microprocessors are commonly used and inexpensive, and they can perform the same function as a comparator in addition to other useful functions such as temperature dependency compensation, auto-calibration, etc.

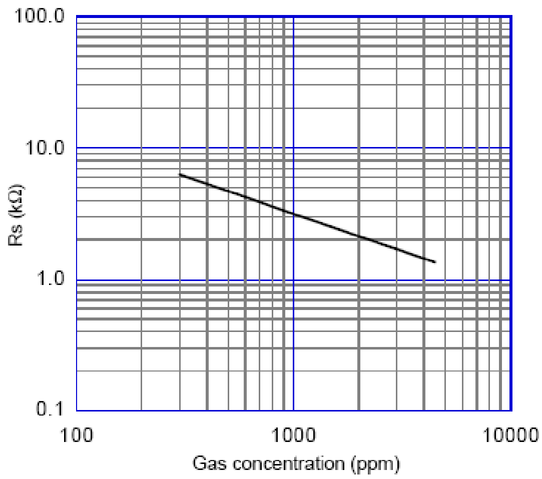


Fig. 11 - Sensitivity characteristics (Rs)

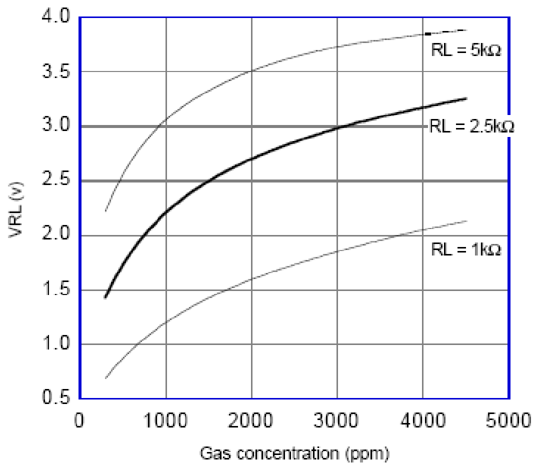


Fig. 12 - Sensitivity characteristics (VRL)

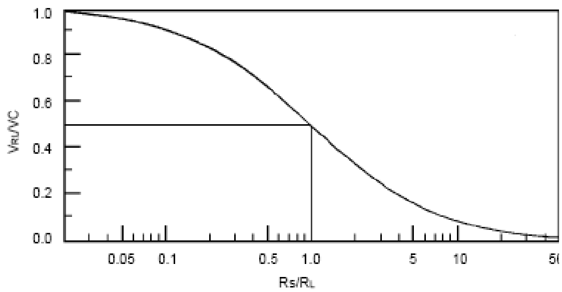


Fig. 13 - Relationship between Rs/RL and VRL/Vc

National Semiconductor

November 2000

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

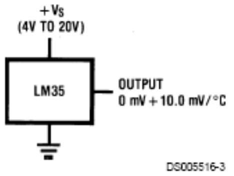
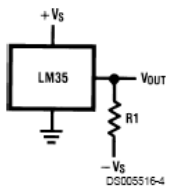


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



Choose $R_1 = -V_S/50\ \mu A$

V_{OUT}=+1,500 mV at +150°C

= +250 mV at +25°C

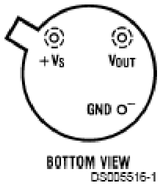
= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

LM 35 PRECISION CENTIGRADE TEMPERATURE SENSOR

Connection Diagrams

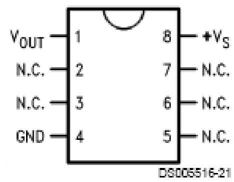
TO-46
Metal Can Package*



*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

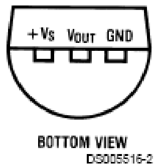
SO-8
Small Outline Molded Package



N.C. = No Connection

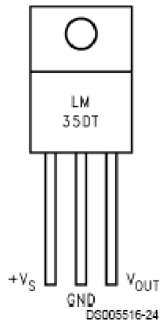
Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package



Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).

Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

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

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60°C to +180°C
TO-92 Package,	−60°C to +150°C
SO-8 Package,	−65°C to +150°C
TO-220 Package,	−65°C to +150°C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300°C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260°C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55°C to +150°C
LM35C, LM35CA	−40°C to +110°C
LM35D	0°C to +100°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A=+25°C	±0.2	±0.5		±0.2	±0.5		°C
	T_A=−10°C	±0.3			±0.3		±1.0	°C
	$T_A=T_{MAX}$	±0.4	±1.0		±0.4	±1.0		°C
	$T_A=T_{MIN}$	±0.4	±1.0		±0.4		±1.5	°C
Nonlinearity (Note 8)	$T_{MIN}≤T_A≤T_{MAX}$	**±0.18**		**±0.35**	**±0.15**		**±0.3**	°C
Sensor Gain (Average Slope)	$T_{MIN}≤T_A≤T_{MAX}$	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/°C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		mV/mA
	$T_{MIN}≤T_A≤T_{MAX}$	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	T_A=+25°C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25°C	56	67		56	67		µA
	V_S=+5V	**105**		**131**	**91**		**114**	µA
	V_S=+30V, +25°C	56.2	68		56.2	68		µA
	V_S=+30V	**105.5**		**133**	**91.5**		**116**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25°C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	**0.5**		**2.0**	**0.5**		**2.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	$T_J=T_{MAX}$, for 1000 hours	±0.08			±0.08			°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy, LM35, LM35C (Note 7)	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		°C
	T_A=−10°C	±0.5			±0.5		±1.5	°C
	$T_A=T_{MAX}$	±0.8	±1.5		±0.8		±1.5	°C
	$T_A=T_{MIN}$	±0.8		±1.5	±0.8		±2.0	°C
Accuracy, LM35D (Note 7)	T_A=+25°C				±0.6	±1.5		°C
	$T_A=T_{MAX}$				±0.9		±2.0	°C
	$T_A=T_{MIN}$				±0.9		±2.0	°C
Nonlinearity (Note 8)	$T_{MIN} \le T_A \le T_{MAX}$	**±0.3**		**±0.5**	**±0.2**		**±0.5**	°C
Sensor Gain (Average Slope)	$T_{MIN} \le T_A \le T_{MAX}$	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/°C
Load Regulation (Note 3) $0 \le I_L \le 1$ mA	T_A=+25°C	±0.4	±2.0		±0.4	±2.0		mV/mA
	$T_{MIN} \le T_A \le T_{MAX}$	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation (Note 3)	T_A=+25°C	±0.01	±0.1		±0.01	±0.1		mV/V
	$4V \le V_S \le 30V$	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25°C	56	80		56	80		µA
	V_S=+5V	**105**		**158**	**91**		**138**	µA
	V_S=+30V, +25°C	56.2	82		56.2	82		µA
	V_S=+30V	**105.5**		**161**	**91.5**		**141**	µA
Change of Quiescent Current (Note 3)	$4V \le V_S \le 30V$, +25°C	0.2	2.0		0.2	2.0		µA
	$4V \le V_S \le 30V$	**0.5**		**3.0**	**0.5**		**3.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	$T_J=T_{MAX}$, for 1000 hours	±0.08			±0.08			°C

Note 1: Unless otherwise noted, these specifications apply: $-55°C \le T_J \le +150°C$ for the LM35 and LM35A; $-40° \le T_J \le +110°C$ for the LM35C and LM35CA; and $0° \le T_J \le +100°C$ for the LM35D. V_S=+5Vdc and I_{LOAD}=50 µA, in the circuit of *Figure 2*. These specifications also apply from +2°C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-220 package is 90°C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in **boldface** apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/°C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in °C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of *Figure 1*.

Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ....-0.3V to +6V
Input Voltages
T_{IN}....................-0.3V to (V_{CC} - 0.3V)
R_{IN} (Except MAX220) ....±30V
R_{IN} (MAX220)....................±25V
T_{OUT} (Except MAX220) (Note 1) ....±15V
T_{OUT} (MAX220)....................±13.2V
Output Voltages
T_{OUT}....................±15V
R_{OUT}....................-0.3V to (V_{CC} + 0.3V)
Driver/Receiver Output Short Circuited to GND.........Continuous
Continuous Power Dissipation (T_A = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C)....842mW
18-Pin Plastic DIP (derate 11.11mW/°C above +70°C)....889mW
20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) ..440mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) ...696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C)......762mW
18-Pin Wide SO (derate 9.52mW/°C above +70°C)......762mW
20-Pin Wide SO (derate 10.00mW/°C above +70°C)....800mW
20-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
16-Pin CERDIP (derate 10.00mW/°C above +70°C).....800mW
18-Pin CERDIP (derate 10.53mW/°C above +70°C).....842mW
Operating Temperature Ranges
MAX2_ _AC_ _, MAX2_ _C_ _ ....0°C to +70°C
MAX2_ _AE_ _, MAX2_ _E_ _ ....-40°C to +85°C
MAX2_ _AM_ _, MAX2_ _M_ _ ....-55°C to +125°C
Storage Temperature Range ....-65°C to +160°C
Lead Temperature (soldering, 10sec) ....+300°C

Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
Note 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

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 in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All devices except MAX220		2	1.4		V
	MAX220: V_{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate				200	116	kb/s
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243 (continued)

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
TTL/CMOS Output Leakage Current	$\overline{SHDN}$ = V_{CC} or $\overline{EN}$ = V_{CC} ($\overline{SHDN}$ = 0V for MAX222), 0V ≤ V_{OUT} ≤ V_{CC}			±0.05	±10	µA
$\overline{EN}$ Input Threshold Low	MAX242			1.4	0.8	V
$\overline{EN}$ Input Threshold High	MAX242		2.0	1.4		V
Operating Supply Voltage			4.5		5.5	V
V_{CC} Supply Current ($\overline{SHDN}$ = V_{CC}), Figures 5, 6, 11, 19	No load	MAX220		0.5	2	mA
		MAX222/232A/233A/242/243		4	10	
	3kΩ load both inputs	MAX220		12		
		MAX222/232A/233A/242/243		15		
Shutdown Supply Current	MAX222/242	T_A = +25°C		0.1	10	µA
		T_A = 0°C to +70°C		2	50	
		T_A = -40°C to +85°C		2	50	
		T_A = -55°C to +125°C		35	100	
$\overline{SHDN}$ Input Leakage Current	MAX222/242				±1	µA
$\overline{SHDN}$ Threshold Low	MAX222/242			1.4	0.8	V
$\overline{SHDN}$ Threshold High	MAX222/242		2.0	1.4		V
Transition Slew Rate	C_L = 50pF to 2500pF, R_L = 3kΩ to 7kΩ, V_{CC} = 5V, T_A = +25°C, measured from +3V to -3V or -3V to +3V	MAX222/232A/233A/242/243	6	12	30	V/µs
		MAX220	1.5	3	30	
Transmitter Propagation Delay TLL to RS-232 (normal operation), Figure 1	t_{PHLT}	MAX222/232A/233A/242/243		1.3	3.5	µs
		MAX220		4	10	
	t_{PLHT}	MAX222/232A/233A/242/243		1.5	3.5	
		MAX220		5	10	
Receiver Propagation Delay RS-232 to TLL (normal operation), Figure 2	t_{PHLR}	MAX222/232A/233A/242/243		0.5	1	µs
		MAX220		0.6	3	
	t_{PLHR}	MAX222/232A/233A/242/243		0.6	1	
		MAX220		0.8	3	
Receiver Propagation Delay RS-232 to TLL (shutdown), Figure 2	t_{PHLS}	MAX242		0.5	10	µs
	t_{PLHS}	MAX242		2.5	10	
Receiver-Output Enable Time, Figure 3	t_{ER}	MAX242		125	500	ns
Receiver-Output Disable Time, Figure 3	t_{DR}	MAX242		160	500	ns
Transmitter-Output Enable Time ($\overline{SHDN}$ goes high), Figure 4	t_{ET}	MAX222/242, 0.1µF caps (includes charge-pump start-up)		250		µs
Transmitter-Output Disable Time ($\overline{SHDN}$ goes low), Figure 4	t_{DT}	MAX222/242, 0.1µF caps		600		ns
Transmitter + to - Propagation Delay Difference (normal operation)	t_{PHLT} - t_{PLHT}	MAX222/232A/233A/242/243		300		ns
		MAX220		2000		
Receiver + to - Propagation Delay Difference (normal operation)	t_{PHLR} - t_{PLHR}	MAX222/232A/233A/242/243		100		ns
		MAX220		225		

Note 3: MAX243 $R2_{OUT}$ is guaranteed to be low when $R2_{IN}$ is ≥ 0V or is floating.

REFERENCES

1. **A book on "89C52 MICROCONTROLLER " by John Peatman.**
2. **Data sheets of various IC'S.**
3. **www.carbodyengineering.com**
4. **www.visualbasic.com**