(1^{st} March’14 to 31st Suptember’14)

“Security System for High Lift Platform”

A

Project Report

Submitted to

NCRA • TIFR

GIANT METREWAVE RADIO TELESCOPE,

(National Centre for Radio Astrophysics),

(Tata Institute of Fundamental Research)

Under the Guidance of

Mr. S. Nayak.

Engineer-E

(GMRT-NCRA-TIFR)

Submitted by

BHOR RAHUL D.

B.E. (E&TC)

(Sinhgad Academy of Engineering, Kondhwa (Bk.), Pune)

CERTIFICATE

This is to certify that Mr. Bhor Rahul D., a graduate
Student of B.E (E&TC), Sinhgad Academy of Engineering,
Kondhwa (Bk.), Pune has undertaken a project entitled

"Security System for High Lift Platform"

at this institute from 01-03-2014 to 01-09-2014. He has done
very good work and completed the project successfully.

Mr. S. Nayak.
Engineer-E
(GMRT-NCRA-TIFR)

Date:

Place:

ACKNOWLEDGEMENTS

Working at Giant Meterwave Radio Telescope has been enriching experience for me. This project is result of many people's dedication, inspiration, guidance, knowledge and moral support. I would like to take this opportunity to thank them.

My foremost thanks to my guide, **Mr. S. Nayak**, who has helped me in every possible way in my project. It was really kind of him to devote his precious time and solve my difficulties. He was always eager to help me whenever I approached him. He not only cleared my concepts during this project, but also deepened my knowledge of the related topics.

I am highly grateful to the honorable Center Director of NCRA-GMRT, **Mr. SwarnaKanti Ghosh**; Dean **Mr. Yashwant Gupta**; for giving me opportunity to work in such a fabulous environment with all intellectuals around.

I feel delighted to express my sincere thanks to all the members of the Telemetry Lab, Mr. C. Satheesh, Mr. C. Kanade, Mr. B. Rajendran, Mr. M. Misal, Mr. B. Shete, all the members of the Analog Lab, Mr. N. D. Shinde, Mr. S. Phakatkar, Mr. A. Vishvakarma, Mrs. S. Mule, all the members of Mechanical Department, Mr. M. Patil, Mr. Nandi, Mr. P. Kasar and others for providing me guidance and help through various stages of the project.

Finally, I want to say thanks to all those who helped me directly or indirectly during this project and during my stay at GMRT Observatory.

Bhor Rahul D.

ABSTRACT

Security System for HLP (Heavy Lift Platform) is a microcontroller based system. It will be operated on battery present in the HLP. It consists of control circuit, GSM modem, Alarm, Siren, LCD display, electronic lock and sensors like PIR (Passive Infrared) sensor, Smoke/Fire detector, door closer sensor.

Aim of this project is to provide security to the HLP whenever it is parked near to the antenna base or somewhere on its way. This system will continuously keep checking status of all the connected sensors. Whenever system detects change in output of any sensor, it will inform to maximum five people by sending alert message on their mobile numbers. It will also turn on Alarm for some time and then Siren to indicate that something wrong is happening with the HLP. It will alert people near to the HLP.

With the help of GSM modem we can send commands to this system through SMS to control it remotely. We can install it at many places for providing security. The Microcontroller used in this unit is P89V51RD2BN. It supports in system programing and the GSM modem is SIM900A. These are easily available in the market.

CONTENTS

List of chapters

Chapter Name **Page No.**

Chapter Name **Page No.**

List of Figures

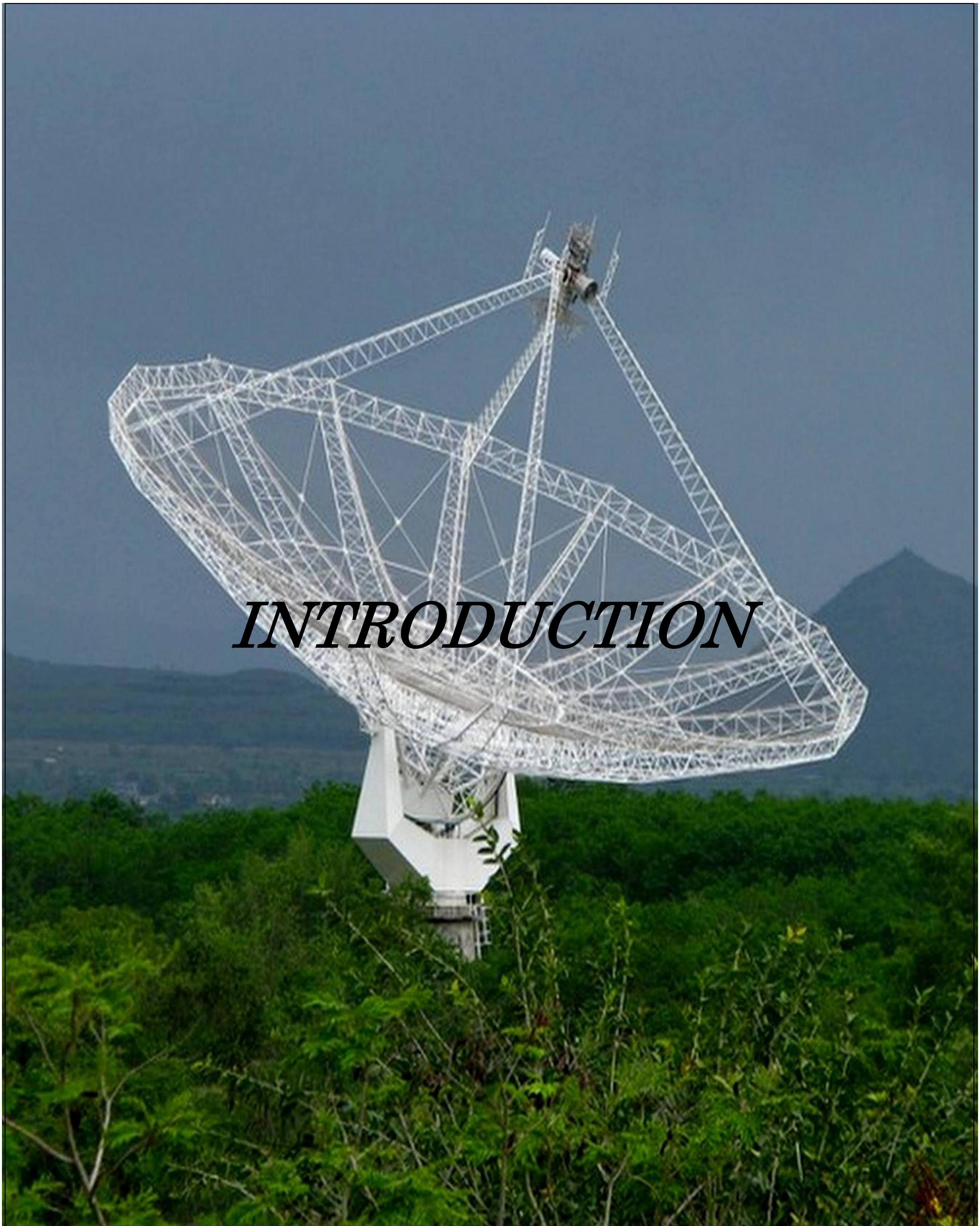
Figure Name **Page No.**

Figure Name **Page No.**

List of Tables

Table Name **Page No.**

INTRODUCTION



1.1 Introducing GMRT:

Fig. 1.1: GMRT antenna at twilight

NCRA has set up a unique facility for radio astronomical research using the metrewavelengths range of the radio spectrum, known as the Giant Metrewave Radio Telescope (GMRT), it is located at a site about 80 km north of Pune. GMRT consists of 30 fully steerable gigantic parabolic dishes of 45m diameter each spread over distances of up to 25 km. GMRT is one of the most challenging experimental programs in sciences undertaken by Indian scientists and engineers.

1.1.1 WHY METRE WAVELENGTH:

The metre wavelength part of the radio spectrum has been particularly chosen for study with GMRT because man-made radio interference is considerably lower in this part of the spectrum in India. Although there are many outstanding astrophysics problems which are best studied at metre wavelengths, there has, so far, been no large facility anywhere in the world to exploit this part of the spectrum for astrophysical research.

1.1.2 SITE:

The Site for GMRT, about 10 km east of Narayangaon town on the Pune-Nasik highway, was selected after an extensive search in many parts of India, considering several important criteria such as low man-made radio noise, availability of good communication, vicinity of industrial, educational and other infrastructure and, a geographical latitude sufficiently north of the geomagnetic equator in order to have a reasonably quiet ionosphere and yet be able to observe a good part of the southern sky as well.

1.1.3 ANTENNA CONFIGURATION:

The number and configuration of the dishes was optimized to meet the principal astrophysical objectives which require sensitivity at high angular resolution as well as ability to image radio emission from diffuse extended regions. Fourteen of the thirty dishes are located more or less randomly in a compact Central Array in a region of about 1 sq. km. The remaining sixteen dishes are spread out along the 3 arms of an approximately `Y'-Shaped configuration over a much larger region, with the longest interferometric baseline of about 25 km.

The multiplication or correlation of radio signals from all the 435 possible pairs of antennas or interferometers over several hours will thus enable radio images of celestial objects to be synthesized with a resolution equivalent to that obtainable with a single gigantic dish 25 kilometer in diameter! The array will operate in six frequency bands centered around 50, 153, 233, 325, 610 and 1420 MHz. All these feeds provide dual polarization outputs. In some configurations, dual-frequency observations are also possible.

The highest angular resolution achievable will range from about 60 arc sec at the lowest frequencies to about 2 arc sec at 1.4 GHz.

1.1.4 DESIGN BREAKTHROUGH:

GMRT is an indigenous project. The construction of 30 large dishes at a relatively small cost has been possible due to an important technological breakthrough achieved by Indian Scientists and Engineers in the design of light-weight, low-cost dishes. The design is based on what is being called the **`SMART'** concept - for **S**tretch **M**esh **A**ttached to **R**ope **T**russes.

The dish has been made light-weight and of low solidity by replacing the conventional back-up structure by a series of rope trusses (made of thin stainless steel wire ropes) stretched between 16 parabolic frames made of tubular steel. The wire ropes are tensioned suitably to make a mosaic of plane facets approximating a parabolic surface. A light-weight thin wire mesh (made of 0.55 mm diameter stainless steel wire) with a grid size varying from 10 X 10 mm in the central part of the dish to 20 X 20 mm in the outer parts, stretched over the rope truss facets forms the reflecting surface of the dish. The low-solidity design cuts down the wind forces by a large factor and is particularly suited to Indian conditions where there is no snowfall in the plains. The overall wind forces and the resulting torques for a 45-m GMRT dish are similar to those for only a 22-m dish of conventional design, thus resulting in substantial savings in cost.

The dish is connected to a `cradle' which is supported by two elevation bearings on a yoke placed on a 3.6 m diameter slewing-ring bearing secured on the top of a 15 metre high concrete tower. The weight of the disk is about 80 tones and the counter-weight is about 40 tones. The dishes have alt-azimuth mount. The salient parameters and specifications of each dish are summarized in the Table.

Fig. 1.2: Few antennas at the Central Square

1.2 GMRT Antenna Specifications:

The large size of the parabolic dishes implies that GMRT will have over three times the collecting area of the Very Large Array (VLA) in New Mexico, USA which consists of 27 antennas of 25 m diameter and is presently the world's largest aperture synthesis telescope operating at centimeter wavelengths. At 327 MHz, GMRT will be about 8 times more sensitive than VLA because of the larger collecting area, higher efficiency of the antennas and a substantially wider usable bandwidth because of the low level of man-made radio interference in India.

1.3 Electronic Frontends and Backends:

Apart from the novel low-cost design of the parabolic dishes, the instrument has state-of-the-art electronics systems developed indigenously and consisting of the following main sub units.

- Antenna feeds at six different frequency bands between 50 MHz and 1500 MHz, having good polarization characteristics as well as simultaneous multiband operation.
- Low-noise amplifiers, local oscillator synthesizers, mixers, IF amplifiers.
- Optical fibers linking the entire array with the CEB. These are used both for the telemetry signals and local oscillator phase reference communication between the CEB and each antenna base.
- A digital 2,30,000-channel FX-type correlator providing up to 128 spectral channels and covering a maximum bandwidth of 32 MHz

1.4 RF Front End Systems:

Giant Meter-wave Radio Telescope (GMRT) Front Ends have been designed to operate at 5 frequency bands centered at 50 MHz, 150 MHz, 235 MHz, 327 MHz, 610 MHz and L-Band extending from 1000 to 1450 MHz. The low noise front end of the receiving system of GMRT has been designed to receive dual polarization. The higher frequency L-Band has dual linear polarization channels (Vertical and Horizontal polarization) and they have been named CH1 and CH2 respectively. The front end system has flexibility to be configured for either dual polarization observation at a single frequency band or single polarization observation at two different frequency bands. The polarization channels can be swapped whenever required.

1.5 Analog Backend system:

The Analog Backend system of the GMRT is the penultimate system of processing the data. The Analog Backend System has been divided into Intermediate Frequency System (IF System), Local Oscillator Synthesizer (LOS) and Local Oscillator Reference (LOR) System and Base Band System.

1.6 Digital Backend System:

The digital backend is the section of the G.M.R.T. which deals with the digital conversion, manipulation and storage of the analog signals coming from the baseband system. The signal from the antenna after going through the baseband system is analog in form. This signal needs to be converted into digital form , so that it could easily be worked upon using digital signal processing. The signal after being converted into digital form is processed through FX Correlator to generate cross amplitude and phase information among the 30 antennas to synthesize the Stokes Parameters. The output from the correlator (FFT subsystem) are given in parallel to the

Array Combiner for generating the Incoherent Array (IA) and Phased Array (PA) outputs for pulsar observations.

1.7 Sentinel System:

GMRT consists of 30 antennas spreading over an area of 50 square kilometers. All the Antenna Shells are equipped with various intelligent sub-systems for providing the best results to this Radio Observatory. Hence it is a prime responsibility for providing safety and protections to all these sub-systems especially in the remote antenna shells from intrusion, fire, smoke, over temperature etc. All these parameters are getting monitored from the remote antenna shells to the Control Room all around the clock. GMRT Sentinel System is taking the full responsibility for monitoring and controlling the above parameters for smooth operation of GMRT.

1.7.1 GMRT Sentinel System consists of:

1. GMRT Antenna Temperature Control and Monitor

2. GMRT Intruder Alarm System

3. GMRT Smoke &Fire Alarm System

4. Correlator Over Temperature Alarm System

5. Electronic Door Closer

6. GMRT Closed Circuit Television Network

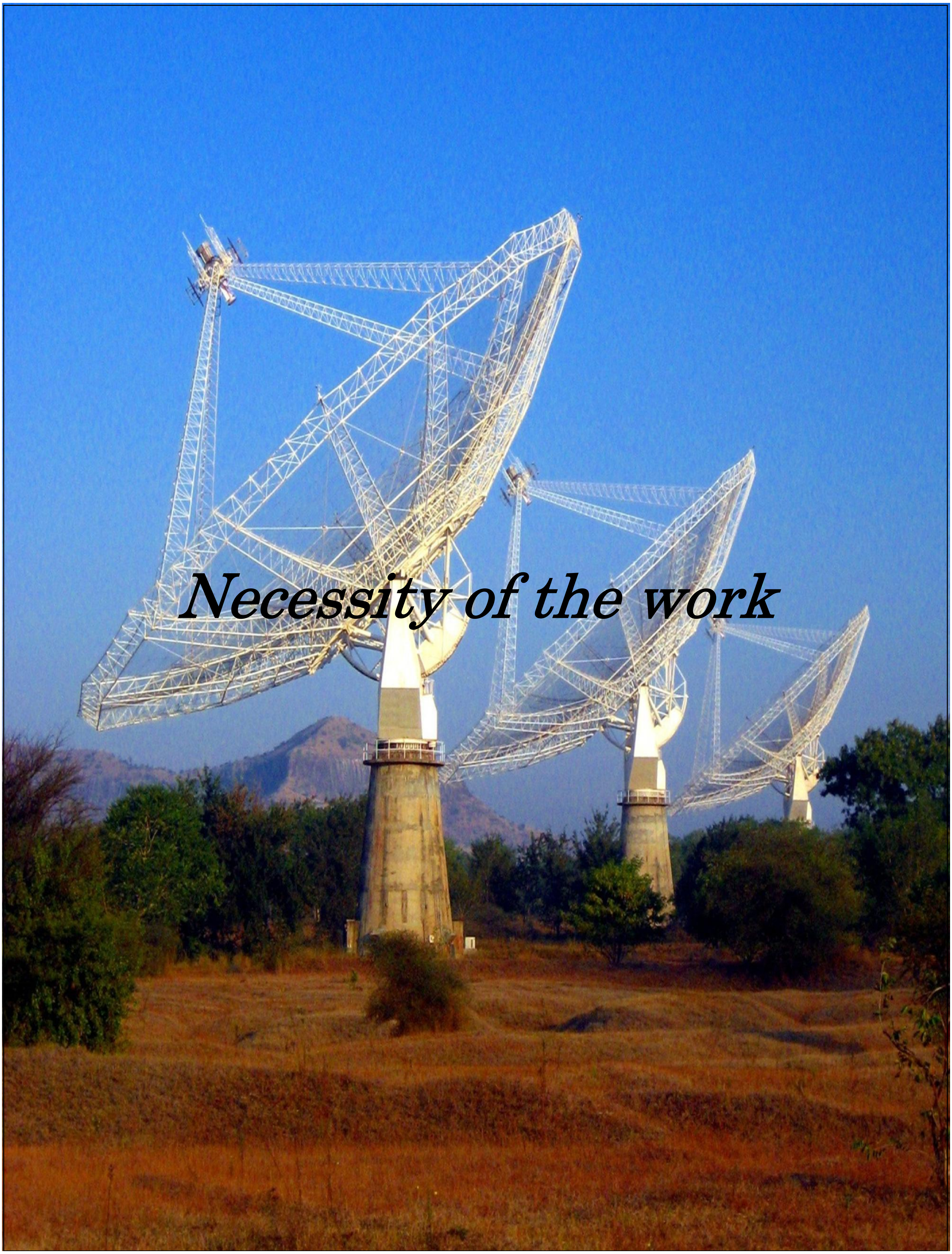
7. Hot-line Telephony System

8. RF Shielding & Measurements

9. NCRA-GMRT Video Conferencing

1.8 Servo System for GMRT Antenna:

The Giant Meterwave Radio Telescope, an aperture synthesis array consisting of 30 fully steerable parabolic dishes of 45 meter diameter each. Motion of these giant antennas need to be controlled by a precession control system. Pointing of the antennas should be accurate i.e. the radio source, antenna focused and the antenna center should be aligned. The GMRT servo system has designed with three nested control loops to achieve the pointing accuracy of (1 or 2) arc minites RMS for wind speed less than 20 km/ph. Because of high weight alt-azimuth mount is most favorable approach for positioning the dish antenna. Here the elevation axis sits on the azimuth drive. The elevation drive moves antenna up and down directions while azimuth drive moves antenna in clockwise & counter-clockwise direction. Hence enabling the antenna to point anywhere in the sky.



Necessity of the work

Necessity of the Security System for HLP:

In GMRT technical staff often has to do maintenance work on the antennas. They have to do some work with the feed positioning system of the antenna, electrical connections, motors, repairing of the antenna systems. They need some lifting platform to go there with some tools and devices. That’s why the HLP (High Lift Platform) is used in GMRT for doing work on antenna and lift some tools and devices. Sometimes we have to park it near to the antenna or if there is some problem in it we have to park it somewhere on its way away from the GMRT campus.

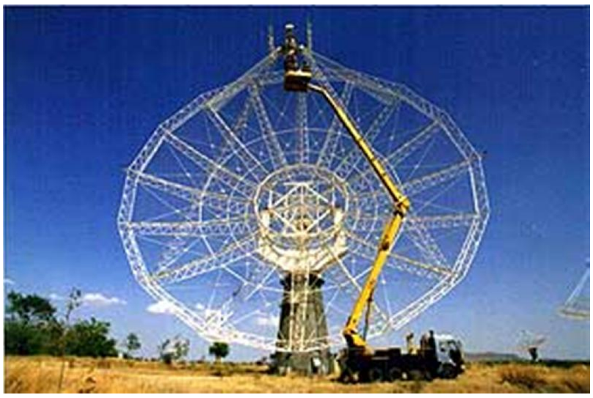
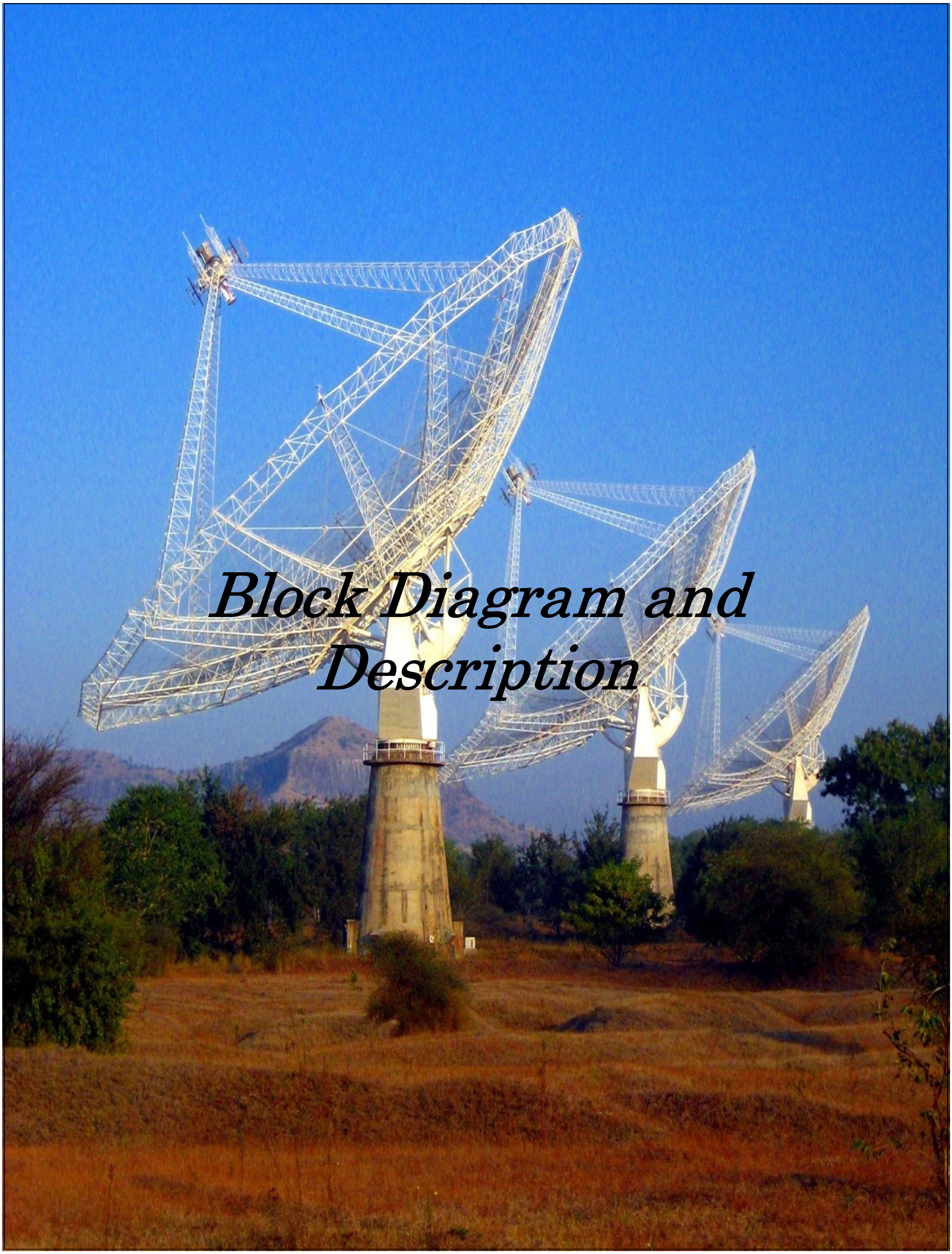


Fig. 2.1: GMRT Antenna with HLP.

To keep HLP safe we have to install such a security system which can alert people while something wrong is happening to the HLP. While designing the security system for HLP we have to give more priority to the safety of battery, diesel and driver’s cabin. To alert people nearby to the HLP we have to use Alarm and Siren. But we also have to use something that can alert people if HLP is parked at some remote place. So that, we have to use a long distance wireless communication medium. We decided to use GSM modem which can alert people by sending SMS or making the call on some mobile numbers.



Block Diagram and Description

3.1 Block Diagram:

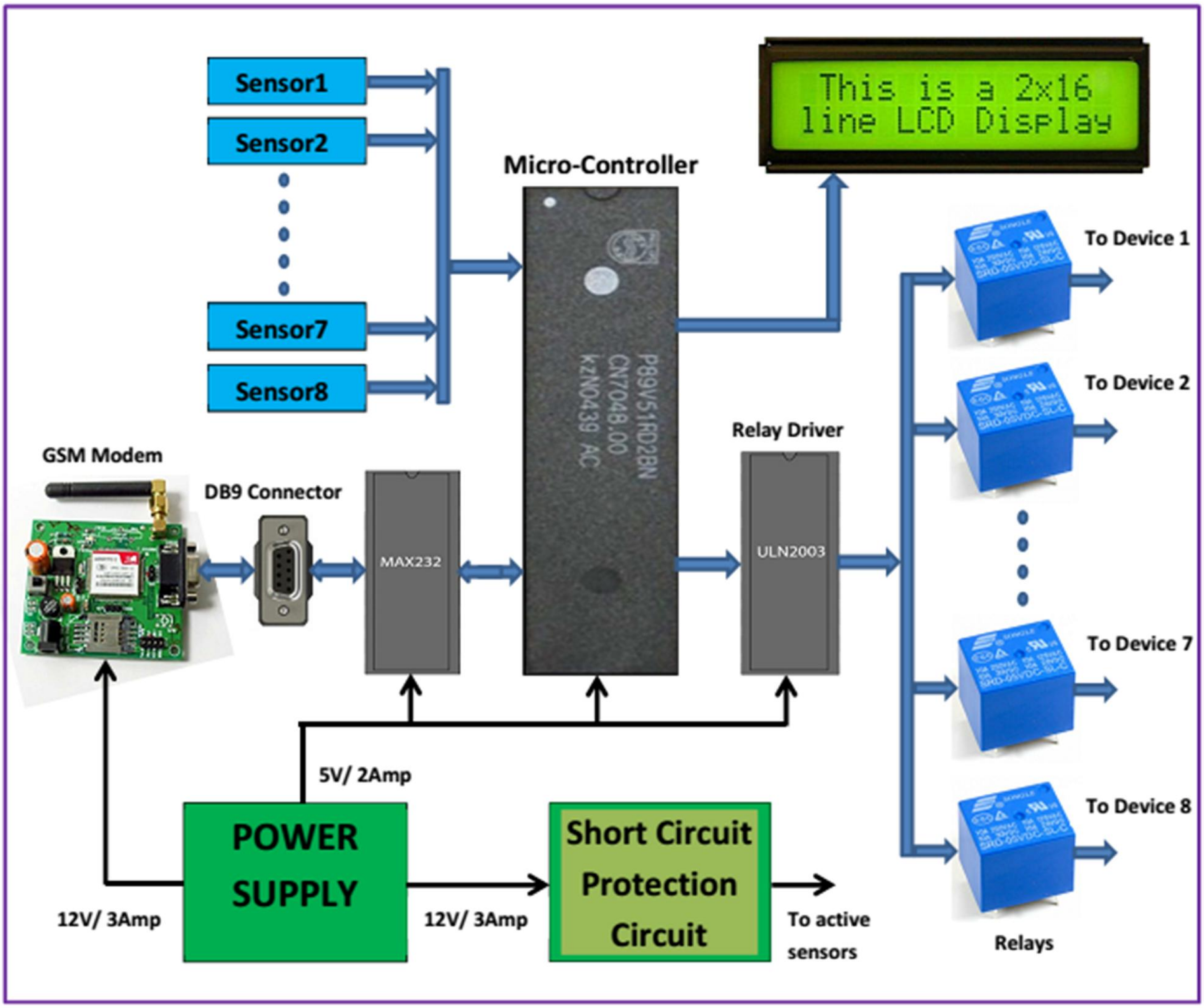


Fig. 3.1: Block Diagram.

3.2 Block Diagram Description:

Figure 3.1 shows the block diagram of Security System for HLP (High Lift Platform).

This block diagram consists of following blocks:

1) Microcontroller (P89V51RD2BN)
2) GSM Modem (SIM900A)
3) LCD
4) Power Supply
5) Short Circuit Protection Circuit
6) MAX232
7) Sensors
8) Relay Driver

3.3 Microcontroller (P89V51RD2BN):

The P89V51RD2 is an 80C51 microcontroller with 64kB Flash and 1024 bytes of data RAM.

A key feature of the P89V51RD2 is its X2 mode option. The design engineer can choose to run the application with the conventional 80C51 clock rate (12 clocks per machine cycle) or select the X2 mode (6 clocks per machine cycle) to achieve twice the throughput at the same clock frequency. Another way to benefit from this feature is to keep the same performance by reducing the clock frequency by half, thus dramatically reducing the EMI.

The Flash program memory supports both parallel programming and in serial In-System Programming (ISP). Parallel programming mode offers gang-programming at high speed, reducing programming costs and time to market. ISP allows a device to be reprogrammed in the end product under software control. The capability to field/update the application firmware makes a wide range of applications possible.

The P89V51RD2 is also In-Application Programmable (IAP), allowing the Flash program memory to be reconfigured even while the application is running.

3.3.1 P89V51RD2BN Block Diagram:

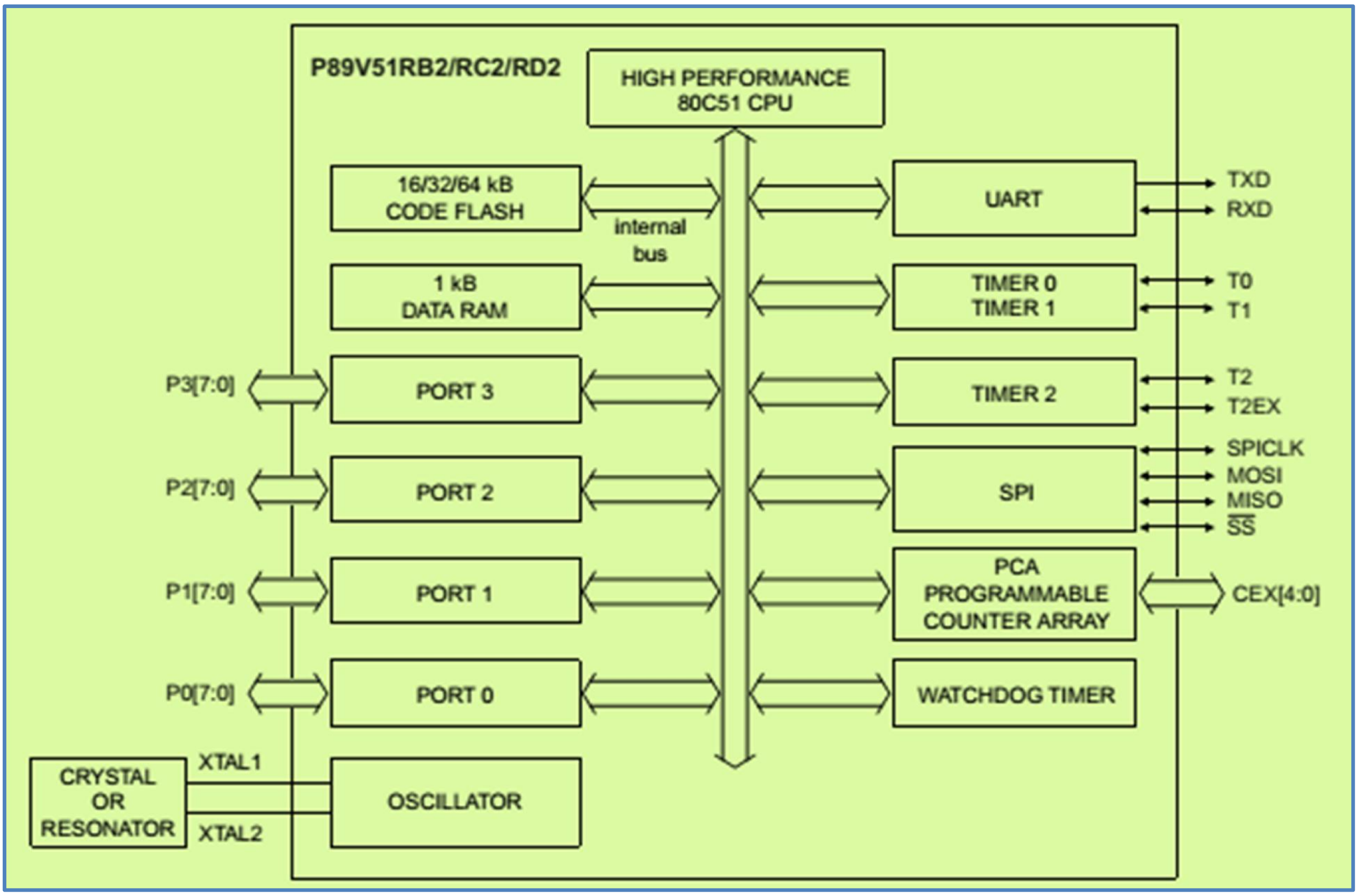


Fig. 3.2: P89V51RD2BN Block Diagram

3.3.2 Features:

- 80C51 Central Processing Unit
- 5 V Operating voltage from 0 to 40 MHz
- 64kB of on-chip Flash program memory with ISP (In-System Programming) and
- IAP (In-Application Programming)
- Supports 12-clock (default) or 6-clock mode selection via software or ISP
- PCA (Programmable Counter Array) with PWM and Capture/Compare functions
- Four 8-bit I/O ports with three high-current Port 1 pins (16 mA each)
- SPI (Serial Peripheral Interface) and enhanced UART
- Eight interrupt sources with four priority levels
- Programmable Watchdog timer (WDT)
- TTL- and CMOS-compatible logic levels
- Three 16-bit timers/counters
- Low EMI mode (ALE inhibit)

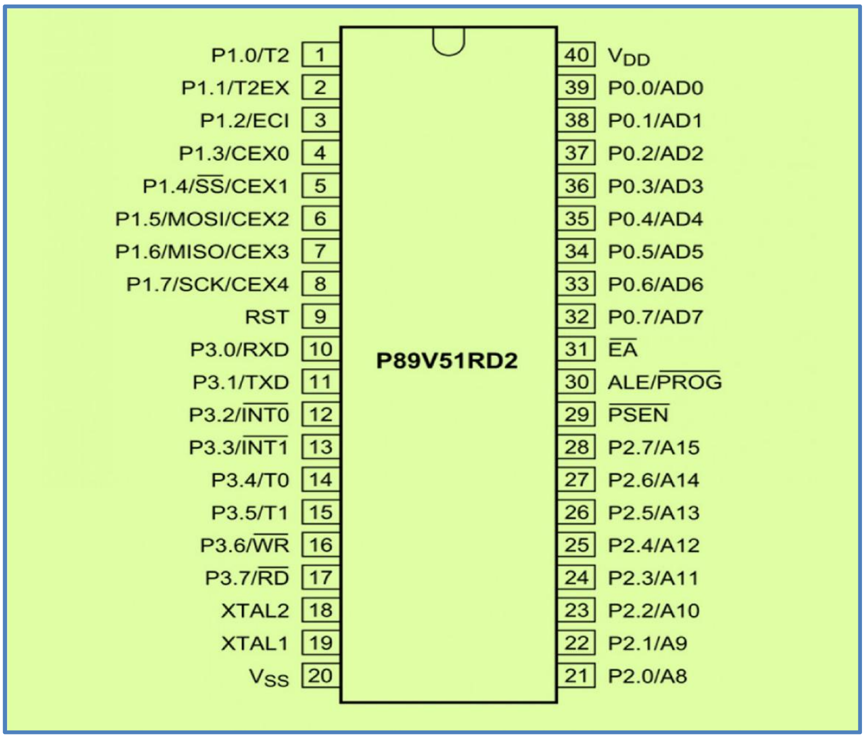


Fig: 3.3: P89V51RD2BN PIN Diagram

3.4 GSM Modem:

A **GSM modem** is a specialized type of modem which accepts a SIM card, and operates over a subscription to a mobile operator, just like a mobile phone. From the mobile operator perspective, a GSM modem looks just like a mobile phone. The GSM/GPRS Modem comes with a serial interface through which the modem can be controlled using AT command interface.

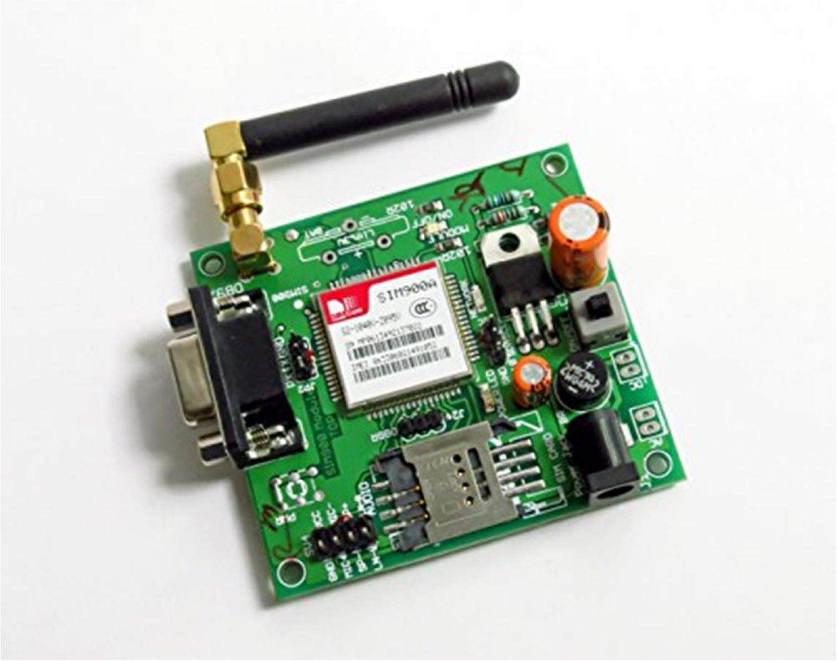


Fig. 3.4: GSM Modem

When a GSM modem is connected to a computer, this allows the computer to use the GSM modem to communicate over the mobile network. While these GSM modems are most frequently used to provide mobile internet connectivity, many of them can also be used for sending and receiving SMS and MMS messages. GSM modems can be a quick and efficient way to get started with SMS, because a special subscription to an SMS service provider is not required. In most parts of the world, GSM modems are a cost effective solution for receiving SMS messages, because the sender is paying for the message delivery.

3.4.1 GSM carrier frequencies:

GSM networks operate in a number of different carrier frequency ranges (separated into GSM frequency ranges for 2G and UMTS frequency bands for 3G), with most 2G GSM networks operating in the 900 MHz or 1800 MHz bands. Where these bands were already allocated, the 850 MHz and 1900 MHz bands were used instead (for example in Canada and the United States).

3.4.2 SIM900 GSM modem Features:

In this security system we will use SIM900 GSM modem. It is having following features:

- Power supply: Single supply voltage from 9V to 12V (DC).
- Onboard power ON and Network indication LED
- 9600 standard baud rate
- RS232 output to connect directly to computer
- Serial TTL on board pins to directly connect to microcontroller.
- Onboard Audio (speaker and microphone) interface pins.
- Dual-Band 900/ 1800 MHz
- Control via AT commands
- Operating temperature: -40° C to +85°C

3.4.3 AT commands used in this system:

Sr. No.	Command	Description
1.	AT	First command to check the status of GSM modem.
2.	ATE0	Echo OFF.
3.	AT+CMGF	To set GSM for SMS in Text mode.
4.	AT+COPS?	To check the name of operator.
5.	ATD	To dial the number.
6.	AT+CNMI	New Message Indications to Terminal Equipment.
7.	AT+CPBR	To read the phonebook.
8.	AT+CPBW	To write new number in the phonebook.
9.	AT+CMGD	To delete the message.
10.	AT+CMGS	To send the message.

Table 3.1: AT commands used in this system.

3.5 POWER SUPPLY:

DC regulated +5 Volt Power is required to power up circuit of the Security System and DC regulated +12 Volt Power is required for the GSM modem, alarm, siren and some active sensors. Connecter J1 is the DC power connecter. We have to use 24v DC to 12v DC (3 Amp) step down power adapter. Output connecter of this adapter should be connected to DC power connecter. Following figure shows the circuit diagram of +5volt regulated power supply used in the Security System circuit.

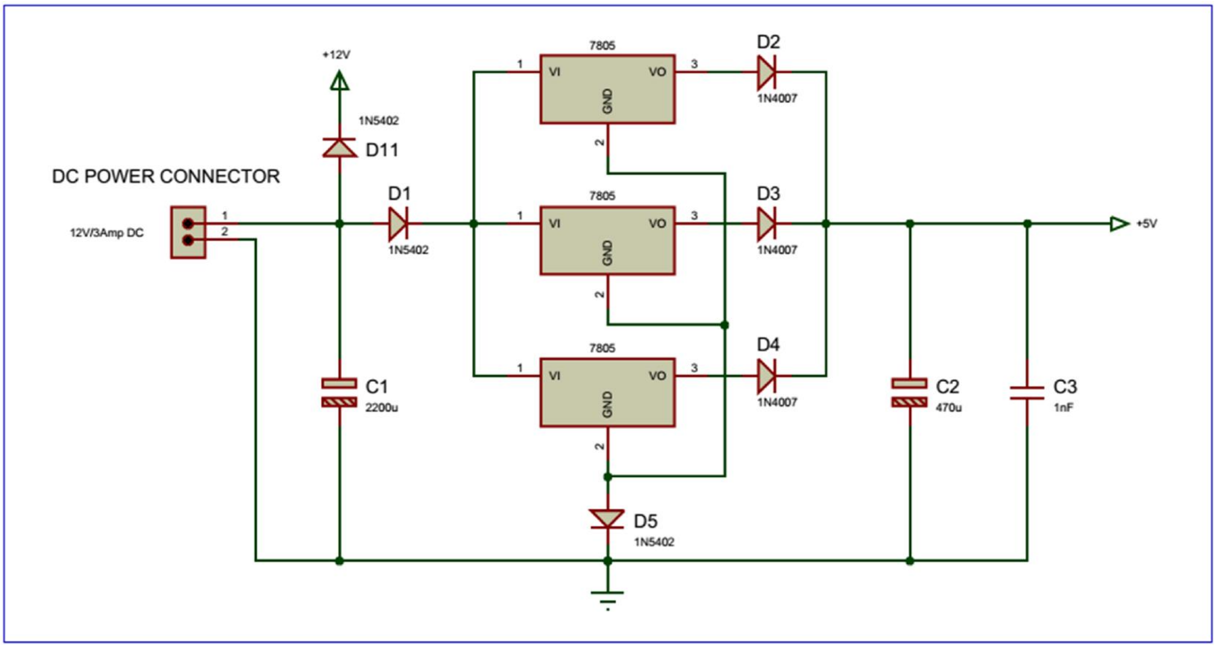


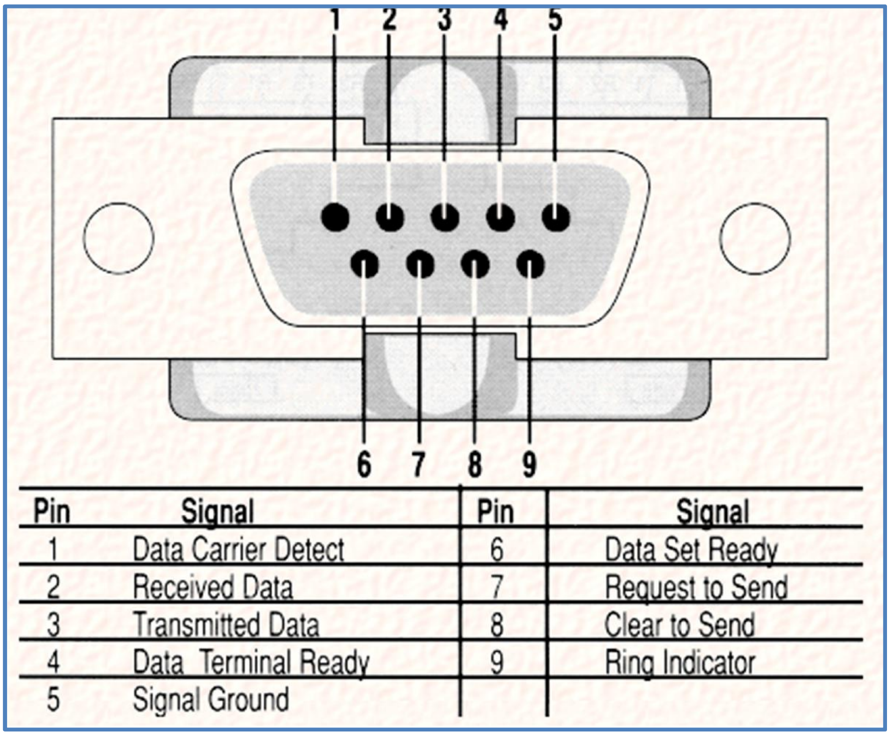
Fig. 3.5: +5 V Regulated Power Supply

The +5 volt power supply is based on the commercial 7805 voltage regulator. It contains all the circuitry needed to accept any input voltage from 7 to 18 volts and produce a steady +5 volt output, accurate to within 5% (0.25 volt). It also contains current-limiting circuitry and thermal overload protection, so that it won't be damaged in case of excessive load current; it will reduce its output voltage instead. Here we have connected three voltage regulators in parallel to increase the current rating of the power supply up to 2 Amp.

3.6 Serial Port Interface:

The RS232 serial port is another option for transmitting data between the PC and the microcontroller. One of the major functions of the serial port is to put data into a serial format so that it can be transmitted via modem. One of the functions that we use it for is to download programs that have been compiled on a PC to run on a microcontroller. The one good feature is that the RS232 would only need three wires between the PC and the microcontroller. One line is data transmit; one line is data receive, and the last line is a common ground between the two devices.

For sending commands to GSM modem and receiving data from it we are using serial UART (Universal Asynchronous Receive Transmit) communication. We can also interface Microcontroller to the serial port of PC for programming it using UART serial protocol. Following figure shows the pin configuration of the serial port available on the PC.



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Fig. 3.6: DB9 Connecter Pin out.

Following diagram shows the circuit diagram for connecting Microcontroller to the PC or any serial device. We cannot directly connect microcontroller to the PC because of logical voltage level difference in between them. This is because the Microcontroller follows TTL logic whereas PC follows CMOS logic.

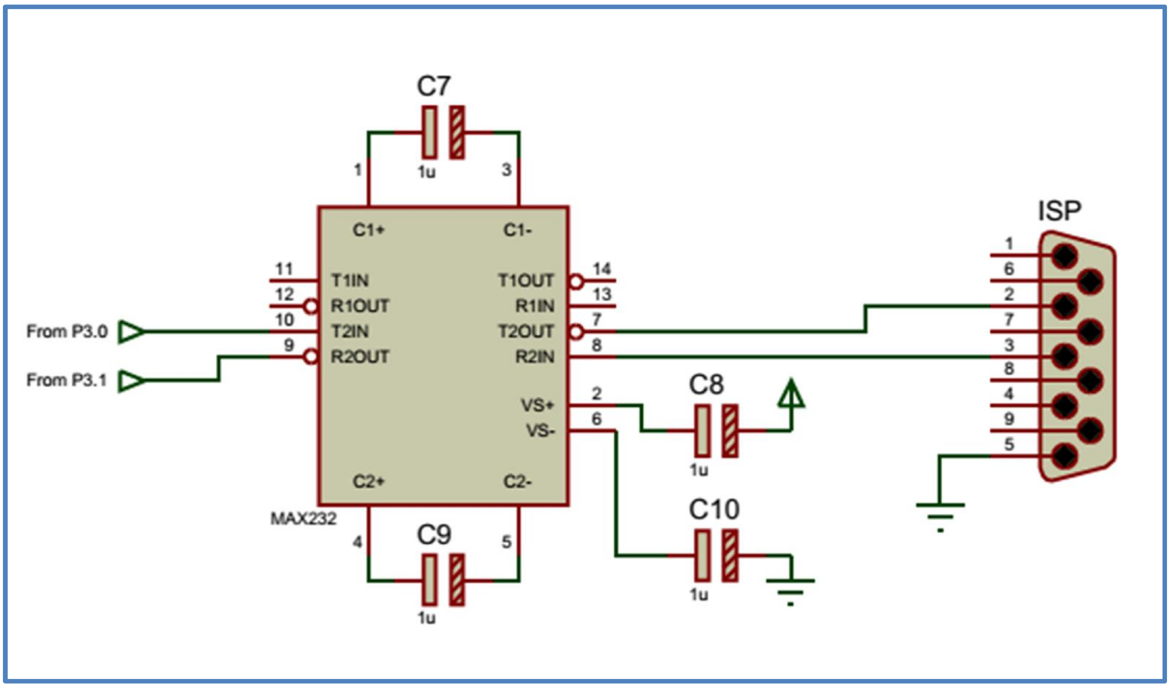


Fig. 3.7: MAX232 circuit diagram.

To connect Microcontroller to the PC we have to use the level converter. It should be capable of converting TTL voltage levels to CMOS voltage levels and CMOS voltage levels to TTL. To do this we can use MAX232 IC. Circuit diagram for MAX232 is as shown above. The MAX232 is a hardware layer protocol converter IC manufactured by the Maxim Corporation. Commonly known as a RS-232 Transceiver, it consists of a pair of drivers and a pair of receivers. At a very basic level, the driver converts TTL and CMOS voltage levels to levels, which are compatible for serial port communications. The receiver performs the reverse conversion.

If you have a microcontroller circuit that requires communication through a serial port, then this is the chip to use. This is a versatile IC, which is one of those wonderful components that solve so many signal conversion problems.

3.7 LCD Display:

LCD stands for Liquid Crystal Display and it is used to display the data. LCD we are using is 16x2 i.e. 16 characters in 1 line, total 2 lines are there. We can use a better resolution LCD but 16x2 LCD is sufficient to fulfill requirements of this project. **This LCD has 8-bit parallel interface. It is possible to use all 8 bits plus 3 control signals or 4 bits plus 3 control signals.** It requires +5V to operate.

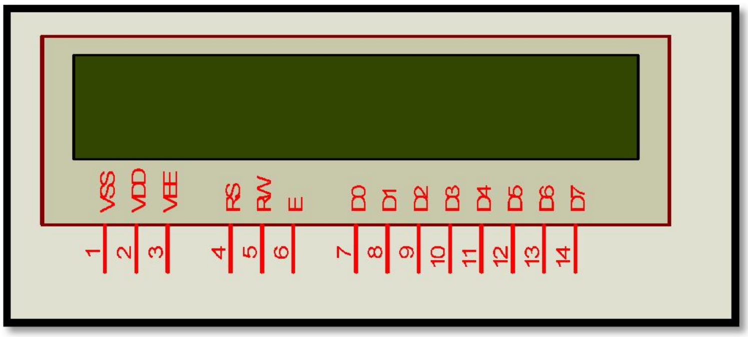


Fig. 3.8: 16*2 LCD Display pin outs.

It is connected to **port 0** of microcontroller. It acts as an output to microcontroller. It uses ASCII values to display the characters. Control pins RS and En are connected to P0.2 and P0.3 respectively. R/W pin is connected to ground and data pins D4 to D7 are connected to P0.4 to P0.7. Eight bit data is transferred in ASCII format after splitting into two 4 bit sets and then.

LCD’s can add a lot to your application in terms of providing a useful interface for the user, debugging an application or just giving it a "professional" look.

Fig. 3.9: 16*2 LCD Display snapshot.

3.7.1 LCD PIN DESCRIPTIONS: -

- **V_{CC}, V_{SS}, V_{EE}: -**

  V_{CC} & V_{SS} provides +5V & ground respectively. V_{EE} is used for controlling LCD contrast.

- **RS (Register Select): -**

  There are two very important registers inside the LCD – Command register and Data register. The RS pin is used for their selection as follows:

  If RS = 0, the instruction command code register is selected, allowing the user to send command such as clear display, cursor at home, etc.

  If RS = 1, the data register is selected, allowing the user to send data to be displayed on LCD.

- **R/W (Read/Write): -**

  R/W input allows the user to write information to the LCD or read information from it.

  R/D = 1, Read operation
  R/D = 0, Write operation

- LCD to latch the information presented to its data pins uses the Enable pin. When data is supplied to data pins, a high – to - low pulse must be applied to this pin in order to latch in the data present at the data pins of LCD. This pulse must be a minimum of 450ns wide

- The 8 bit data pins, D_0-D_7, are used to send information to the LCD or read the contents of the LCD's internal registers. To display letters and numbers, we send ASCII codes for the letters A-Z, a-z and numbers 0-9 to these pins while making RS=1. There are also instruction command codes that can be sent to the LCD to clear the display or force the cursor to the home position or blink the cursor.

- We also use RS = 0 to check the busy flag bit to see if the LCD is ready to receive the information. The busy flag is D7 and can be read when R/W = 1 and RS = 0, as follows: if R/W = 1, RS = 0. When D7 = 1 (busy flag=1), the LCD is busy in taking care of the internal operation and will not accept any new information. When D7 = 0, the LCD is ready to receive new information.

3.7.2 WORKING AND LOGIC FOR LCD:

LCD’s can add a lot to your application in terms of providing a useful interface for the user, debugging an application or just giving it a "professional" look. LCD’s can be added quite easily to an application and use as few as three digital output pins for control.

The pins of LCD are wired as given in table:

Pins	Description
1	Ground, (V_{SS})
2	+5 V power supply, ($V_{CC)}$
3	Power supply to control contrast voltage, (V_{EE})
4	"R/S" _Instruction/Register Select
5	"R/W" _Read/Write LCD Registers
6	"E” Enable Clock
7 - 14	The 8 bit Data Bus (I/O Pins)
15-16	LCD back light.

Table 3.2: LCD pin description.

The interface is a parallel bus, allowing simple and fast reading/writing of data to and from the LCD.

This waveform will write an ASCII Byte out to the LCD's screen. The ASCII code to be displayed is eight bits long and is sent to the LCD either four or eight bits at a time. If four-bit mode is used, two "nibbles" of data (Sent high four bits and then low four bits with an "E" Clock pulse with each nibble) are sent to make up a full eight-bit transfer. The "E" Clock is used to initiate the data transfer within the LCD.

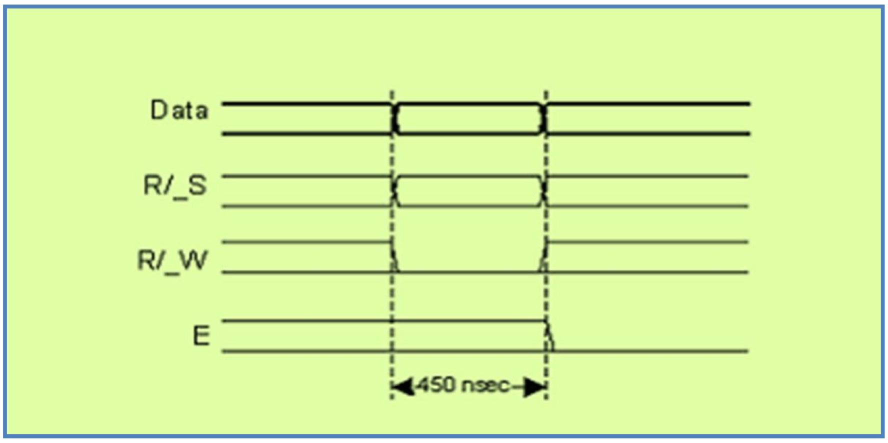


Fig. 3.10: LCD Data write waveform

Sending parallel data, as either four or eight bits are the two primary modes of operation. While there are secondary considerations and modes, deciding how to send the data to the LCD is most critical decision to be made for an LCD interface application.

Eight-bit mode is best used when speed is required in an application and at least ten I/O pins are available. Four-bit mode requires a minimum of six bits. To wire a Microcontroller to an LCD in four-bit mode, just the top four bits (DB4-7) are written to.

The "R/S" bit is used to select whether data or an instruction is being transferred between the Microcontroller and the LCD. If the Bit is set, then the byte at the current LCD "Cursor" Position can be read or written. When the Bit is reset, either an instruction is being sent to the LCD or the execution status of the last instruction is read back (whether or not it has completed).

3.7.3 LCD Instruction Set:

Following table shows instructions for LCD interface, their decimal and hexadecimal codes. We can send these commands to the LCD on data lines.

INSTRUCTION	Decimal	Hexadecimal
Function set (8-bit interface, 2 lines, 5*7 Pixels)	56	38
Function set (8-bit interface, 1 line, 5*7 Pixels)	48	30
Function set (4-bit interface, 2 lines, 5*7 Pixels)	40	28
Function set (4-bit interface, 1 line, 5*7 Pixels)	32	20
Entry mode set	See Below	See Below
Scroll display one character right (all lines)	28	1E
Scroll display one character left (all lines)	24	18
Home (move cursor to top/left character position)	2	2
Move cursor one character left	16	10
Move cursor one character right	20	14
Turn on visible underline cursor	14	0E
Turn on visible blinking-block cursor	15	0F
Make cursor invisible	12	0C
Blank the display (without clearing)	8	08
Restore the display (with cursor hidden)	12	0C
Clear Screen	1	01
Set cursor position (DDRAM address)	128 + addr	80+ addr
Set pointer in character-generator RAM (CG RAM address)	64 + addr	40+ addr
Go to second line	192	C0

Table 3.3: LCD instruction set.

3.8 Short Circuit protection Circuit:

In this system we are using some active sensors like Smoke/Fire detector, PIR (Passive Infrared) sensor. We have to give +12 volt power supply for these sensors. There is very rare possibility of the short circuit in these sensors. To avoid any damage to the system and the HLP's battery we have to take care of that. So we have to implement a short circuit protection circuit in this system. Following diagram shows how we implemented it.

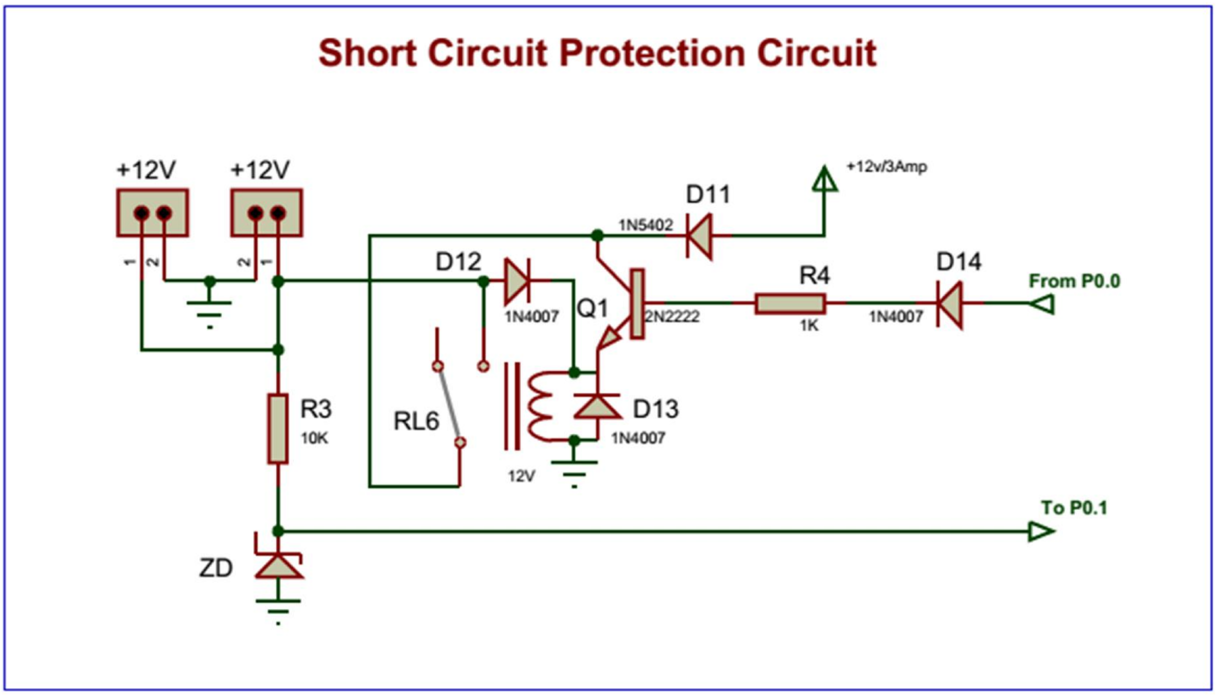


Fig. 3.11: Short circuit protection circuit.

Initially transistor Q1 will be ON because of logic high signal on P0.0 of the Microcontroller. When relay coil will be energized it will switch common point of it from NC (Normally Closed) to NO (Normally Open) and 12 volt supply will be passed active sensors and relay coil. After some time Q1 will turn OFF. But relay remains ON continuously. Across the zener diode we will get 5 volt. If there is short circuit in the any active sensor then relay will turn OFF automatically and across the zener diode we will not get 5 volt. We have to keep watch on this. When we will not get 5 volt it will indicate that there is short circuit in the 12 volt power supply. In this way we can avoid any damage due to short circuit without the need of changing any component from the circuit. Next time if we restart the system after disconnecting the faulty sensor from system we can get 12 volt supply after short circuit protection circuit.

3.9 Sensors:

In this security system we have to use different types of active and passive sensors like PIR sensor, Smoke/Fire detector sensor, Door closer sensor etc. All active sensors will be powered through short circuit protection circuit to avoid any damage to the control circuit in case of short circuit in active sensors.

3.9.1 PIR Sensor:

A **passive infrared sensor** (**PIR sensor**) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors. PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out. PIRs are basically made of a Pyro electric Sensor which can detect levels of infrared radiation.

Fig. 3.12: PIR Sensor.

3.9.1.1 PIR Operating Principles:

All objects with a temperature above absolute zero emit heat energy in the form of radiation. Usually this radiation is invisible to the human eye because it radiates at infrared wavelengths, but it can be detected by electronic devices designed for such a purpose.

The term *passive* in this instance refers to the fact that PIR devices do not generate or radiate any energy for detection purposes. They work entirely by detecting the energy given off by other objects. It is important to note that PIR sensors don't detect or measure "heat" per sec; instead they detect the Infrared radiation emitted from an object which is different from but often associated/correlated with the object's temperature (e.g., a detector of X-rays or gamma rays would not be considered a heat detector, though high temperatures may cause the emission of X or gamma radiation).

3.9.1.2 PIR Operation:

An individual PIR sensor detects changes in the amount of infrared radiation impinging upon it, which varies depending on the temperature and surface characteristics of the objects in front of the sensor. When an object, such as a human, passes in front of the background, such as a wall, the temperature at that point in the sensor's field of view will rise from room temperature to body temperature, and then back again. The sensor converts the resulting change in the incoming infrared radiation into a change in the output voltage, and this triggers the detection. The PIR sensor itself has two slots in it. Each slot is made of a special material that is sensitive to IR. The lens used here is not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.

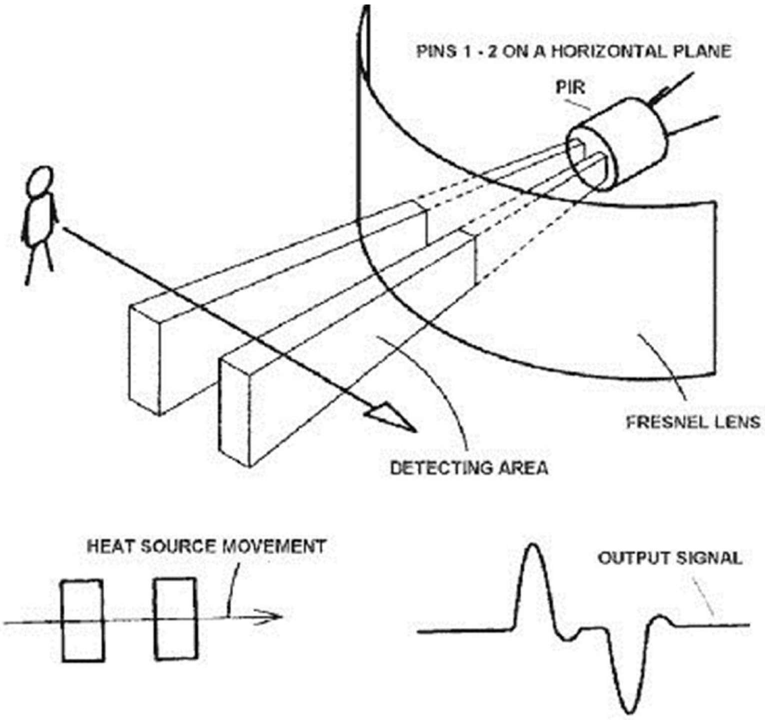


Fig. 3.13: PIR Sensor Operation.

3.9.2 Smoke/Fire Detector:

A **smoke detector** is a device that senses smoke, typically as an indicator of fire. There are two main types of smoke detectors: ionization detectors and photoelectric detectors. A smoke alarm uses one or both methods, sometimes plus a heat detector, to warn of a fire.

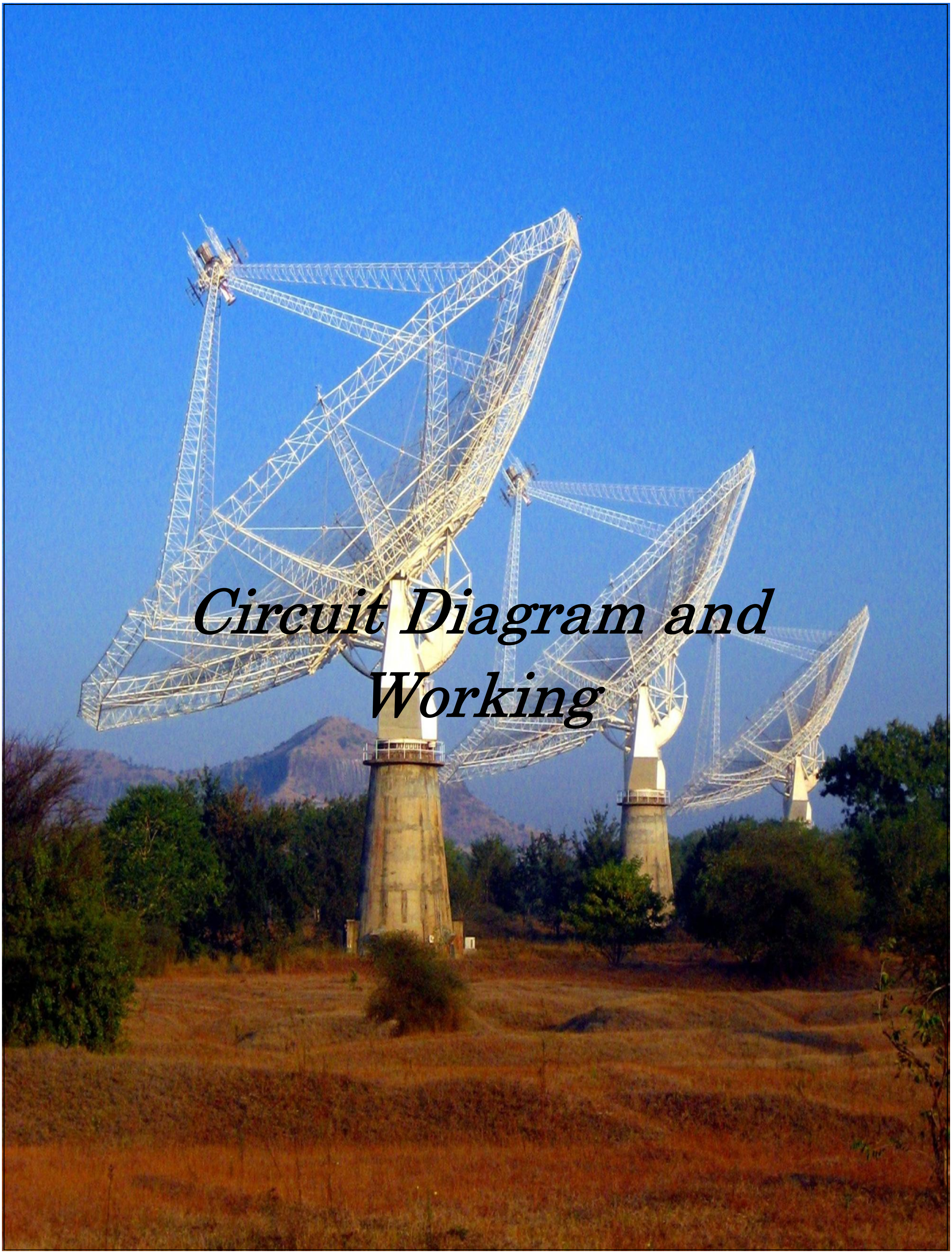
Fig. 3.14: Smoke/Fire Detector.

3.9.2.1 Ionization Detectors:

Ionization detectors have an ionization chamber and a source of ionizing radiation. The source of ionizing radiation is a minute quantity of americium-241 (perhaps 1/5000th of a gram), which is a source of alpha particles (helium nuclei). The ionization chamber consists of two plates separated by about a centimeter. The battery applies a voltage to the plates, charging one plate positive and the other plate negative. Alpha particles constantly released by the americium knock electrons off of the atoms in the air, ionizing the oxygen and nitrogen atoms in the chamber. The positively-charged oxygen and nitrogen atoms are attracted to the negative plate and the electrons are attracted to the positive plate, generating a small, continuous electric current. When smoke enters the ionization chamber, the smoke particles attach to the ions and neutralize them, so they do not reach the plate. The drop in current between the plates triggers the alarm.

3.9.2.2 Photoelectric Detectors:

In one type of photoelectric device, smoke can block a light beam. In this case, the reduction in light reaching a photocell sets off the alarm. In the most common type of photoelectric unit, however, light is scattered by smoke particles onto a photocell, initiating an alarm. In this type of detector there is a T-shaped chamber with a light-emitting diode (LED) that shoots a beam of light across the horizontal bar of the T. A photocell, positioned at the bottom of the vertical base of the T, generates a current when it is exposed to light. Under smoke-free conditions, the light beam crosses the top of the T in an uninterrupted straight line, not striking the photocell positioned at a right angle below the beam. When smoke is present, the light is scattered by smoke particles, and some of the light is directed down the vertical part of the T to strike the photocell. When sufficient light hits the cell, the current triggers the alarm



Circuit Diagram and Working

4.1 Circuit Diagram:

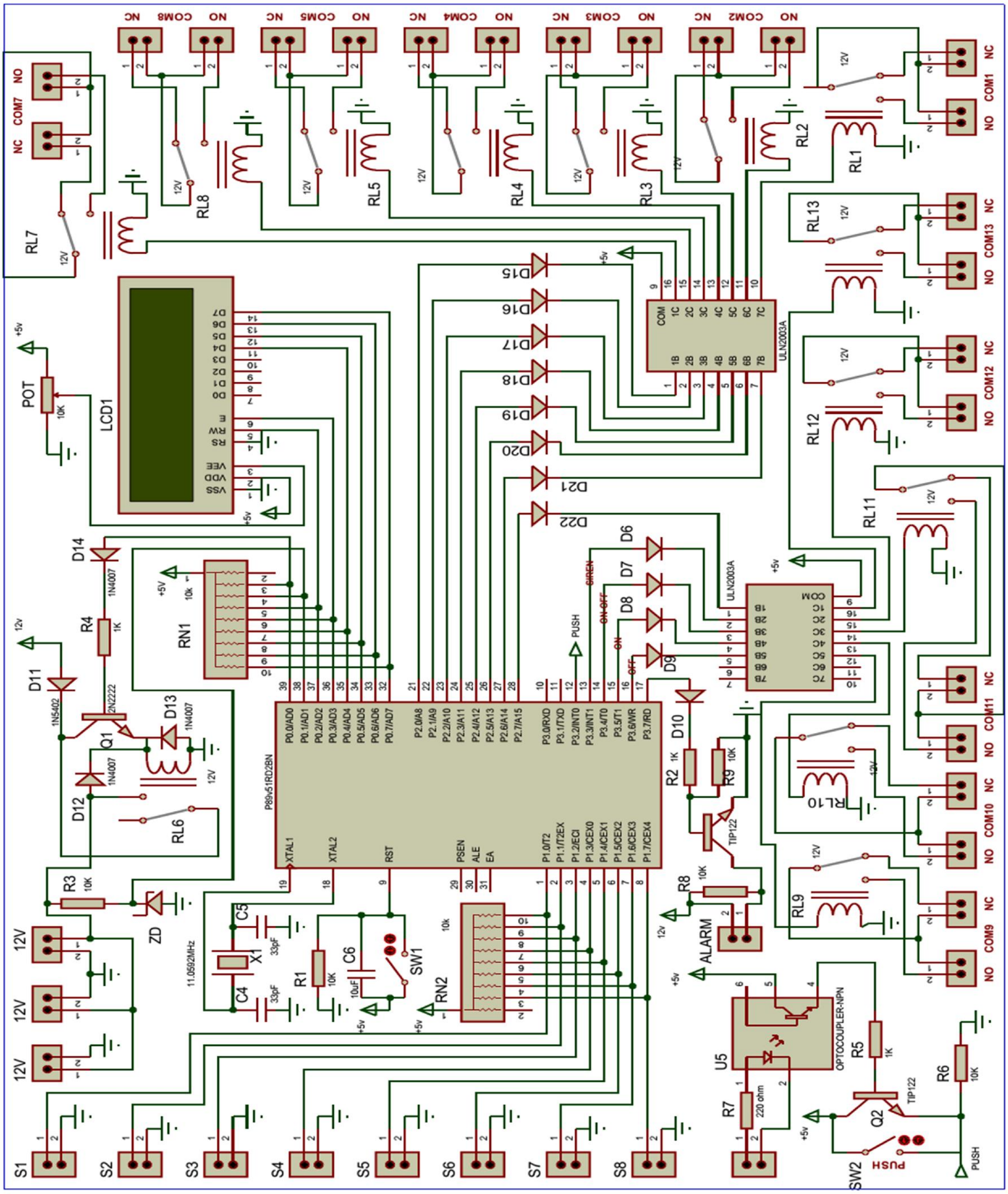


Fig. 4.1: Circuit Diagram.

4.2 Working:

In this system we are using P89V51RD2BN Microcontroller which is in system programmable having 1kB RAM and 64kB ROM. It is sufficient for our application. This Microcontroller is easily available in the electronic market. We have to use the battery (24 volt / 100 Amph) present in the HLP. But to power up the security system we requires only 12 Volt / 3 Ampere (max). So that, we have to use DC to DC buck (step down) converter, which can give 12 Volt / 3 Ampere output.

In this system we have to use door closer sensors, PIR sensor and smoke sensor . We can connect eight sensors to the connectors marked as S1 to S8. We can connect more than eight sensors by connecting some of them in series. For example if we want to connect five door closer sensors and there is only one connector available on the circuit then we have to connect all these sensors in series to that connector. On other hand if we want to connect less than eight sensors then we have to short the remaining connectors from S1 to S8. We have to configure all these sensors to give output in the form of NO/NC contact. For powering up the smoke sensor and PIR sensor we have to use 12 volt and for control circuit we have to use 5 volt from power supply.

12 volt power is given to all active sensors through short circuit protection circuit. If in case there is short circuit in any sensor connected to the 12 volt power supply then power supply connection to the active sensors will be disconnected automatically with the help of a relay. And it will be remain as it is till the system will not restart by the authorized person. Authorized person is the one who is having key of the lock present in the system.

We have to insert one SIM card in the GSM modem for SMS alert and control and monitoring of the system from authorized person’s mobile number. We can connect eight devices to the system which can be controlled by sending commands through SMS from the authorized person’s mobile number only. Authorized person can add, delete or change the numbers linked to the security system. He can add up to five numbers to the system. By default there will be one number present in the system which is nothing but the authorized person’s number.

When authorized person will turn on the system with the help of one key “Initializing” will be displayed on the LCD screen and alarm will be turned ON. It will take more than one minute to initialize the system. Because GSM modem requires near about 30 seconds to initialize and authenticate the SIM number on GSM network. Before initialization completes the person has to leave the vehicle and close all the doors from outside. During this time some AT commands require to use GSM modem for the particular application will be send to the modem from Microcontroller. And microcontroller will read all numbers added to the system using some AT commands. These numbers will be present in the SIM card’s phonebook memory. After initializing if all sensors are working fine and are in normal state then “Welcome to Security System” will be displayed on the screen and same message will be sent to all numbers linked with the system and the alarm will be turned OFF. Authorized person will get SIM card account balance and validity information along with this message. Otherwise, "System Error" will be displayed on the screen and same will be sent to all these numbers. After few seconds sensor numbers which are having problem will be displayed and alarm will remain ON till the authorized person does not turn OFF the system with the help of key.

If everything is ok and "Welcome to Security System is displayed on the screen then, Microcontroller will continuously check for the sensors to become NO (Normally Open) from NC (Normally Close). If this happens then alarm will be turned ON and "Detected Sensor" followed by sensor number will be displayed on the screen and same will be sent to all linked numbers with the system. After two minutes alarm will be turned OFF and siren will be turned ON for ten minutes. Then system will remain in this state till the authorized person does not restart the system.

4.3 Security System commands for GSM modem:

We can use these commands to control and monitor the Security System. System will accept these commands from authorized mobile number only. These are all case sensitive commands.

Sr. No.	Commands	Description	Reply From Security System
1.	Status	This command will return status of the security system.	Security System is working fine. (On authorized number only)
2.	Num2 <10 Digit Number> Num3 <10 Digit Number> Num4 <10 Digit Number> Num5 <10 Digit Number>	To add new number to location 2, 3, 4 or 5	i) <10 Digit Number> is added to security system. (On authorized number) ii) Your number is added to security system. (On newly added number)
3.	Num2 x Num3 x Num4 x Num5 x	To delete number from location 2, 3, 4 or 5	<10 Digit number> is deleted from location 1, 2, 3, 4, or 5. (On authorized number only)
4.	Get num	To get all available numbers from Security System.	1. <10 Digit Number> ....Authorized Number 2. <10 Digit Number> ....2nd Number 3. <10 Digit Number> ....3rd Number 4. <10 Digit Number> ....4th Number 5. <10 Digit Number> ....5th Number (On authorized number only)

Table 4.1: Security system commands.

PCB Layout

5.1 PCB Layout:

5.1.1 Top Layer:

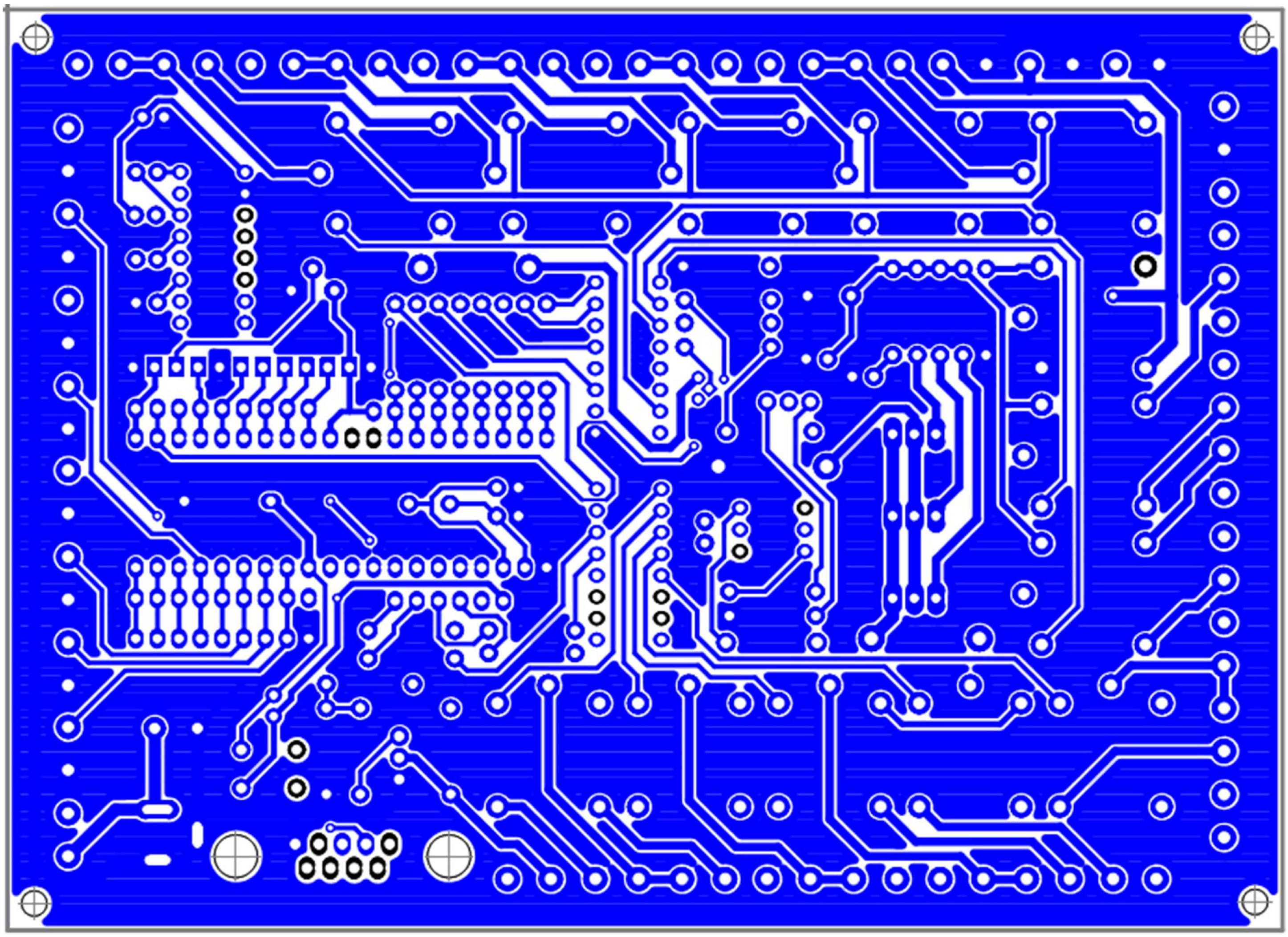


Fig. 5.1: PCB Layout Top Side.

5.1.2 Bottom Layer:

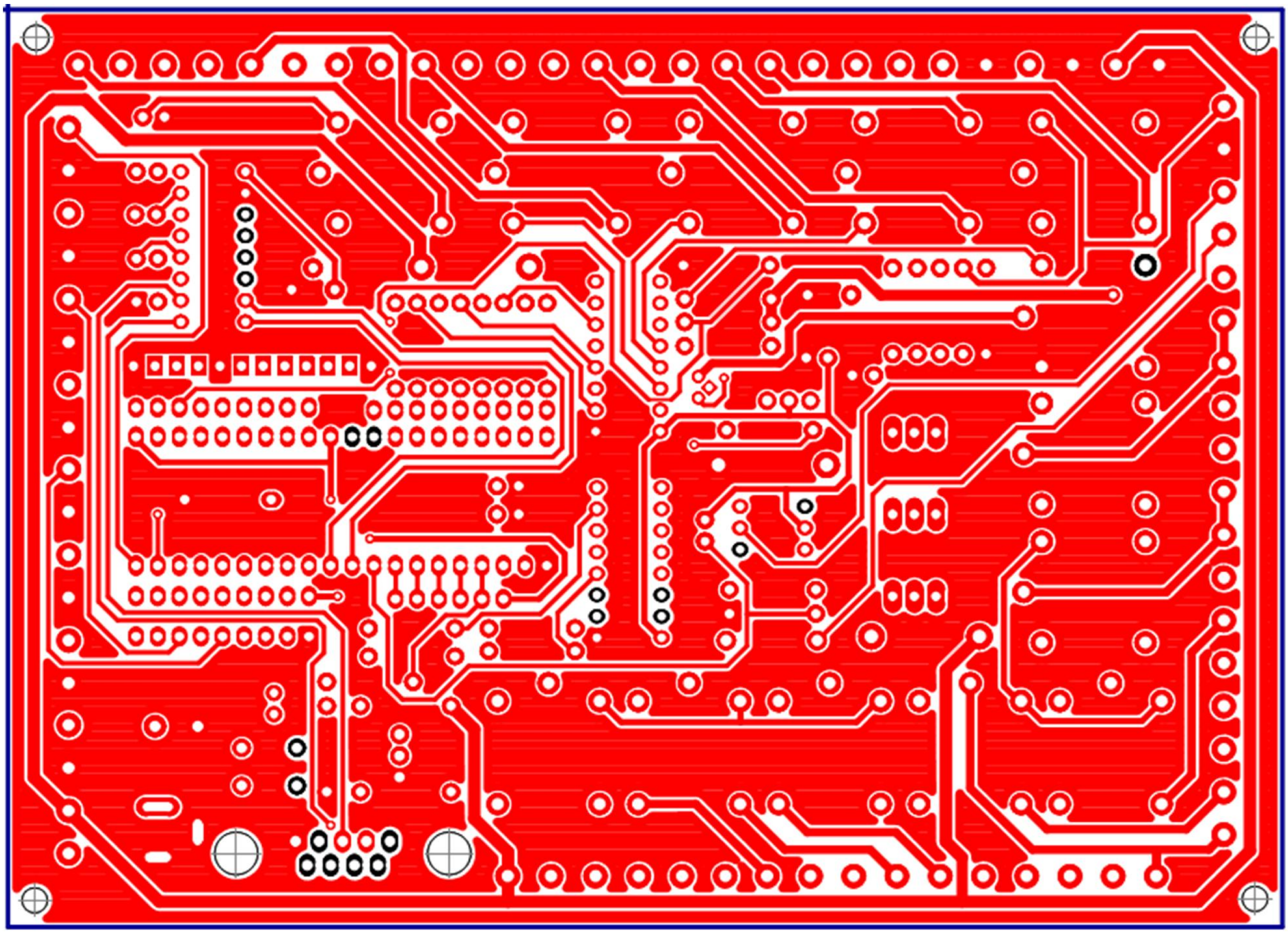


Fig. 5.2: PCB Layout Bottom Side.

5.1.3 Top Silk Layer:

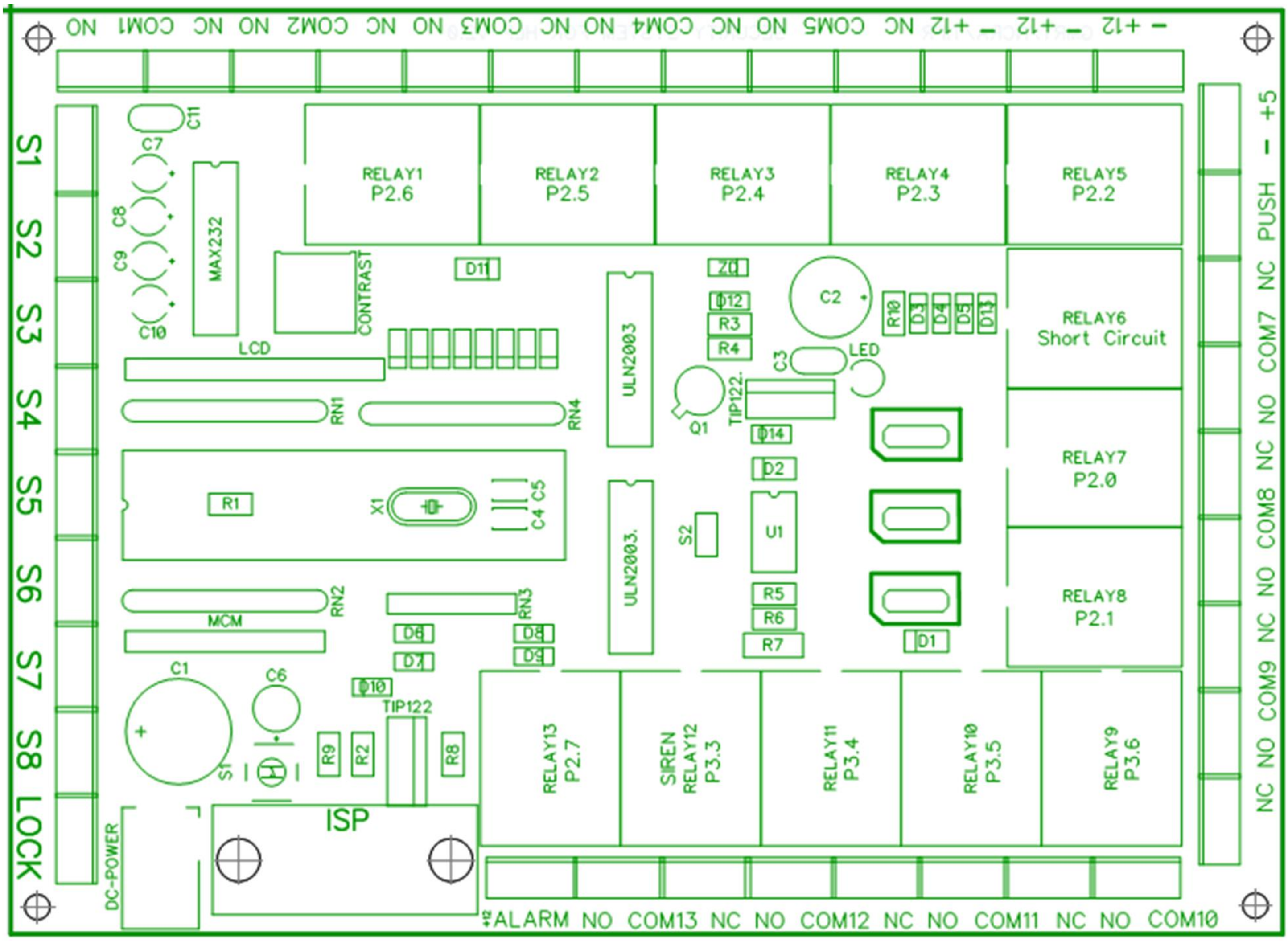


Fig. 5.3: PCB Top Silk.

SOFTWARE USED

6.1 Programing Software (KEIL):

Copyright © 1997 - 2005 Keil Software, 2005 - 2009 ARM Ltd. All rights reserved.
This product is protected by US and international laws.

Fig. 6.1: Keil snapshot-1.

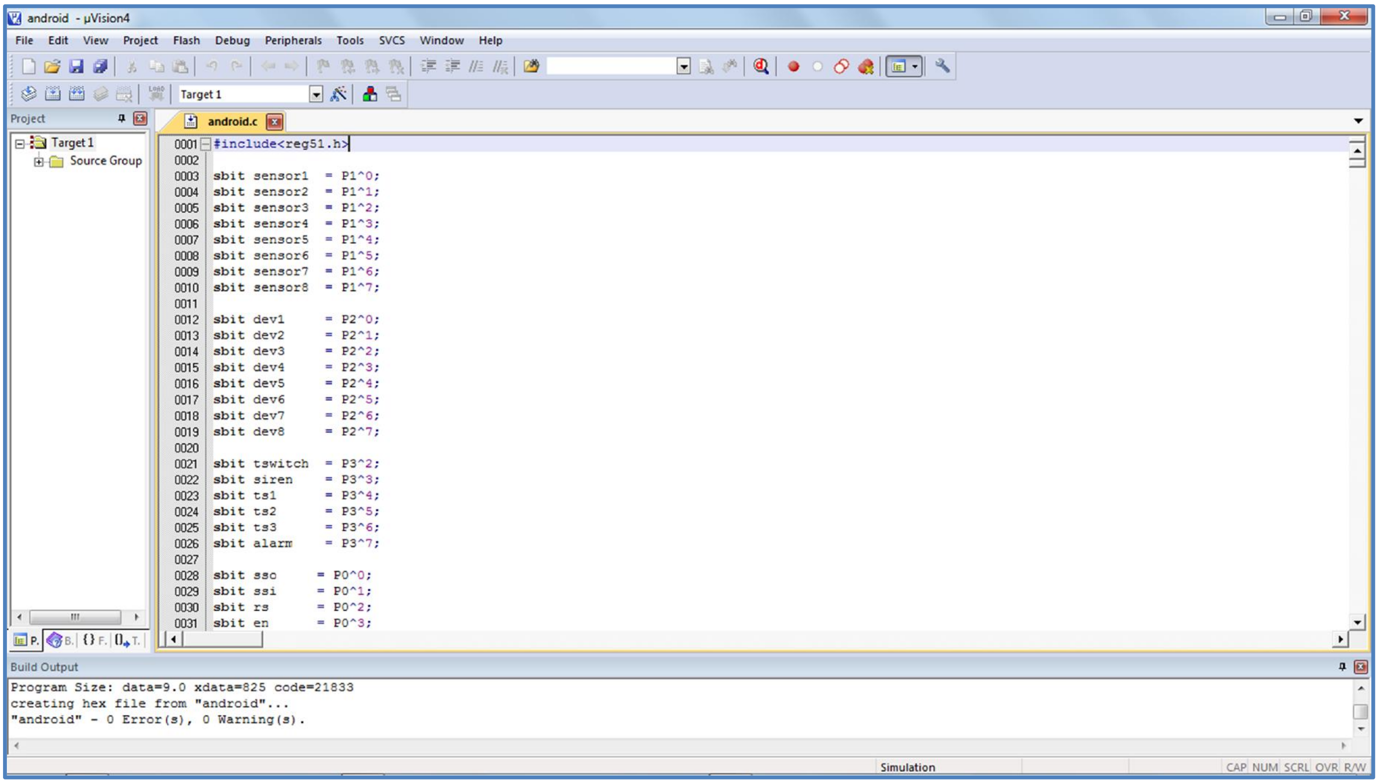
6.1.1 Overview:

The µVision4 IDE is a windows based software development platform that combines a robust and modern editor, project manager, and make facility. µVision4 integrates all the tools you need to develop embedded applications including C/C++ compiler, macro assembler, linker/locator, and a HEX file generator. µVision4 helps expedite the development process of your embedded application by providing the following:

- Full-featured **source code editor**.
- **Device Database** for configuring the development tool.
- **Project Manager** for creating and maintaining your projects.
- Integrated **Make Utility** functionality for assembling, compiling, and linking your embedded applications.
- **Dialogs** for all development environment settings.
- True integrated source-level and assembler-level **Debugger** with high-speed CPU and peripheral **Simulator**.
- Advanced GDI interface for software debugging in the target hardware and for connecting to the Keil ULINK Adapter family.
- **Flash programming utility** for downloading the application program into Flash ROM.

The µVision4 IDE offers numerous features and advantages that help you to develop embedded applications quickly and successfully. The Keil tools are easy to use, and are guaranteed to help you achieve your design goals in a timely manner. The µVision4 IDE & Debugger is the central part of the Keil development tool chain. µVision4 offers a Build Mode and a Debug Mode. In Build Mode you maintain the project, the project files, write your code, select the target hardware and device, and generate the application. In Debug Mode you verify and debug your program with the integrated, powerful Simulator or directly on target hardware with the Keil ULINK USB-JTAG Adapter (or other AGDI drivers), or analyze the application behavior.

The Keil Compilers support all 8051, 251, C16x/ST10, and ARM compatible devices. However, there are various meanings to "support", all of which are explained below. The Keil Compiler generates code for any device that is compatible with the 8051, 251, C16x/ST10, or ARM microcontrollers. The only exception to this would be a device that has removed or altered the instruction set. However, that device would no longer be a compatible part.



```
#include<reg51.h>

sbit sensor1  = P1^0;
sbit sensor2  = P1^1;
sbit sensor3  = P1^2;
sbit sensor4  = P1^3;
sbit sensor5  = P1^4;
sbit sensor6  = P1^5;
sbit sensor7  = P1^6;
sbit sensor8  = P1^7;

sbit dev1     = P2^0;
sbit dev2     = P2^1;
sbit dev3     = P2^2;
sbit dev4     = P2^3;
sbit dev5     = P2^4;
sbit dev6     = P2^5;
sbit dev7     = P2^6;
sbit dev8     = P2^7;

sbit tswitch  = P3^2;
sbit siren    = P3^3;
sbit ts1      = P3^4;
sbit ts2      = P3^5;
sbit ts3      = P3^6;
sbit alarm    = P3^7;

sbit sso     = P0^0;
sbit ssi     = P0^1;
sbit rs      = P0^2;
sbit en      = P0^3;
```

```
Program Size: data=9.0 xdata=825 code=21833
creating hex file from "android"...
"android" - 0 Error(s), 0 Warning(s).
```

Fig: 6.2: Keil Snapshot-2.

When you start a project using the Keil uVision IDE, you must select a chip from the Device Database. Keil is constantly updating the database and adding new parts. To ensure that you always have the latest database, you may download the latest updates from the Keil Website. If the latest software's Device Database does not include the part you use, you may add a Device Database entry we may refer to UVISION: ADDING CUSTOM PARTS TO THE DEVICE DATABASE for more information. The Device Database is simply a way to specify the default compiler, assembler, and linker settings. For new devices, you may simply copy the settings for a similar device.

Each microcontroller has its own unique set of Special Function Registers. The SFRs for a chip MAY be identical to those of another device. Keil Software provides custom header files that define the SFRs for almost every 8051, 251, and C16x compatible device. However, there may be new devices for which we have not yet created a header file. That does not mean that this chip is not supported. Creating header files for new devices is easy. On occasion, new devices have architectural changes that require new instructions or addressing modes to be added to the compiler, there are only a few such devices. In most cases, these devices offer a 100% compatible mode of operation. Typically, we are able to add compiler support before these chips are available. Sometimes, however, this is not the case. When support is not yet available, you may use these new devices in the compatible mode of operation. When support is integrated into the compiler, assembler, and linker, the device database will be updated with the appropriate controls.

Once you compile, assemble, and link your program, you will need a method of testing it. The Keil uVision IDE supports two distinct methods of program testing: simulation and target debugging. In simulation, Keil Software or the silicon vendor has created a DLL that simulates the on-chip peripherals of the selected device. With over 350 devices in the database, it is impossible to provide simulation support for all of them. However, it is our goal to simulate as many as possible. Even if complete simulation is not available, partial simulation (base timers, counters, interrupts, and I/O ports) are supported. To discover what on-chip peripherals for a particular chip are simulated, check the Device Database.

Target debugging allows you to run your program on your target hardware. There are several different ways to do this. Each utilizes an interface DLL that is created either by Keil Software or by the silicon vendor or by the emulator vendor. The method of target debugging is via debugging hardware (JTAG and Serial) that is built into the chip. Companies like Triscend, Infineon, and Cygnal all have built-in debugging solutions. Contact your silicon vendor to see if they have uVision Debugger Drivers for their parts.

6.2 PCB DESIGN SOFTWARE (DIPTRACE):

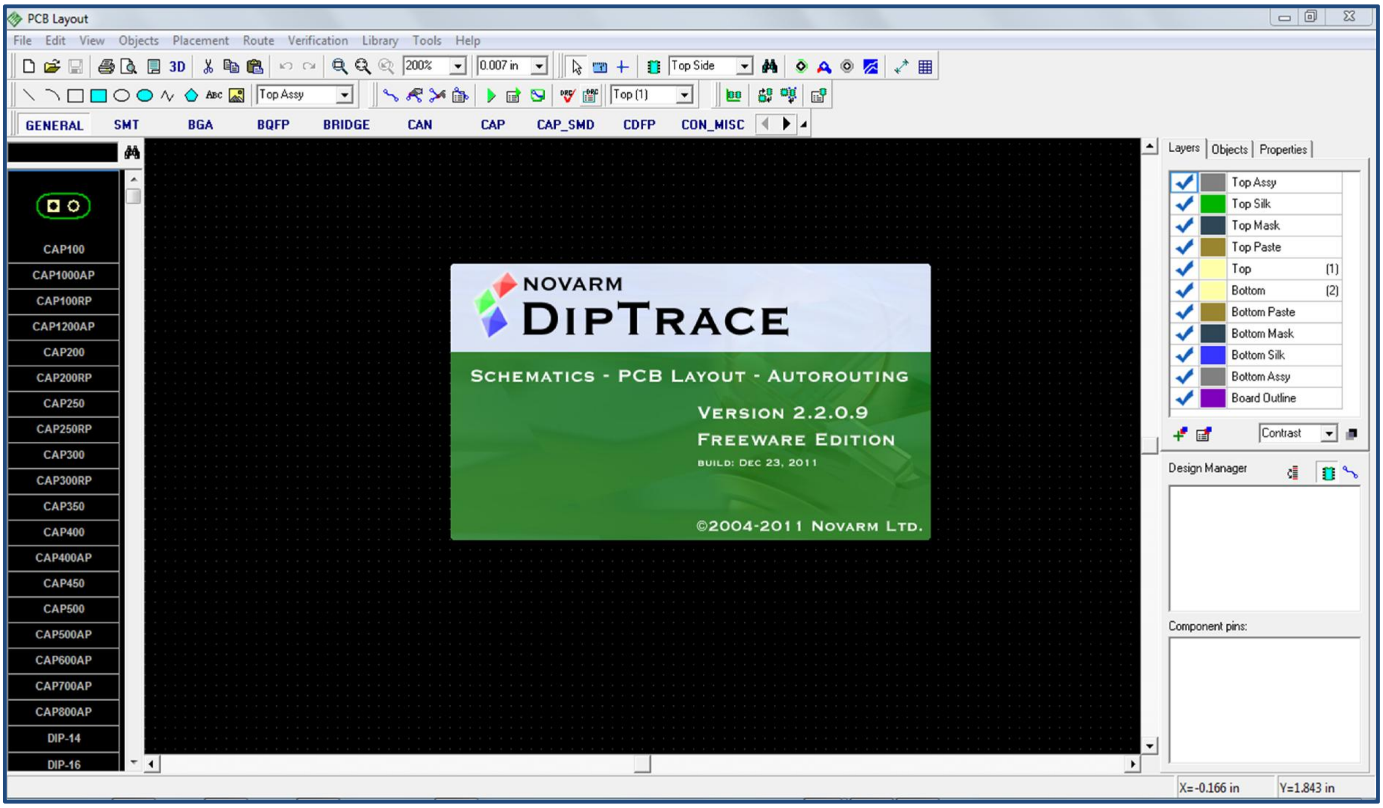


Fig: 6.3: Diptrace Snapshot.

6.2.1 DIPTRACE COMPLETE PCB DESIGN SYSTEM:

DipTrace is a complete PCB Design system. It includes four program modules:

1. **PCB Layout** - PCB design with easy-to-use manual routing tools, shape-based auto router, various verifications and 3D Preview with export.

2. **Schematic** - creates multi-sheet and multi-level hierarchical schematic and converts circuit to PCB Layout.

3. **Pattern Editor** - allows creating package footprints (patterns).

4. **Component Editor** - allows drawing parts and symbols and attaching patterns to them.

6.2.2 Features:

DipTrace provides following features:

- **Easy to learn user interface:**

To design a schematic, simply select and place components onto your document and connect them together using the wire and bus tools. Multi-sheet design is supported. Then select the menu option 'Switch to Board' to convert the schematic to PCB. Layout can be updated from Schematic in a few clicks at any time. When you create or edit design objects they are highlighted to improve your work. Step-by-step tutorial available from web-site guides you through the design process and allows to get started with ease.

- **Easy to use manual and powerful automatic routing:**

DipTrace PCB software includes an advanced automatic router that is able to route single-layer and multi-layer boards. It is available with a 'rip-up and retry' algorithm. Auto router achieves high completion rates by going back and re-routing nets to make space for connections that could not be routed on a previous pass. Intelligent manual routing tools allow you to create and edit traces by 90, 45 degree or without any limitations. Through, blind or buried vias can be used in automatic and manual routing. Unlimited board size is supported.

- **Multi-sheet and multi-level hierarchical circuit:**

Schematic module features multi-sheet and multi-level hierarchical structure. These features allow for easy and convenient design. Each sheet of the schematic can be converted into hierarchy block. Blocks can be inserted into main sheet and into each other. PCB Layout allows grouping components into blocks directly on the board and automatically applying placement and routing from one block to another.

- **Smart placement and auto-placement features:**

After converting Schematic to PCB layout place board outline and arrange components. Then use "Placement by list" for chips/connectors and auto-placement for other components to get acceptable result in a few minutes and start routing.

- **Import/Export features:**

Exchange schematics, layouts and libraries (DXF, Eagle, P-CAD, PADS, OrCAD). DipTrace supports netlists of Accel, Allegro, Mentor, PADS, P-CAD, Protel and Tango netlist formats.

- **Manufacturing output:**

DipTrace allows for Gerber RS-274X and Excellon N/C Drill files export. These file formats are the most popular among PCB manufacturers all over the world. TrueType fonts and raster images are exported as well. DXF output is available. DXF file can be easily converted to G-Code and used for CNC drilling machines (milling method).

- **Real-time 3D PCB preview with VRML 2.0 export:**

3D preview module allows rotating your board in three axes, zooming in and out in real time and adjusting color settings of the preview. DipTrace supports .vrml and .3ds models. More than 2,500 3D models are available for free. Export allows to open board in any software with VRML support.

- **Standard component libraries:**

DipTrace package includes component and pattern libraries with 100.000+ components from different manufacturers.

- **Creation of your own libraries:**

Component and Pattern Editors allow designing your own symbols and patterns. To create complete components, simply connect them together using Component Editor.

- **Advanced verification features:**

Schematic and PCB design modules have number of verification features that help to control project accuracy on different design stages: The ERC function in Schematic shows possible errors in pin connections and allows you to correct errors step-by-step. DRC function in PCB Layout module checks clearance between design objects, minimum size of tracks and drills. Real-time DRC checks each user's action in real time and shows errors while routing traces, moving or editing objects. Errors are displayed graphically and can be easily fixed. Net Connectivity Check verifies if all nets of the PCB are electrically connected. This feature uses traces, copper pour filled areas and shapes to control connectivity. Then reports broken and merged nets with area details. Comparing to Schematic allows you to check if routed PCB is identical to schematic.

6.3 Flash Magic:

FlashMagic

Version 3.13

Developed for Philips Semiconductors
© Embedded Systems Academy, Inc. 2000 - 2006
All Rights Reserved

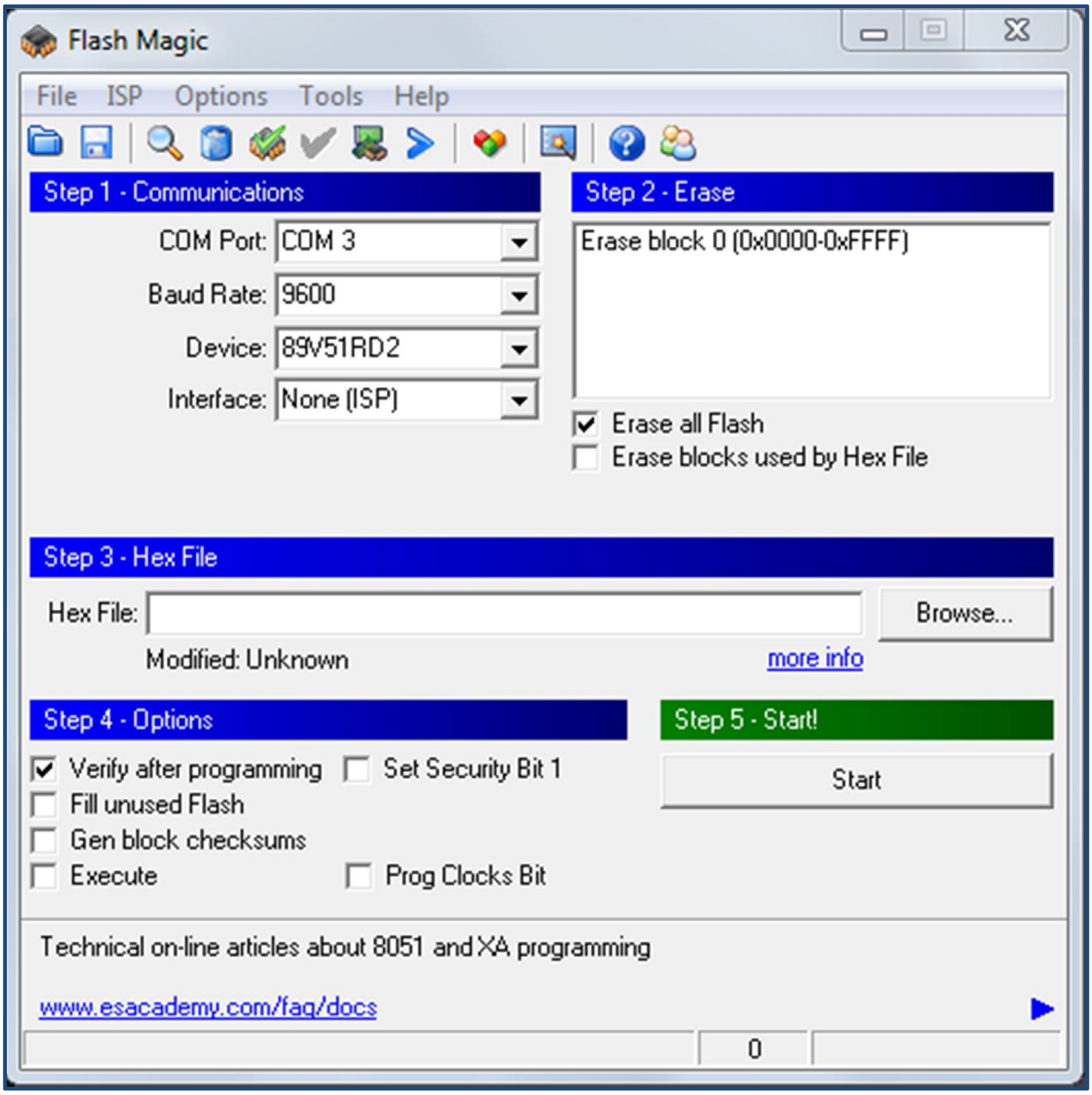
Fig. 6.4: Flash Magic snapshot-1.

Philips Semiconductors produce a range of Microcontrollers that feature both on-chip Flash memory and the ability to be reprogrammed using In-System Programming technology. Flash Magic is Windows software from the Embedded Systems Academy that allows easy access to all the ISP features provided by the devices.

6.3.1 Features:

- Erasing the Flash memory (individual blocks or the whole device)
- Programming the Flash memory
- Modifying the Boot Vector and Status Byte
- Reading Flash memory
- Performing a blank check on a section of Flash memory
- Reading the signature bytes
- Reading and writing the security bits
- Direct load of a new baud rate (high speed communications)
- Sending commands to place device in Boot loader mode

The window is divided up into five sections. Work your way from section 1 to section 5 to program a device using the most common functions. At the very bottom left of the window is an area where progress messages will be displayed and at the very bottom right is where the progress bar is displayed. In between the messages and the progress bar is a count of the number of times the currently selected hex file has been programmed since it was last modified or selected. Just above the progress information Embedded Hints are displayed. These are rotating Internet links that you can click on to go to a web page using your default browser.



Step 1 - Communications

COM Port: COM 3

Baud Rate: 9600

Device: 89V51RD2

Interface: None (ISP)

Step 2 - Erase

Erase block 0 (0x0000-0xFFFF)

☑ Erase all Flash

□ Erase blocks used by Hex File

Step 3 - Hex File

Hex File:

Modified: Unknown

Step 4 - Options

☑ Verify after programming

□ Set Security Bit 1

□ Fill unused Flash

□ Gen block checksums

□ Execute

□ Prog Clocks Bit

Step 5 - Start!

Technical on-line articles about 8051 and XA programming

www.esacademy.com/faq/docs

0

Fig. 6.5: Flash Magic Snapshot-2.

Flash Magic provides a clear and simple user interface to these features and more as described in the following sections. Under Windows, only one application may have access the COM Port at any one time, preventing other applications from using the COM Port. Flash Magic only obtains access to the selected COM Port when ISP operations are being performed. This means that other applications that need to use the COM Port, such as debugging tools, may be used while Flash Magic is loaded.

6.3.2 Minimum Requirements

- Windows 95/98/ME/NT/2000/XP
- Mouse
- COM Port
- 16Mb RAM
- 3Mb Disk Space

Component List

7.1 Component List:

Sr. No.	Component	Description	Reference	Quantity	Price
1.	Integrated Circuits	P89v51RD2BN		1	Rs. 400/-
		MAX232		1	Rs. 10/-
		ULN2003		2	Rs. 30/-
		7805		3	Rs. 30/-
2.	IC Base	40 Pin		1	Rs. 6/-
		16 Pin		3	Rs. 9/-
		6 Pin		1	Rs. 3/-
3.	Diodes	1N4007	D2-D4, D6-D10, D12-D22	19	Rs. 19/-
		1N5402	D1, D5, D11	3	Rs. 6/-
4.	Capacitors	2200 µF/25 Volt	C1	1	Rs. 4/-
		470 µF/25 Volt	C2	1	Rs. 4/-
		1 nF	C3, C11	1	Rs. 1/-
		1 µF/16 Volt	C7-C10	4	Rs. 4/-
		33 pF	C4, C5	2	Rs. 1/-
		10 µF/16 Volt	C6	1	Rs. 1/-
5.	Resisters	10 KΩ/ ¼ watt	R1, R3, R6, R8, R9	5	Rs. 1/-
		1 KΩ / ¼ watt	R2, R4, R5	3	Rs. 1/-
		220 Ω/ 1 watt	R7	1	Rs. 2/-
		10 kΩ Trim pot	CONTRAST	1	Rs. 6/-
		A472J (Resister bank)	RN1-RN4	4	Rs. 2/-
6.	Crystal	11.0592 MHz	X1	1	Rs. 10/-
7.	Connecters	DB9 (Female-90°)	ISP	1	Rs. 20/-
		DB9 (Male)		2	Rs. 20/-
		2 Pin Relimate (M-F Pair)		2	Rs. 6/-
		12 Pin Relimate (M-F Pair)		1	Rs. 4/-
		16 Pin Relimate (M-F Pair)		1	Rs. 6/-
		4mm Power Connecters (M-F Pair)		20	Rs. 180/-
		Screw Connecters (2 Pin)		39	Rs. 117/-
		DC Power Connecter (M)	POWER	1	Rs. 8/-
		DC Power Connecter (F)		2	Rs. 16/-

Sr. No.	Component	Description	Reference	Quantity	Price
8.	LCD Display	16*2		1	Rs. 140/-
9.	Relays	5V	Relay1-Relay5, Relay7-Relay13	12	Rs. 180/-
		12V	Relay6	1	Rs. 20/-
10.	Switches	Slider Switch		1	Rs. 8/-
		Push to ON Switch	S1	1	Rs. 4/-
11.	LEDs	5mm (Red)	LED	1	Rs. 1/-
12.	LED Holder	5mm (Metal)		1	Rs. 2/-
13.	Plastic Box	8” X 10”		1	Rs. 85/-
14.	PCB	15cm * 10cm		1	Rs. 285/-
15.	GSM Modem	SIM900		1	Rs. 1200/-
16.	Alarm	12v (Solid State)		1	Rs. 30/-
17.	Siren	12v (Solid State)		1	Rs. 180/-
18.	Electronic Lock	Metal		1	Rs. 60/-
19.	Power Supply	DC to DC buck converter		1	Rs. 120/-
20.	Wires	1 Core, 4 Core & 10 Core		-	Rs. 100/-
				TOTAL =	**Rs. 3,288/-**

Table 7.1: Component list.

Overview of the work

8.1 Overview of the work that has to be done:

1. Field survey to understand basic requirements.
2. Market survey for the available components.
3. To choose the proper wireless communication medium.
4. Circuit and PCB layout design.
5. C Program development and testing on target device.
6. Testing of the system for reliability and GSM system performance.
7. Front panel and casing design for this system to fix it in HLP (High Lift Platform) driver cabin.
8. Sensors testing and installation in HLP.
9. Installation and final testing of the system in HLP.

8.2 Design Specifications:

1) Power Supply required: 12 Volts/ 3 Ampere.
2) Display: 16*2 LCD Display.
3) Controller: P89v51RD2BN.
4) In System Programmable.
5) Number of connections for Sensors: 8
6) Wireless communication over GSM network using GSM modem (SIM900A).
7) Controls eight devices by sending commands through SMS.
8) Maximum five mobile numbers can be added to the system.
9) Can be powered ON and OFF using a key of electronic lock only.
10) Siren and alarm working on 12 volt DC power supply.
11) Both active and passive sensors can be interfaced with this system.
12) Short circuit protection circuit for 12 volt DC power supply.

Results

9.1 Assembled circuit for the system:

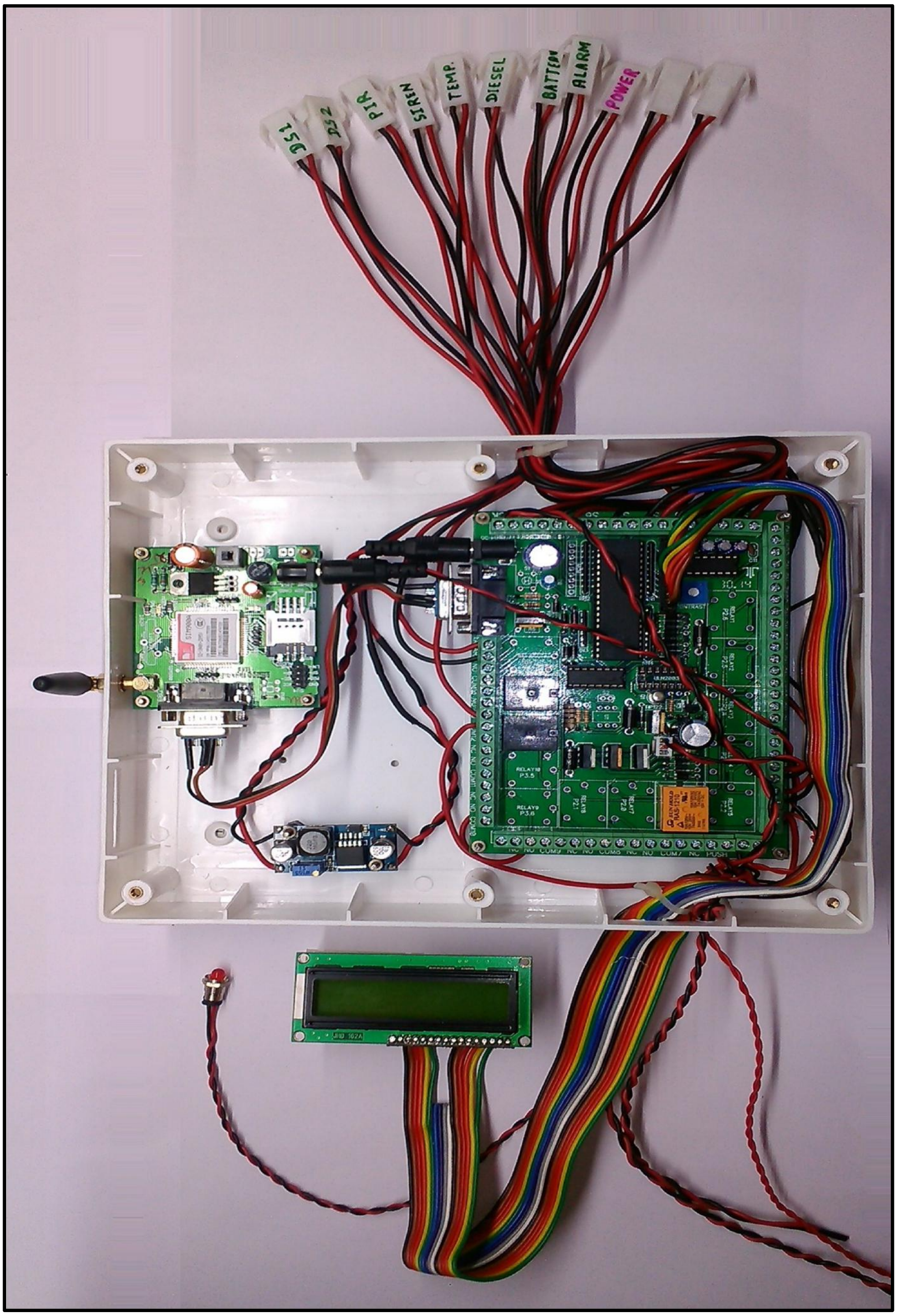


Fig. 9.3: Assembled circuit for the system.

9.2 Installed Sensors:

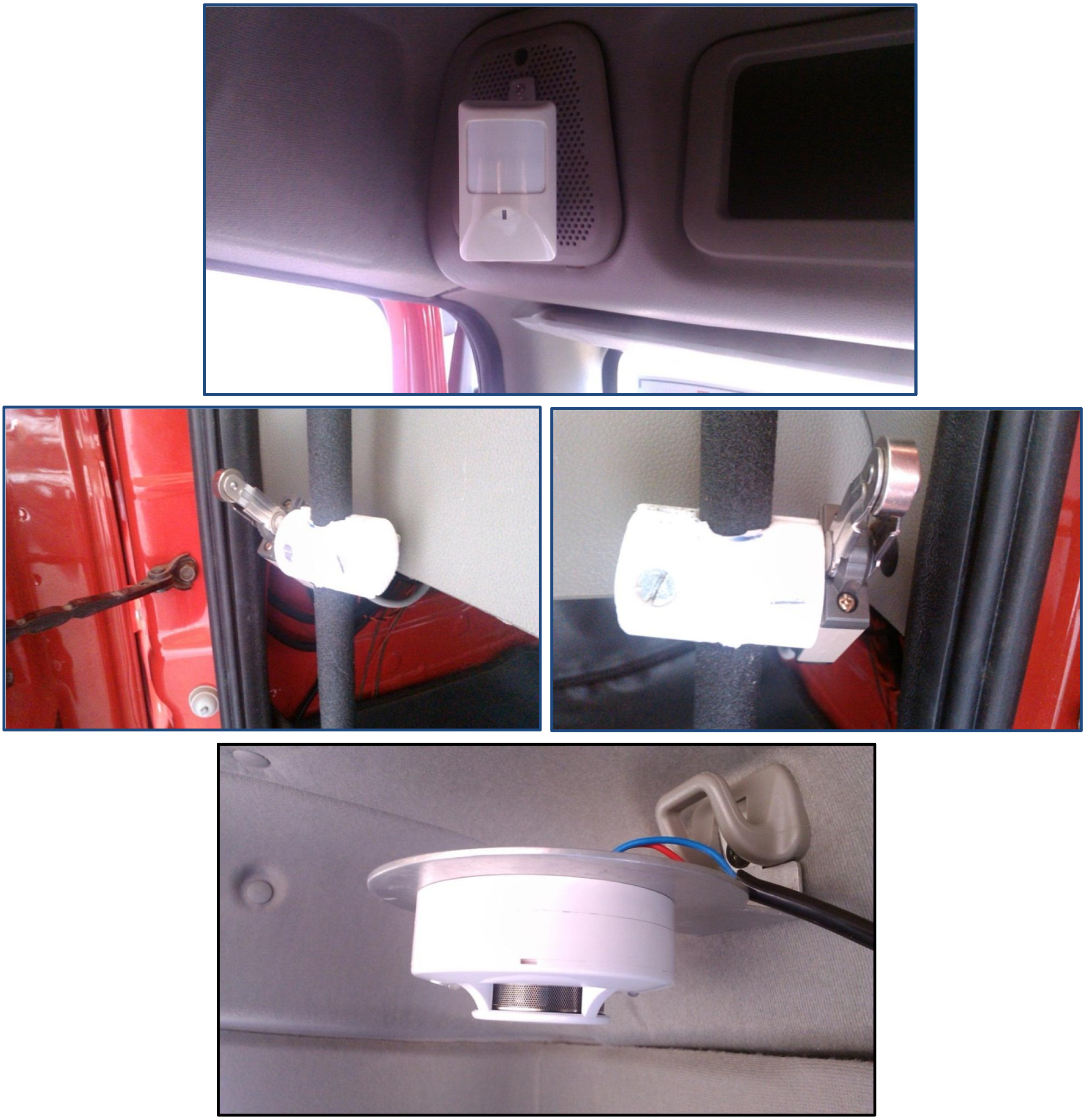


Fig. 9.3: Sensors installed in HLP.

9.3 Front Panel installed in HLP:

To fix this system in HLP we have made a base with front panel. Following figure shows the front panel with LCD Display, GSM modem ON/OFF switch and POWER indication LED. This front panel is made up of Aluminium.

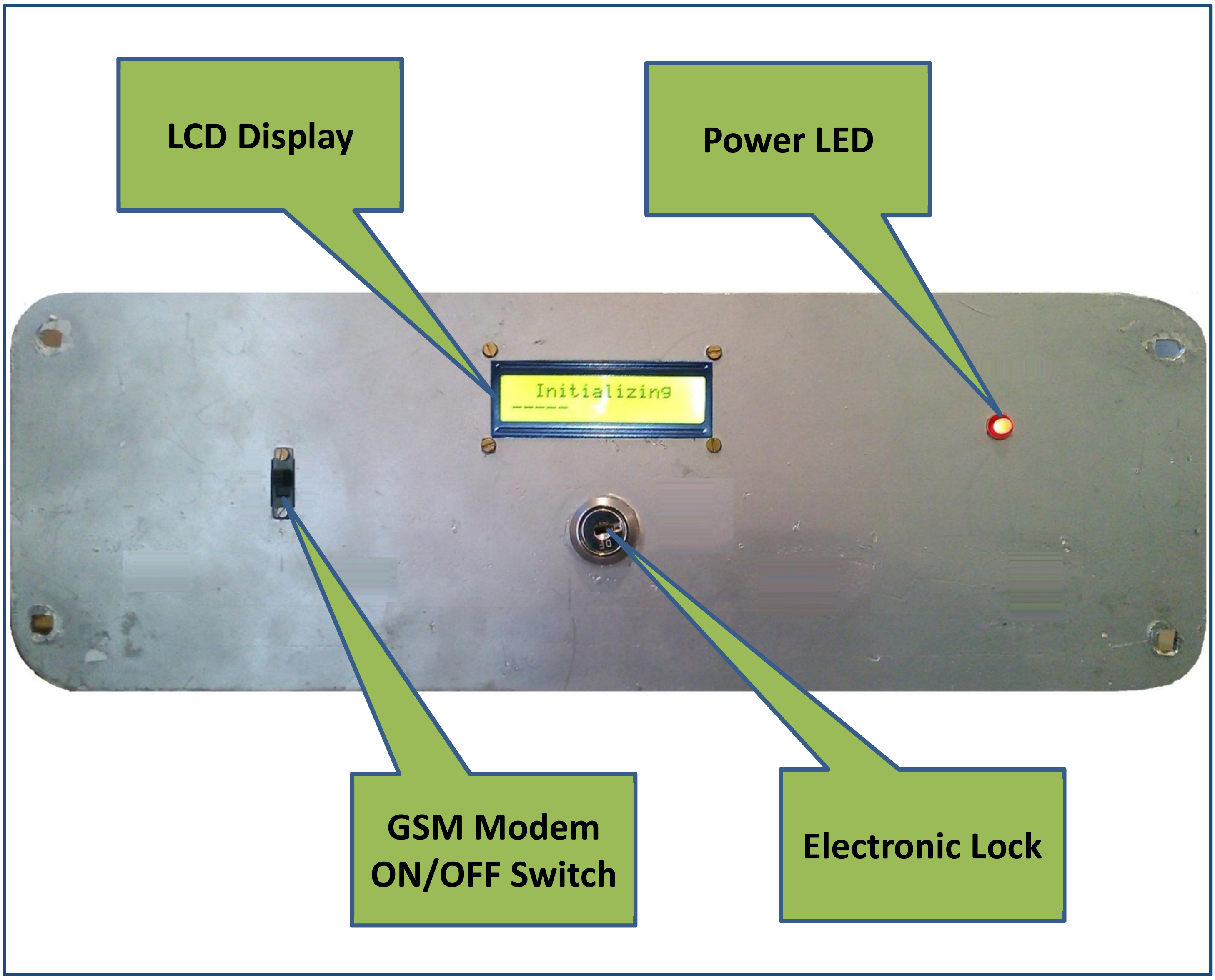


Fig. 9.3: Front panel of System for HLP.

PROGRAM

```
#include<reg51.h>

sbit sensor1  = P1^0;
sbit sensor2  = P1^1;
sbit sensor3  = P1^2;
sbit sensor4  = P1^3;
sbit sensor5  = P1^4;
sbit sensor6  = P1^5;
sbit sensor7  = P1^6;
sbit sensor8  = P1^7;

sbit dev1     = P2^0;
sbit dev2     = P2^1;
sbit dev3     = P2^2;
sbit dev4     = P2^3;
sbit dev5     = P2^4;
sbit dev6     = P2^5;
sbit dev7     = P2^6;
sbit dev8     = P2^7;

sbit tswitch  = P3^2;
sbit siren    = P3^3;
sbit ts1      = P3^4;
sbit ts2      = P3^5;
sbit ts3      = P3^6;
sbit alarm    = P3^7;

sbit sso     = P0^0;
sbit ssi     = P0^1;
sbit rs      = P0^2;
sbit en      = P0^3;
sfr DATA = 0x80;           //port0 as lcd data

void LCDINIT(void);
void SERIALINIT();
void GSMINIT();
void serial(unsigned char x);
void Send2Gsm(char *aaa);
void Send2PC(char *aaa);
void LCDINIT_Write(unsigned char a);
void LCDCOMMAND(unsigned char);
void LCDDATA(unsigned char);
void send_string(unsigned char *var);
void MSDELAY(unsigned int);
char getCharacter (void);
char getNumber (void);
char getResp (void);
void SMScon();

unsigned char
autnum1[15]=("\"+918380934131\"");
unsigned long d=0;
unsigned int i=0, j=0, count=0, ss=0, s1=0,
s2=0, s3=0, s4=0, s5=0, s6=0, s7=0, n1, n2,
n3, n4, n5, a=0, b=0, c=0, dd=0, e=0, f=0,
z=0, repeat=0;
unsigned int p=0, q=0, r=0, l=0, m=0, n=0,
k=0, idea=0, cellone=0, vodaphone=0,
airtel=0, bsnl=0, docomo=0, auth=0, ssc=0,
gg=0, hh=0, sirenoff=0;
unsigned char idea1[4]=("IDEA"),
cellone1[4]=("Cell"),
vodaphone1[4]=("Voda"),
airtel1[4]=("AirT"), bsnl1[4]=("BSNL"),
docomo1[4]=("TATA");
unsigned char operator[4]=("    "),
chr1[1]='0',
Num[45]=("AT+CPBW=1,\"+91000000000
0\",129,\"RECEIVER1\""), resp;
unsigned char autnum[15]={0},
msg7[5]=("Statu"), charr[85]={0},
count1[3]={0}, chnum[10]={0},
New[10]={0};
unsigned char
char1[30]=("AT+CMGS=\"+91
\"\r\n"), char2[30]=("AT+CMGS=\"+91
\"\r\n"), char3[30]=("AT+CMGS=\"+91
\"\r\n");
unsigned char
char4[30]=("AT+CMGS=\"+91
\"\r\n"), char5[30]=("AT+CMGS=\"+91
\"\r\n"), string1[100], bal[160]={0};
unsigned char msg[5]={0},
msg1[5]=("Num1 "), msg2[5]=("Num2 "),
msg3[5]=("Num3 "), msg4[5]=("Num4 "),
msg5[5]=("Num5 "), msg6[5]=("Get n");

void timer0_isr(void) interrupt 1
{
TF0=0;
if(tswitch==1)
{
while(tswitch==1)
{
}
ts1=~ts1;
if(p==0)
{
ts2=1;
MSDELAY(200);
ts2=0;
p=p+1;
}
else if(p!=0)
{
ts3=1;
MSDELAY(200);
ts3=0;
p=0;
}
}
}

void main()
{
P1=0xFF;
P2=0x00;
P3=0x03;
P0=0x03;
ss=0;
dev1=0;
dev2=0;
dev3=0;
dev4=0;
dev5=0;
dev6=0;
dev7=0;
dev8=0;
MSDELAY(50);
LCDINIT();
alarm=1;
sso=1;
count1[1]='\r';
count1[2]='\n';
sso=1;
send_string("  Initializing  ");
sso=1;
MSDELAY(2000);
sso=0;
LCDCOMMAND(0xC0);
MSDELAY(20);
LCDDATA('-');
SERIALINIT();
MSDELAY(2000);
LCDDATA('-');
MSDELAY(1500);
LCDDATA('-');
MSDELAY(1000);
LCDDATA('-');
MSDELAY(1000);
LCDDATA('-');

GSMINIT();
Num[8]=Num[39]='1';
j=0;
for(i=10;i<24;i++)
{
Num[i]=autnum1[j];
j=j+1;
}
```

```
Send2Gsm(Num);
Send2Gsm("\r\n");
MSDELAY(500);
LCDDATA('-');

z=0;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("AT+CPBR=6\r\n");
while(string1[0]!='C' && string1[0]!='K')
{
string1[0]=getCharacter();
z=z+1;
if(z==100)
break;
}

if(string1[0]=='K')
{
count=1;
Send2Gsm("AT+CPBW=6,\"+91111111111
1\",129,\"RECEIVER6\"");
MSDELAY(100);
}

if(string1[0]=='C')
{
for(i=0;i<50;i++)
{
string1[i]=getCharacter();
}
i=0;
while(string1[i]!='"')
{
i=i+1;
if(i>50)
{
break;
}
}
```

```
if(i<50)
{
i=i+7;
count=string1[i];
}
else
{
LCDINIT();
send_string("Problem to read ");
LCDCOMMAND(0xC0);
send_string("the Phone Book  ");
}
}

if(count==0)
{
LCDINIT();
send_string("  GSM Modem is  ");
LCDCOMMAND(0xC0);
send_string("  Disconnected  ");
}
else
{
MSDELAY(300);
LCDDATA('-');
z=0;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("AT+CPBR=1\r\n");
while(string1[0]!='C' && string1[0]!='K')
{
string1[0]=getCharacter();
z=z+1;
if(z==100)
break;
}

if(string1[0]=='K')
{
n1=1;
string1[0]=' ';
}

if(string1[0]=='C')
{
n1=0;
for(i=0;i<50;i++)
{
string1[i]=getCharacter();
}
i=0;
while(string1[i]!='"')
{
i=i+1;
if(i>50)
{
break;
}
}
if(i<50)
{
i=i+1;
j=9;
while(string1[i]!='"')
{
char1[j]=string1[i];
j=j+1;
i=i+1;
MSDELAY(1);
}
}
else
{
LCDINIT();
send_string("Problem to read ");
LCDCOMMAND(0xC0);
send_string("1st Receiver Num");
}
}
```

```
if(count>'1')
{
MSDELAY(300);
LCDDATA('-');
z=0;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("AT+CPBR=2\r\n");
while(string1[0]!='C' && string1[0]!='K')
{
string1[0]=getCharacter();
z=z+1;
if(z==100)
break;
}

if(string1[0]=='K')
{
n2=1;
string1[0]=' ';
}

if(string1[0]=='C')
{
n2=0;
for(i=0;i<50;i++)
{
string1[i]=getCharacter();
}
i=0;
while(string1[i]!='"')
{
i=i+1;
if(i>50)
{
break;
}
}
if(i<50)
{
```

```
i=i+1;
j=9;
while(string1[i]!='"')
{
char2[j]=string1[i];
j=j+1;
i=i+1;
MSDELAY(1);
}
}
else
{
LCDINIT();
send_string("Problem to read ");
LCDCOMMAND(0xC0);
send_string("2nd Receiver Num");
}
}
}

if(count>'2')
{
MSDELAY(300);
LCDDATA('-');
z=0;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("AT+CPBR=3\r\n");
while(string1[0]!='C' && string1[0]!='K')
{
string1[0]=getCharacter();
z=z+1;
if(z==100)
break;
}

if(string1[0]=='K')
{
n3=1;
string1[0]=' ';
}

if(string1[0]=='C')
{
n3=0;
for(i=0;i<50;i++)
{
string1[i]=getCharacter();
}
i=0;
while(string1[i]!='"')
{
i=i+1;
if(i>50)
{
break;
}
}
if(i<50)
{
i=i+1;
j=9;
while(string1[i]!='"')
{
char3[j]=string1[i];
j=j+1;
i=i+1;
MSDELAY(1);
}
}
else
{
LCDINIT();
send_string("Problem to read ");
LCDCOMMAND(0xC0);
send_string("3rd Receiver Num");
}
}
}
```

```
if(count>'3')
{
MSDELAY(300);
LCDDATA('-');
z=0;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("AT+CPBR=4\r\n");
while(string1[0]!='C' && string1[0]!='K')
{
string1[0]=getCharacter();
z=z+1;
if(z==100)
break;
}

if(string1[0]=='K')
{
n4=1;
string1[0]=' ';
}

if(string1[0]=='C')
{
n4=0;
for(i=0;i<50;i++)
{
string1[i]=getCharacter();
}
i=0;
while(string1[i]!='"')
{
i=i+1;
if(i>50)
{
break;
}
}
if(i<50)
{
i=i+1;
j=9;
while(string1[i]!='"')
{
char4[j]=string1[i];
j=j+1;
i=i+1;
MSDELAY(1);
}
}
else
{
LCDINIT();
send_string("Problem to read ");
LCDCOMMAND(0xC0);
send_string("4th Receiver Num");
}
}
}

if(count>'4')
{
MSDELAY(300);
LCDDATA('-');
z=0;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("AT+CPBR=5\r\n");
while(string1[0]!='C' && string1[0]!='K')
{
string1[0]=getCharacter();
z=z+1;
if(z==100)
break;
}

if(string1[0]=='K')
{
n5=1;
string1[0]=' ';
}

if(string1[0]=='C')
{
n5=0;
for(i=0;i<50;i++)
{
string1[i]=getCharacter();
}
i=0;
while(string1[i]!='"')
{
i=i+1;
if(i>50)
{
break;
}
}
if(i<50)
{
i=i+1;
j=9;
while(string1[i]!='"')
{
char5[j]=string1[i];
j=j+1;
i=i+1;
MSDELAY(1);
}
}
else
{
LCDINIT();
send_string("Problem to read ");
LCDCOMMAND(0xC0);
send_string("5th Receiver Num");
}
}
}
MSDELAY(500);
```

```
LCDDATA('-');
z=0;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("AT+COPS?\r\n");
while(operator[0]!='"')
{
operator[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
if(z<100)
{
for(i=0;i<4;i++)
{
operator[i]=getCharacter();
}
for(i=0;i<4;i++)
{
if(operator[i]!=airtel1[i])
{
airtel=1;
}
}
for(i=0;i<4;i++)
{
if(operator[i]!=idea1[i])
{
idea=1;
}
}
for(i=0;i<4;i++)
{
if(operator[i]!=cellone1[i])
{
cellone=1;
}
}
for(i=0;i<4;i++)
```

```
{
if(operator[i]!=bsnl1[i])
{
bsnl=1;
}
}
for(i=0;i<4;i++)
{
if(operator[i]!=vodaphone1[i])
{
vodaphone=1;
}
}
for(i=0;i<4;i++)
{
if(operator[i]!=docomo1[i])
{
docomo=1;
}
}
LCDDATA('-');

if(idea==0)
{
z=0;
k=1;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("ATD*121#\r\n");
while(bal[0]!=':')
{
bal[0]=getCharacter();
z=z+1;
if(z>1000)
break;
}
getCharacter();
getCharacter();
getCharacter();
getCharacter();
```

```
for(i=0;i<160;i++)
{
bal[i]=getCharacter();
}
i=0;
while(bal[i]!='R' || bal[i-1]!='N' || bal[i-
2]!='I')
{
i=i+1;
if(i>160)
break;
}
i=i+3;
bal[i]='"';
}

if(airtel==0)
{
z=0;
k=1;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("ATD*123#\r\n");
while(bal[0]!='"')
{
bal[0]=getCharacter();
z=z+1;
if(z>1000)
break;
}
for(i=0;i<160;i++)
{
bal[i]=getCharacter();
}
}

if(docomo==0)
{
z=0;
k=1;
```

```
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("ATD*111#\r\n");
while(bal[0]!='"')
{
bal[0]=getCharacter();
z=z+1;
if(z>1000)
break;
}
for(i=0;i<160;i++)
{
bal[i]=getCharacter();
}
}

if(cellone1==0)
{
z=0;
k=1;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("ATD*123#\r\n");
while(bal[0]!='"')
{
bal[0]=getCharacter();
z=z+1;
if(z>1000)
break;
}
for(i=0;i<160;i++)
{
bal[i]=getCharacter();
}
}

if(vodaphone1==0)
{
z=0;
k=1;
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("ATD*141#\r\n");
while(bal[0]!='"')
{
bal[0]=getCharacter();
z=z+1;
if(z>1000)
break;
}
for(i=0;i<160;i++)
{
bal[i]=getCharacter();
}
}
}
}
LCDDATA('-');

while(1)
{
IE=0x00;
TR0=0;
a=0;
b=0;
c=0;
d=0;
dd=0;
e=0;
f=0;
p=0;
q=0;
d=0;
r=0;
l=0;
m=0;
n=0;
gg=0;
hh=0;
ts1=0;
```

```
ts2=0;
ts3=0;
siren=0;
ss=0;
GSMINIT();
alarm=1;
z=0;
MSDELAY(200);
LCDDATA('-');
if(P1!=0x00)
{
if(ss==0 && ssi==0)
{
ss=1;
MSDELAY(200);
alarm=1;
LCDCOMMAND(0x80);
send_string(" Short Circuit  ");
LCDCOMMAND(0xC0);
send_string(" In the System  ");
MSDELAY(200);
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
Send2Gsm("\r\n");
if(k==1)
{
i=0;
while(bal[i]!='"')
{
serial(bal[i]);
i=i+1;
}
}
MSDELAY(30);
serial(0x1A);
}
if(count>'1' && n2==0)
{
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
if(chr1[0]=='R')
{
repeat=1;
}
chr1[0]=' ';
z=0;
Send2Gsm(char2);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'2' && n3==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char3);
MSDELAY(60);
```

```
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'3' && n4==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char4);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'4' && n5==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char5);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
MSDELAY(200);
}
if(repeat==1)
{
MSDELAY(500);
repeat=0;
Send2Gsm(char1);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
}
}
}
else
{
MSDELAY(1000);
alarm=1;
LCDCOMMAND(0x80);
send_string("                ");
LCDCOMMAND(0xC0);
send_string("                ");
LCDCOMMAND(0x80);
send_string("  System Error  ");
MSDELAY(100);
repeat=0;
if(count!=0)
{
if(n1==0)
{
```

```
Send2Gsm(char1);
MSDELAY(60);
Send2Gsm("System Error.\r\n");
Send2Gsm("\r\n");
if(k==1)
{
i=0;
while(bal[i]!="")
{
serial(bal[i]);
i=i+1;
}
}
MSDELAY(30);
serial(0x1A);
}
if(count>'1' && n2==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>200)
break;
}
if(chr1[0]=='R')
{
repeat=1;
}
chr1[0]=' ';
z=0;
Send2Gsm(char2);
MSDELAY(60);
Send2Gsm("System Error.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'2' && n3==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>200)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char3);
MSDELAY(60);
Send2Gsm("System Error.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'3' && n4==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>200)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char4);
MSDELAY(60);
Send2Gsm("System Error.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'4' && n5==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>200)
break;
}
```

```
chr1[0]=' ';
z=0;
Send2Gsm(char5);
MSDELAY(60);
Send2Gsm("System Error.\r\n");
MSDELAY(30);
serial(0x1A);
MSDELAY(200);
}
if(repeat==1)
{
MSDELAY(300);
repeat=0;
Send2Gsm(char1);
MSDELAY(60);
Send2Gsm("System Error.\r\n");
MSDELAY(30);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>200)
break;
}
chr1[0]=' ';
}
}
MSDELAY(2000);
LCDCOMMAND(0x80);
send_string("Error in sensor:");
LCDCOMMAND(0xC0);
if(sensor1==1)
{
send_string("1 ");
}
if(sensor2==1)
{
send_string("2 ");
}
if(sensor3==1)
{
send_string("3 ");
}
if(sensor4==1)
{
send_string("4 ");
}
if(sensor5==1)
{
send_string("5 ");
}
if(sensor6==1)
{
send_string("6 ");
}
if(sensor7==1)
{
send_string("7 ");
}
if(sensor8==1)
{
send_string("8 ");
}
}
while(P1!=0x00)
{
}
break;
}

else if(P1==0x00)
{
GSMINIT();
LCDDATA('-');
MSDELAY(1500);
alarm=0;
siren=0;
LCDCOMMAND(0x80);
```

```
send_string("                ");
LCDCOMMAND(0xC0);
send_string("                ");
LCDCOMMAND(0x80);
send_string("   Welcome to   ");
LCDCOMMAND(0xC0);
send_string("Security System ");
MSDELAY(200);
IE=0x82;
TR0=1;
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(60);
Send2Gsm("Welcome to Security
System.\r\n");
Send2Gsm("\r\n");
if(k==1)
{
i=0;
while(bal[i]!='"')
{
serial(bal[i]);
i=i+1;
}
}
MSDELAY(30);
serial(0x1A);
}
if(count>'1' && n2==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
if(chr1[0]=='R')
{
repeat=1;
}
chr1[0]=' ';
z=0;
Send2Gsm(char2);
MSDELAY(60);
Send2Gsm("Welcome to Security
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'2' && n3==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char3);
MSDELAY(60);
Send2Gsm("Welcome to Security
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'3' && n4==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
```

```
}
chr1[0]=' ';
z=0;
Send2Gsm(char4);
MSDELAY(60);
Send2Gsm("Welcome to Security
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'4' && n5==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char5);
MSDELAY(60);
Send2Gsm("Welcome to Security
System.\r\n");
MSDELAY(30);
serial(0x1A);
MSDELAY(200);
}
if(repeat==1)
{
MSDELAY(300);
repeat=0;
Send2Gsm(char1);
MSDELAY(60);
Send2Gsm("Welcome to Security
System.\r\n");
MSDELAY(30);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
}
}
}
IE=0x82;
TR0=1;
MSDELAY(100);
RI=0;
while(P1==0x00)
{
if(ss==0 && ssi==0)
{
ss=1;
MSDELAY(200);
alarm=1;
LCDCOMMAND(0x80);
send_string(" Short Circuit  ");
LCDCOMMAND(0xC0);
send_string(" In the System  ");
MSDELAY(200);
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
Send2Gsm("\r\n");
if(k==1)
{
```

```
i=0;
while(bal[i+1]!='"')
{
serial(bal[i]);
i=i+1;
}
}
MSDELAY(30);
serial(0x1A);
}
if(count>'1' && n2==0)
{
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
if(chr1[0]=='R')
{
repeat=1;
}
chr1[0]=' ';
z=0;
Send2Gsm(char2);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'2' && n3==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char3);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'3' && n4==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char4);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'4' && n5==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
```

```
Send2Gsm(char5);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
MSDELAY(200);
}
if(repeat==1)
{
MSDELAY(500);
repeat=0;
Send2Gsm(char1);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
}
}
}

if(RI==1)
{
IE=0x00;
TR0=0;
SMScon();
RI=0;
IE=0x82;
TR0=1;
}
```

```
IE=0x82;
TR0=1;
}
MSDELAY(5);

if(P1!=0x00)
{
MSDELAY(5);
LCDCOMMAND(0xC0);
send_string("                ");
LCDCOMMAND(0x80);
send_string("Detected Sensor:");
alarm=1;
LCDCOMMAND(0xC0);

while(1)
{
IE=0x82;
TR0=1;

if(ss==0 && ssi==0)
{
ss=1;
MSDELAY(200);
alarm=1;
LCDCOMMAND(0x80);
send_string(" Short Circuit  ");
LCDCOMMAND(0xC0);
send_string(" In the System  ");
MSDELAY(200);
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
Send2Gsm("\r\n");
```

```
if(k==1)
{
i=0;
while(bal[i+1]!='"')
{
serial(bal[i]);
i=i+1;
}
}
MSDELAY(30);
serial(0x1A);
}
if(count>'1' && n2==0)
{
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
if(chr1[0]=='R')
{
repeat=1;
}
chr1[0]=' ';
z=0;
Send2Gsm(char2);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'2' && n3==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char3);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'3' && n4==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
z=0;
Send2Gsm(char4);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
}
if(count>'4' && n5==0)
{
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
```

```
chr1[0]=' ';
z=0;
Send2Gsm(char5);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
MSDELAY(200);
}
if(repeat==1)
{
MSDELAY(500);
repeat=0;
Send2Gsm(char1);
MSDELAY(60);
Send2Gsm("Short Circuit in the
System.\r\n");
MSDELAY(30);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
chr1[0]=' ';
}
}
}

if(RI==1)
{
IE=0x00;
TR0=0;
SMScon();
RI=0;
IE=0x82;
TR0=1;
}

if(sensor1==1 && a==0)
{
alarm=1;
if(a==0)
{
send_string("1 ");
a=1;
}
if(m==0)
{
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S1.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'1' && n2==0)
{
MSDELAY(1200);
Send2Gsm(char2);
MSDELAY(50);
Send2Gsm("Detected Sensor S1.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'2' && n3==0)
{
MSDELAY(1200);
Send2Gsm(char3);
MSDELAY(50);
Send2Gsm("Detected Sensor S1.\r\n");
MSDELAY(20);
```

```
serial(0x1A);
}
if(count>'3' && n4==0)
{
MSDELAY(1200);
Send2Gsm(char4);
MSDELAY(50);
Send2Gsm("Detected Sensor S1.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'4' && n5==0)
{
MSDELAY(1200);
Send2Gsm(char5);
MSDELAY(50);
Send2Gsm("Detected Sensor S1.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
if(repeat==1)
{
repeat=0;
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S1.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
}
}
m=1;
}


if(sensor2==1 && b==0)
{
alarm=1;
if(b==0)
{
send_string("2 ");
b=1;
}
if(m==0)
{
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S2.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'1' && n2==0)
{
MSDELAY(1200);
Send2Gsm(char2);
MSDELAY(50);
Send2Gsm("Detected Sensor S2.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'2' && n3==0)
{
MSDELAY(1200);
Send2Gsm(char3);
MSDELAY(50);
Send2Gsm("Detected Sensor S2.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'3' && n4==0)
{
MSDELAY(1200);
Send2Gsm(char4);
```

```
MSDELAY(50);
Send2Gsm("Detected Sensor S2.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'4' && n5==0)
{
MSDELAY(1200);
Send2Gsm(char5);
MSDELAY(50);
Send2Gsm("Detected Sensor S2.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
if(repeat==1)
{
repeat=0;
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S2.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
}
}
m=1;
}

if(sensor3==1 && c==0)
{
alarm=1;
if(c==0)
{
send_string("3 ");
c=1;
}
if(m==0)
{
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S3.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'1' && n2==0)
{
MSDELAY(1200);
Send2Gsm(char2);
MSDELAY(50);
Send2Gsm("Detected Sensor S3.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'2' && n3==0)
{
MSDELAY(1200);
Send2Gsm(char3);
MSDELAY(50);
Send2Gsm("Detected Sensor S3.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'3' && n4==0)
{
MSDELAY(1200);
Send2Gsm(char4);
MSDELAY(50);
Send2Gsm("Detected Sensor S3.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'4' && n5==0)
{
```

```
MSDELAY(1200);
Send2Gsm(char5);
MSDELAY(50);
Send2Gsm("Detected Sensor S3.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
if(repeat==1)
{
repeat=0;
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S3.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
}
}
m=1;
}

if(sensor4==1 && dd==0)
{
alarm=1;
if(dd==0)
{
send_string("4 ");
dd=1;
}
if(m==0)
{
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S4.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'1' && n2==0)
{
MSDELAY(1200);
Send2Gsm(char2);
MSDELAY(50);
Send2Gsm("Detected Sensor S4.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'2' && n3==0)
{
MSDELAY(1200);
Send2Gsm(char3);
MSDELAY(50);
Send2Gsm("Detected Sensor S4.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'3' && n4==0)
{
MSDELAY(1200);
Send2Gsm(char4);
MSDELAY(50);
Send2Gsm("Detected Sensor S4.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'4' && n5==0)
{
MSDELAY(1200);
Send2Gsm(char5);
MSDELAY(50);
Send2Gsm("Detected Sensor S4.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
```

```
}
if(repeat==1)
{
repeat=0;
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S4.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
}
}
m=1;
}

if(sensor5==1 && e==0)
{
alarm=1;
if(e==0)
{
send_string("5 ");
e=1;
}
if(m==0)
{
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S5.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'1' && n2==0)
{
MSDELAY(1200);
Send2Gsm(char2);
MSDELAY(50);
Send2Gsm("Detected Sensor S5.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'2' && n3==0)
{
MSDELAY(1200);
Send2Gsm(char3);
MSDELAY(50);
Send2Gsm("Detected Sensor S5.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'3' && n4==0)
{
MSDELAY(1200);
Send2Gsm(char4);
MSDELAY(50);
Send2Gsm("Detected Sensor S5.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'4' && n5==0)
{
MSDELAY(1200);
Send2Gsm(char5);
MSDELAY(50);
Send2Gsm("Detected Sensor S5.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
if(repeat==1)
{
repeat=0;
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S5.\r\n");
```

```
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
}
}
m=1;
}

if(sensor6==1 && f==0)
{
alarm=1;
if(f==0)
{
send_string("6 ");
f=1;
}
if(m==0)
{
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S6.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'1' && n2==0)
{
MSDELAY(1200);
Send2Gsm(char2);
MSDELAY(50);
Send2Gsm("Detected Sensor S6.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'2' && n3==0)
{
MSDELAY(1200);
Send2Gsm(char3);
MSDELAY(50);
Send2Gsm("Detected Sensor S6.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'3' && n4==0)
{
MSDELAY(1200);
Send2Gsm(char4);
MSDELAY(50);
Send2Gsm("Detected Sensor S6.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'4' && n5==0)
{
MSDELAY(1200);
Send2Gsm(char5);
MSDELAY(50);
Send2Gsm("Detected Sensor S6.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
if(repeat==1)
{
repeat=0;
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S6.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
}
}
m=1;
```

```
}

if(sensor7==1 && l==0)
{
//alarm=1;
if(l==0)
{
if(siren==0)
{
alarm=1;
}
if(gg==0)
{
send_string("7 ");
gg=1;
}
//send_string("7 ");
MSDELAY(10);
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S7.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'1' && n2==0)
{
MSDELAY(1200);
Send2Gsm(char2);
MSDELAY(50);
Send2Gsm("Detected Sensor S7.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'2' && n3==0)
{
MSDELAY(1200);
Send2Gsm(char3);
MSDELAY(50);
Send2Gsm("Detected Sensor S7.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'3' && n4==0)
{
MSDELAY(1200);
Send2Gsm(char4);
MSDELAY(50);
Send2Gsm("Detected Sensor S7.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'4' && n5==0)
{
MSDELAY(1200);
Send2Gsm(char5);
MSDELAY(50);
Send2Gsm("Detected Sensor S7.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
if(repeat==1)
{
repeat=0;
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S7.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
}
}
l=1;
}
```

```
if(sensor8==1 && n==0)
{
//alarm=1;
if(n==0)
{
if(siren==0)
{
alarm=1;
}
if(hh==0)
{
send_string("8 ");
hh=1;
}
//send_string("8 ");
MSDELAY(10);
repeat=0;
if(count!=0)
{
if(n1==0)
{
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S8.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'1' && n2==0)
{
MSDELAY(1200);
Send2Gsm(char2);
MSDELAY(50);
Send2Gsm("Detected Sensor S8.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'2' && n3==0)
{
MSDELAY(1200);
Send2Gsm(char3);
MSDELAY(50);
Send2Gsm("Detected Sensor S8.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'3' && n4==0)
{
MSDELAY(1200);
Send2Gsm(char4);
MSDELAY(50);
Send2Gsm("Detected Sensor S8.\r\n");
MSDELAY(20);
serial(0x1A);
}
if(count>'4' && n5==0)
{
MSDELAY(1200);
Send2Gsm(char5);
MSDELAY(50);
Send2Gsm("Detected Sensor S8.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
if(repeat==1)
{
repeat=0;
Send2Gsm(char1);
MSDELAY(50);
Send2Gsm("Detected Sensor S8.\r\n");
MSDELAY(20);
serial(0x1A);
MSDELAY(500);
}
}
}
n=1;
}
```

```
//if(P1!=0x00)
//{
if(siren==1)
{

for(i=0;i<150;i++)
{
MSDELAY(100);
if(sensor1==1 && a==0)
{
send_string("1 ");
a=1;
}
if(sensor2==1 && b==0)
{
send_string("2 ");
b=1;
}
if(sensor3==1 && c==0)
{
send_string("3 ");
c=1;
}
if(sensor4==1 && dd==0)
{
send_string("4 ");
dd=1;
}
if(sensor5==1 && e==0)
{
send_string("5 ");
e=1;
}
if(sensor6==1 && f==0)
{
send_string("6 ");
f=1;
}
if(sensor7==1 && gg==0)
{
send_string("7 ");
gg=1;
}
if(sensor8==1 && hh==0)
{
send_string("8 ");
hh=1;
}
}
sirenoff=1;
alarm=1;
siren=0;
siren=0;
}

if(siren==0 && sirenoff==0)
{

for(i=0;i<100;i++)
{
MSDELAY(100);
if(sensor1==1 && a==0)
{
send_string("1 ");
a=1;
}
if(sensor2==1 && b==0)
{
send_string("2 ");
b=1;
}
if(sensor3==1 && c==0)
{
send_string("3 ");
c=1;
}
if(sensor4==1 && dd==0)
{
send_string("4 ");
dd=1;
}
if(sensor5==1 && e==0)
{
send_string("5 ");
e=1;
}
if(sensor6==1 && f==0)
{
send_string("6 ");
f=1;
}
if(sensor7==1 && gg==0)
{
send_string("7 ");
gg=1;
}
if(sensor8==1 && hh==0)
{
send_string("8 ");
hh=1;
}
}

alarm=0;
siren=1;
siren=1;
}

//}
}
}
}
}

void SMScon()
{
s1=0;
s2=0;
s3=0;
s4=0;
s5=0;
s6=0;
s7=0;
auth=0;
z=0;
MSDELAY(300);
charr[0]=' ';
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("AT+CMGR=1\r\n");
while(charr[0]!='G' && charr[0]!='K')
{
charr[0]=getCharacter();
z=z+1;
if(z>100)
break;
}
if(charr[0]=='G')
{
for(i=0;i<85;i++)
{
charr[i]=getCharacter();
}
MSDELAY(100);
RI=0;
i=0;
j=0;
while(charr[i]!='+')
{
i=i+1;
if(i>75)
{
break;
}
}
i=i-1;
for(j=0;j<15;j++)
{
autnum[j]=charr[i];
i=i+1;
```

```
}

for(i=0;i<15;i++)
{
if(autnum[i]!=autnum1[i])
auth=1;
}

if(auth==0)
{
i=0;
j=0;
while((charr[i]!='"') || (j!=8))
{
i=i+1;
if(charr[i]=='"')
{
j=j+1;
}
if(i>100)
{
break;
}
}

if(i<100)
{
i=i+3;
for(j=0;j<5;j++)
{
msg[j]=charr[i];
i=i+1;
}

for(k=0;k<5;k++)
{
if(msg[k]!=msg1[k])
{
s1=1;
}
}
for(k=0;k<5;k++)
{
if(msg[k]!=msg2[k])
{
s2=1;
}
}
for(k=0;k<5;k++)
{
if(msg[k]!=msg3[k])
{
s3=1;
}
}
for(k=0;k<5;k++)
{
if(msg[k]!=msg4[k])
{
s4=1;
}
}
for(k=0;k<5;k++)
{
if(msg[k]!=msg5[k])
{
s5=1;
}
}
for(k=0;k<5;k++)
{
if(msg[k]!=msg6[k])
{
s6=1;
}
}
for(k=0;k<5;k++)
{
if(msg[k]!=msg7[k])
{
```

```
s7=1;
}
}

if(s1==0)
{

}

if(s2==0)
{
if(charr[i]=='x')
{
n2=1;
Send2Gsm("AT+CPBW=2\r\n");
MSDELAY(100);
Send2Gsm(char1);
MSDELAY(100);
for(j=12;j<22;j++)
{
serial(char2[j]);
char2[j]=' ';
}
Send2Gsm("\r\n");
Send2Gsm("is deleted from location
2.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
z=0;
MSDELAY(20);
}
else
{
n2=0;
for(j=14;j<24;j++)
{
Num[j]=charr[i];
char2[j-2]=charr[i];
New[j-14]=charr[i];
i=i+1;
}
Num[8]=Num[39]='2';
MSDELAY(100);
Send2Gsm(Num);
Send2Gsm("\r\n");
MSDELAY(100);
Send2Gsm(char2);
MSDELAY(100);
Send2Gsm("Your Number is added to
Security System.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
z=0;
MSDELAY(100);
Send2Gsm(char1);
MSDELAY(100);
Send2Gsm(New);
Send2Gsm("\r\n");
MSDELAY(100);
Send2Gsm("is added to Security System as
a 2nd Receiver.\r\n");
MSDELAY(100);
```

```
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
z=0;
MSDELAY(100);
if(count<'2')
{
count='2';
Send2Gsm("AT+CPBW=6,\"+91222222222
2\",129,\"RECEIVER6\"");
Send2Gsm("\r\n");
MSDELAY(50);
}
}
}

if(s3==0)
{
if(charr[i]=='x')
{
n3=1;
Send2Gsm("AT+CPBW=3\r\n");
MSDELAY(100);
Send2Gsm(char1);
MSDELAY(100);
for(j=12;j<22;j++)
{
serial(char3[j]);
char3[j]=' ';
}
Send2Gsm("\r\n");
Send2Gsm("is deleted from location
3.\r\n");
```

```
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
z=0;
MSDELAY(20);
}
else
{
n3=0;
for(j=14;j<24;j++)
{
Num[j]=charr[i];
char3[j-2]=charr[i];
New[j-14]=charr[i];
i=i+1;
}
Num[8]=Num[39]='3';
MSDELAY(100);
Send2Gsm(Num);
Send2Gsm("\r\n");
MSDELAY(100);
Send2Gsm(char3);
MSDELAY(100);
Send2Gsm("Your Number is added to
Security System.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
MSDELAY(100);
Send2Gsm(char1);
MSDELAY(100);
Send2Gsm(New);
Send2Gsm("\r\n");
MSDELAY(100);
Send2Gsm("is added to Security System as
a 3rd Receiver.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
z=0;
MSDELAY(100);
if(count<'3')
{
count='3';
Send2Gsm("AT+CPBW=6,\"+91333333333
3\",129,\"RECEIVER6\"");
Send2Gsm("\r\n");
MSDELAY(50);
}
}
}

if(s4==0)
{
if(charr[i]=='x')
{
```

```
n4=1;
Send2Gsm("AT+CPBW=4\r\n");
MSDELAY(100);
Send2Gsm(char1);
MSDELAY(100);
for(j=12;j<22;j++)
{
serial(char4[j]);
char4[j]=' ';
}
Send2Gsm("\r\n");
Send2Gsm("is deleted from location
4.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
z=0;
MSDELAY(20);
}
else
{
n4=0;
for(j=14;j<24;j++)
{
Num[j]=charr[i];
char4[j-2]=charr[i];
New[j-14]=charr[i];
i=i+1;
}
Num[8]=Num[39]='4';
MSDELAY(100);
Send2Gsm(Num);
```

```
Send2Gsm("\r\n");
MSDELAY(100);
Send2Gsm(char4);
MSDELAY(100);
Send2Gsm("Your Number is added to
Security System.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
MSDELAY(100);
Send2Gsm(char1);
MSDELAY(100);
Send2Gsm(New);
Send2Gsm("\r\n");
MSDELAY(100);
Send2Gsm("is added to Security System as
a 4th Receiver.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
z=0;
MSDELAY(100);
if(count<'4')
{
count='4';
Send2Gsm("AT+CPBW=6,\"+91444444444
4\",129,\"RECEIVER6\"");
Send2Gsm("\r\n");
MSDELAY(50);
}
}
}

if(s5==0)
{
if(charr[i]=='x')
{
n5=1;
Send2Gsm("AT+CPBW=5\r\n");
MSDELAY(100);
Send2Gsm(char1);
MSDELAY(100);
for(j=12;j<22;j++)
{
serial(char5[j]);
char5[j]=' ';
}
Send2Gsm("\r\n");
Send2Gsm("is deleted from location
5.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
z=0;
MSDELAY(20);
}
```

```
else
{
n5=0;
for(j=14;j<24;j++)
{
Num[j]=charr[i];
char5[j-2]=charr[i];
New[j-14]=charr[i];
i=i+1;
}
Num[8]=Num[39]='5';
MSDELAY(100);
Send2Gsm(Num);
Send2Gsm("\r\n");
MSDELAY(100);
Send2Gsm(char5);
MSDELAY(100);
Send2Gsm("Your Number is added to Security System.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
MSDELAY(100);
Send2Gsm(char1);
MSDELAY(100);
Send2Gsm(New);
Send2Gsm("\r\n");
MSDELAY(100);
Send2Gsm("is added to Security System as a 5th Receiver.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
z=0;
MSDELAY(100);
if(count<'5')
{
count='5';
Send2Gsm("AT+CPBW=6,\"+915555555555\",129,\"RECEIVER6\"");
Send2Gsm("\r\n");
MSDELAY(50);
}
}
}

if(s6==0)
{
Send2Gsm(char1);
MSDELAY(100);
Send2Gsm("1. ");
if(n1==0 && count>'0')
{
for(i=12;i<22;i++)
{
serial(char1[i]);
}
}
Send2Gsm("\r\n");
Send2Gsm("2. ");
if(n2==0 && count>'1')
{
for(i=12;i<22;i++)
{
```

```
serial(char2[i]);
}
}
Send2Gsm("\r\n");
Send2Gsm("3. ");
if(n3==0 && count>'2')
{
for(i=12;i<22;i++)
{
serial(char3[i]);
}
}
Send2Gsm("\r\n");
Send2Gsm("4. ");
if(n4==0 && count>'3')
{
for(i=12;i<22;i++)
{
serial(char4[i]);
}
}
Send2Gsm("\r\n");
Send2Gsm("5. ");
if(n5==0 && count>'4')
{
for(i=12;i<22;i++)
{
serial(char5[i]);
}
}
Send2Gsm("\r\n");
MSDELAY(200);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
MSDELAY(100);
}

if(s7==0)
{
Send2Gsm(char1);
MSDELAY(100);
Send2Gsm("System is working fine.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
chr1[0]=' ';
MSDELAY(100);
}

else if(s1==1 && s2==1 && s3==1 &&
s4==1 && s5==1 && s6==1)
{
Send2Gsm(char1);
MSDELAY(100);
Send2Gsm("Wrong Command.\r\n");
MSDELAY(100);
serial(0x1A);
z=0;
while(chr1[0]!='K' && chr1[0]!='R')
{
chr1[0]=getCharacter();
z=z+1;
if(z>300)
break;
}
```

```
chr1[0]=' ';
MSDELAY(100);
}
}
}
MSDELAY(100);
Send2Gsm("AT\r\n");
MSDELAY(1000);
Send2Gsm("AT+CMGD=1\r\n");
MSDELAY(100);
RI=0;
}
}

void LCDCOMMAND(unsigned char
value)
{
unsigned char temp;
temp = value;
temp&=0xf0;
DATA&=0x0f;
DATA|=temp;
rs=0;
//rw=0;
en=1;
MSDELAY(2);
en=0;
temp=value<<4;
temp&=0xf0;
DATA&=0x0f;
DATA|=temp;
en=1;
MSDELAY(2);
en=0;
}

void LCDDATA(unsigned char value)
{
unsigned char temp;
temp = value;
temp&=0xf0;
DATA&=0x0f;
DATA|=temp;
rs=1;
//rw=0;
en=1;
MSDELAY(2);
en=0;
temp=value<<4;
temp&=0xf0;
DATA&=0x0f;
DATA|=temp;
en=1;
MSDELAY(2);
en=0;                    // Give pulse on E pin
}

void LCDINIT(void)
{
LCDINIT_Write(0x30);    // initialization in
5X7 matrix ,8 data lines ,16X2 mode
MSDELAY(1);
LCDINIT_Write(0x30);    // initialization in
5X7 matrix ,8 data lines ,16X2 mode
MSDELAY(1);
LCDINIT_Write(0x30);    // initialization in
5X7 matrix ,8 data lines ,16X2 mode
MSDELAY(1);
LCDINIT_Write(0x20);    // initialization in
5X7 matrix ,8 data lines ,16X2 mode
MSDELAY(1);
LCDCOMMAND(0x28);    // initialization
in 5X7 matrix ,8 data lines ,16X2 mode
MSDELAY(1);
LCDCOMMAND(0x0E);    //display on
cursor off
MSDELAY(1);
LCDCOMMAND(0x01);    //clear LCD
MSDELAY(1);
```

```
LCDCOMMAND(0x80);     //line 1,position 1
MSDELAY(1);
LCDCOMMAND(0x0C);     //curser off
MSDELAY(1);
}

void LCDINIT_Write(unsigned char a)
{
rs=0;
//rw=0;
DATA=a;
en=1;
MSDELAY(1);
en=0;
}

void send_string(unsigned char *var)
{
while(*var)              //till string ends
LCDDATA(*var++);  //send characters one by one
MSDELAY(1);
}

void SERIALINIT()
{
TMOD=0x22;          // Mode=2
TH1=0xfd;          // 9600 baud
SCON=0x50;        // Serial mode=1 ,8-Bit data,1 Stop bit ,1 Start bit  , Receiving on
TH0=0x00;
TL0=0x00;
TR1=1;          // Start timer
}

void GSMINIT()
{
Send2Gsm("AT\r\n");
MSDELAY(100);
Send2Gsm("ATE0\r\n");
MSDELAY(100);
Send2Gsm("AT+CMGF=1\r\n");
MSDELAY(100);
Send2Gsm("AT+CNMI=1,1,0,0,0\r\n");
MSDELAY(100);
Send2Gsm("AT+CMGD=1\r\n");
MSDELAY(100);
}

void serial(unsigned char x)
{
SBUF=x;
while(TI==0);
TI=0;
}

void Send2Gsm(char *aaa)
{
unsigned int i;
for(i=0;aaa[i]!=0;i++)
{
serial(aaa[i]);
}
}

void MSDELAY(unsigned int item)
{
unsigned int i,j;
for(i=0;i<item;i++)
{
for(j=0;j<600;j++)
{
}
}
}

char getCharacter (void)
{
char chr;
```

```
unsigned long c=0;
while (RI != 1)
{
c=c+1;
if(c>1500)
break;
}
chr = SBUF;
RI = 0;
return(chr);
}
```

DATASHEETS

P89V51RD2

8-bit 80C51 5 V low power 64 kB Flash microcontroller with 1 kB RAM

Rev. 01 — 01 March 2004 Product data

1. General description

The P89V51RD2 is an 80C51 microcontroller with 64 kB Flash and 1024 bytes of data RAM.

A key feature of the P89V51RD2 is its X2 mode option. The design engineer can choose to run the application with the conventional 80C51 clock rate (12 clocks per machine cycle) or select the X2 mode (6 clocks per machine cycle) to achieve twice the throughput at the same clock frequency. Another way to benefit from this feature is to keep the same performance by reducing the clock frequency by half, thus dramatically reducing the EMI.

The Flash program memory supports both parallel programming and in serial In-System Programming (ISP). Parallel programming mode offers gang-programming at high speed, reducing programming costs and time to market. ISP allows a device to be reprogrammed in the end product under software control. The capability to field/update the application firmware makes a wide range of applications possible.

The P89V51RD2 is also In-Application Programmable (IAP), allowing the Flash program memory to be reconfigured even while the application is running.

2. Features

- 80C51 Central Processing Unit
- 5 V Operating voltage from 0 to 40 MHz
- 64 kB of on-chip Flash program memory with ISP (In-System Programming) and IAP (In-Application Programming)
- Supports 12-clock (default) or 6-clock mode selection via software or ISP
- SPI (Serial Peripheral Interface) and enhanced UART
- PCA (Programmable Counter Array) with PWM and Capture/Compare functions
- Four 8-bit I/O ports with three high-current Port 1 pins (16 mA each)
- Three 16-bit timers/counters
- Programmable Watchdog timer (WDT)
- Eight interrupt sources with four priority levels
- Second DPTR register
- Low EMI mode (ALE inhibit)
- TTL- and CMOS-compatible logic levels

4. Block diagram

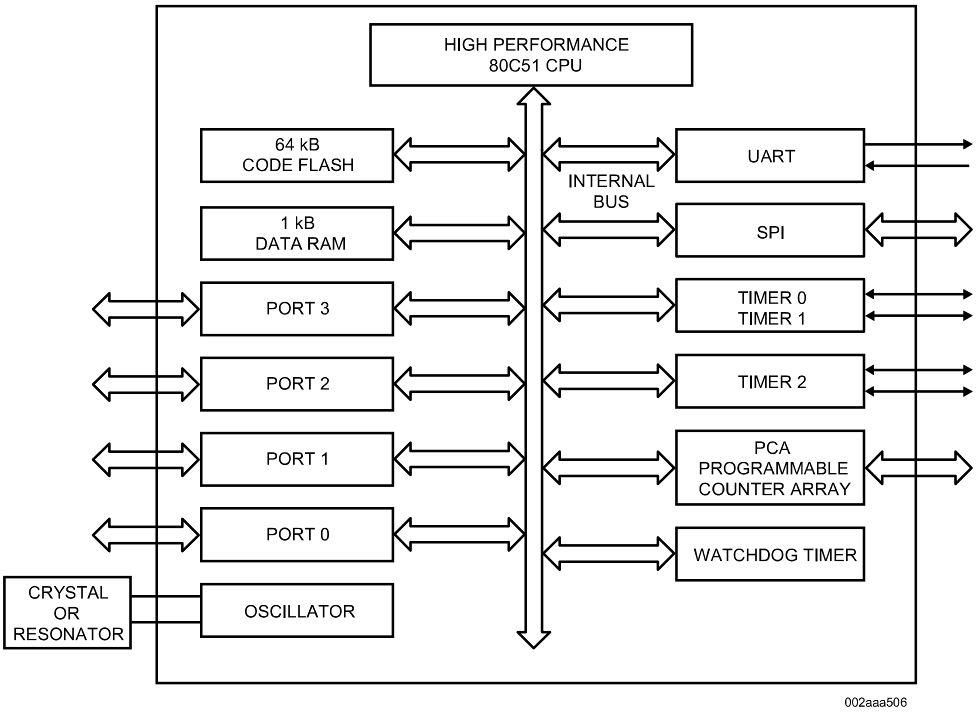


Fig 1. P89V51RD2 block diagram.

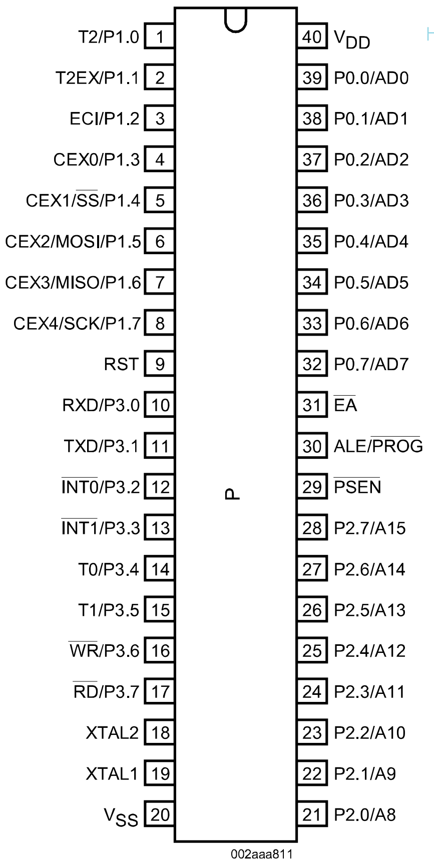


Fig 3. PDIP40 pin configuration.

5.2 Pin description

Table 3: P89V51RD2 pin description

Symbol	Pin			Type	Description
	DIP40	TQFP44	PLCC44		
P0.0 to P0.7	39-32	37-30	43-36	I/O	Port 0: Port 0 is an 8-bit open drain bi-directional I/O port. Port 0 pins that have in this state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external code and data memory. In this application, it uses strong internal pull-ups when transitioning to the code bytes during the external host mode programming, and outputs the code bytes during the external host mode verification. External pull-ups are required during program verification or as a general purpose I/O port.
P1.0 to P1.7	1-8	40-44, 1-3	2-9	I/O with internal pull-up	Port 1: Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 1 pins are pulled high by the internal pull-ups when be used as inputs in this state. As inputs, Port 1 pins that are externally pulled LOW will source current (I_{IL}) because of the internal pull-ups. P1.5, P1.6, P1.7 have high current drive of 16 mA. Port 1 also receives the low-order address bytes during the external host mode programming and verification.
P1.0	1	40	2	I/O	T2: External count input to Timer/Counter 2 or Clock-out from Timer/Counter 2
P1.1	2	41	3	I	T2EX: Timer/Counter 2 capture/reload trigger and direction control
P1.2	3	42	4	I	ECI: External clock input. This signal is the external clock input for the PCA.
P1.3	4	43	5	I/O	CEX0: Capture/compare external I/O for PCA Module 0. Each capture/compare module connects to a Port 1 pin for external I/O. When not used by the PCA, this pin can handle standard I/O.
P1.4	5	44	6	I/O	~~SS~~: Slave port select input for SPI CEX1: Capture/compare external I/O for PCA Module 1
P1.5	6	1	7	I/O	MOSI: Master Output Slave Input for SPI CEX2: Capture/compare external I/O for PCA Module 2
P1.6	7	2	8	I/O	MISO: Master Input Slave Output for SPI CEX3: Capture/compare external I/O for PCA Module 3
P1.7	8	3	9	I/O	SCK: Master Output Slave Input for SPI CEX4: Capture/compare external I/O for PCA Module 4

Table 3: P89V51RD2 pin description...continued

Symbol	Pin			Type	Description
	DIP40	TQFP44	PLCC44		
P2.0 to P2.7	21-28	18-25	24-31	I/O with internal pull-up	Port 2: Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. Port 2 pins are pulled HIGH by the internal pull-ups when be used as inputs in this state. As inputs, Port 2 pins that are externally pulled LOW will source current (I_{IL}) because of the internal pull-ups. Port 2 sends the high-order address byte during fetches from external program memory and during accesses to external Data Memory that use 16-bit address (MOVX@DPTR). In this application, it uses strong internal pull-ups when transitioning to signals and a partial of high-order address bits during the external host mode programming and verification.
P3.0 to P3.7	10-17	5, 7-13	11, 13-19	I/O with internal pull-up	Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins are pulled HIGH by the internal pull-ups when be used as inputs in this state. As inputs, Port 3 pins that are externally pulled LOW will source current (I_{IL}) because of the internal pull-ups. Port 3 also receives some control signals and a partial of high-order address bits during the external host mode programming and verification.
P3.0	10	5	11	I	RXD: serial input port
P3.1	11	7	13	O	TXD: serial output port
P3.2	12	8	14	I	$\overline{\text{INT0}}$: external interrupt 0 input
P3.3	13	9	15	I	$\overline{\text{INT1}}$: external interrupt 1 input
P3.4	14	10	16	I	T0: external count input to Timer/Counter 0
P3.5	15	11	17	I	T1: external count input to Timer/Counter 1
P3.6	16	12	18	O	$\overline{\text{WR}}$: external data memory write strobe
P3.7	17	13	19	O	$\overline{\text{RD}}$: external data memory read strobe
$\overline{\text{PSEN}}$	29	26	32	I/O	Program Store Enable: $\overline{\text{PSEN}}$ is the read strobe for external program memory. When the device is executing from internal program memory, $\overline{\text{PSEN}}$ is inactive (HIGH). When the device 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. A forced HIGH-to-LOW input transition on the $\overline{\text{PSEN}}$ pin while the RST input is continually held HIGH for more than 10 machine cycles will cause the device to enter external host mode programming.
RST	9	4	10	I	Reset: While the oscillator is running, a HIGH logic state on this pin for two machine cycles will reset the device. If the $\overline{\text{PSEN}}$ pin is driven by a HIGH-to-LOW input transition while the RST input pin is held HIGH, the device will enter the external host mode, otherwise the device will enter the normal operation mode.

Table 3: P89V51RD2 pin description...continued

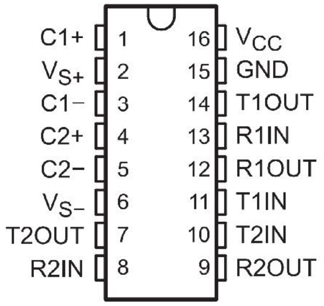
Symbol	Pin			Type	Description
	DIP40	TQFP44	PLCC44		
~~EA~~	31	29	35	I	External Access Enable: ~~EA~~ must be connected to V $_{\text{SS}}$ in order to enable the device to fetch code from the external program memory. $\overline{\text{EA}}$ must be strapped to V_{DD} for internal program execution. However, Security lock level 4 will disable $\overline{\text{EA}}$, and program execution is only possible from internal program memory. The ~~EA~~ pin can tolerate a high voltage of 12 V.
ALE/ $\overline{\text{PROG}}$	30	27	33	I/O	Address Latch Enable: ALE is the output signal for latching the low byte of the address during an access to external memory. This pin is also the programming pulse input ($\overline{\text{PROG}}$) for flash programming. Normally the ALE[1] is emitted at a constant rate of $^1/_6$ the crystal frequency[2] and can be used for external timing and clocking. One ALE pulse is skipped during each access to external data memory. However, if AO is set to ALE is disabled.
NC	-	6, 17, 28, 39	1, 12, 23, 34	I/O	No Connect
XTAL1	19	15	21	I	Crystal 1: Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	14	20	O	Crystal 2: Output from the inverting oscillator amplifier.
V_{DD}	40	38	44	I	Power supply
V_{SS}	20	16	22	I	Ground

[1] ALE loading issue: When ALE pin experiences higher loading (>30 pF) during the reset, the microcontroller may accidentally enter into modes other than normal working mode. The solution is to add a pull-up resistor of 3 kΩ to 50 kΩ to V_{DD}, e.g., for ALE pin.

[2] For 6-clock mode, ALE is emitted at $^1/_3$ of crystal frequency.

- **Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28**
- **Operates From a Single 5-V Power Supply With 1.0-μF Charge-Pump Capacitors**
- **Operates Up To 120 kbit/s**
- **Two Drivers and Two Receivers**
- **±30-V Input Levels**
- **Low Supply Current . . . 8 mA Typical**
- **ESD Protection Exceeds JESD 22**
  - **2000-V Human-Body Model (A114-A)**
- **Upgrade With Improved ESD (15-kV HBM) and 0.1-μF Charge-Pump Capacitors is Available With the MAX202**
- **Applications**
  - **TIA/EIA-232-F, Battery-Powered Systems, Terminals, Modems, and Computers**

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube of 25	MAX232N	MAX232N
	SOIC (D)	Tube of 40	MAX232D	MAX232
		Reel of 2500	MAX232DR	
	SOIC (DW)	Tube of 40	MAX232DW	MAX232
		Reel of 2000	MAX232DWR	
	SOP (NS)	Reel of 2000	MAX232NSR	MAX232
−40°C to 85°C	PDIP (N)	Tube of 25	MAX232IN	MAX232IN
	SOIC (D)	Tube of 40	MAX232ID	MAX232I
		Reel of 2500	MAX232IDR	
	SOIC (DW)	Tube of 40	MAX232IDW	MAX232I
		Reel of 2000	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

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.

LinASIC is a trademark of Texas Instruments.

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

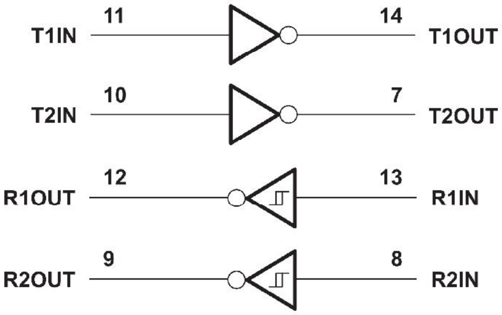
H = high level, L = low level

EACH RECEIVER

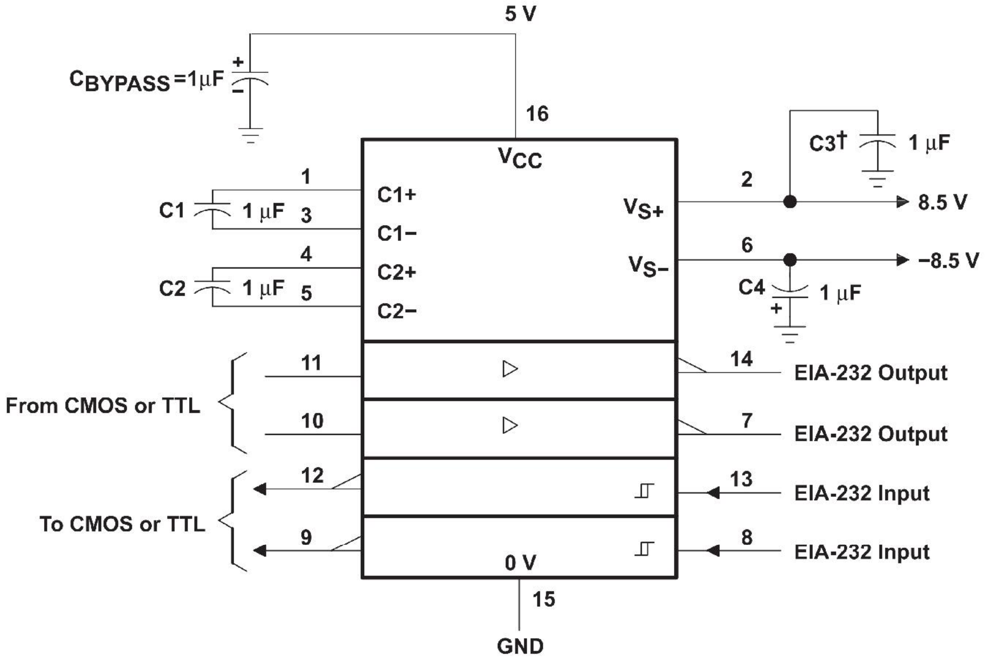
INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

logic diagram (positive logic)



APPLICATION INFORMATION



† C3 can be connected to V_{CC} or GND.

NOTES: A. Resistor values shown are nominal.

B. Nonpolarized ceramic capacitors are acceptable. If polarized tantalum or electrolytic capacitors are used, they should be connected as shown. In addition to the 1-µF capacitors shown, the MAX202 can operate with 0.1-µF capacitors.

Figure 4. Typical Operating Circuit

FAIRCHILD
SEMICONDUCTOR®

August 2013

LM78XX / LM78XXA
3-Terminal 1 A Positive Voltage Regulator

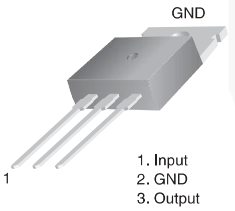
Features

- Output Current up to 1 A
- Output Voltages: 5, 6, 8, 9, 10, 12, 15, 18, 24 V
- Thermal Overload Protection
- Short-Circuit Protection
- Output Transistor Safe Operating Area Protection

Description

The LM78XX series of three-terminal positive regulators is available in the TO-220 package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut-down, and safe operating area protection. If adequate heat sinking is provided, they can deliver over 1 A output current. Although designed primarily as fixed-voltage regulators, these devices can be used with external components for adjustable voltages and currents.

TO-220 (Single Gauge)



Ordering Information[1]

Product Number	Output Voltage Tolerance	Package	Operating Temperature	Packing Method
LM7805CT	±4%	TO-220 (Single Gauge)	-40°C to +125°C	Rail
LM7806CT				
LM7808CT				
LM7809CT				
LM7810CT				
LM7812CT				
LM7815CT				
LM7818CT				
LM7824CT				
LM7805ACT	±2%		0°C to +125°C	
LM7809ACT				
LM7810ACT				
LM7812ACT				
LM7815ACT				

Note:

1. Above output voltage tolerance is available at 25°C.

Block Diagram

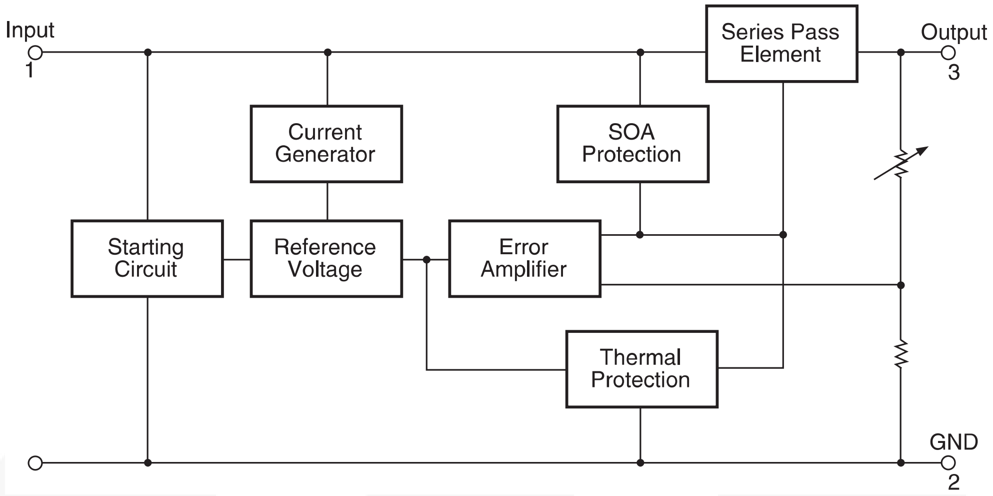


Figure 1. Block Diagram

Absolute Maximum Ratings

Stresses exceeding the absolute maximum ratings may damage the device. The device may not function or be operable above the recommended operating conditions and stressing the parts to these levels is not recommended. In addition, extended exposure to stresses above the recommended operating conditions may affect device reliability. The absolute maximum ratings are stress ratings only. Values are at T_A = 25°C unless otherwise noted.

Symbol	Parameter		Value	Unit
V_I	Input Voltage	V_O = 5 V to 18 V	35	V
		V_O = 24 V	40	
$R_{\theta JC}$	Thermal Resistance, Junction-Case (TO-220)		5	°C/W
$R_{\theta JA}$	Thermal Resistance, Junction-Air (TO-220)		65	°C/W
T_{OPR}	Operating Temperature Range	LM78xx	-40 to +125	°C
		LM78xxA	0 to +125	
T_{STG}	Storage Temperature Range		- 65 to +150	°C

SIM900
GSM/GPRS Module

The SIM900 is a complete Quad-band GSM/GPRS solution in a SMT module which can be embedded in the customer applications.

Featuring an industry-standard interface, the SIM900 delivers GSM/GPRS 850/900/1800/1900MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. With a tiny configuration of 24mm x 24mm x 3 mm, SIM900 can fit almost all the space requirements in your M2M application, especially for slim and compact demand of design.

- SIM900 is designed with a very powerful single-chip processor integrating AMR926EJ-S core
- Quad - band GSM/GPRS module with a size of 24mmx24mmx3mm
- SMT type suit for customer application
- An embedded Powerful TCP/IP protocol stack
- Based upon mature and field-proven platform, backed up by our support service, from definition to design and production

General featrues

- Quad-Band 850/ 900/ 1800/ 1900 MHz
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Compliant to GSM phase 2/2+
  – Class 4 (2 W @850/ 900 MHz)
  – Class 1 (1 W @ 1800/1900MHz)
- Dimensions: 24* 24 * 3 mm
- Weight: 3.4g
- Control via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- SIM application toolkit
- Supply voltage range 3.4 ... 4.5 V
- Low power consumption
- Operation temperature:
  -30 °C to +80 °C

Specifications for fax

- Group 3, class 1

Specifications for data

- GPRS class 10: max. 85.6 kbps (downlink)
- PBCCH support
- Coding schemes CS 1, 2, 3, 4
- CSD up to 14.4 kbps
- USSD
- Non transparent mode
- PPP-stack

Specifications for SMS via GSM / GPRS

- Point-to-point MO and MT
- SMS cell broadcast
- Text and PDU mode

Drivers

- MUX Driver

Specifications for voice

- Tricodec
  – Half rate (HR)
  – Full rate (FR)
  – Enhanced Full rate (EFR)
- Hands-free operation
  （Echo suppression）
- AMR
  Half Rate(HR)
  Full Rate(FR)

Interfaces

- Interface to external SIM 3V/ 1.8V
- analog audio interface
- RTC backup
- SPI interface
- Serial interface
- Antenna pad
- I2C
- GPIO
- PWM
- ADC

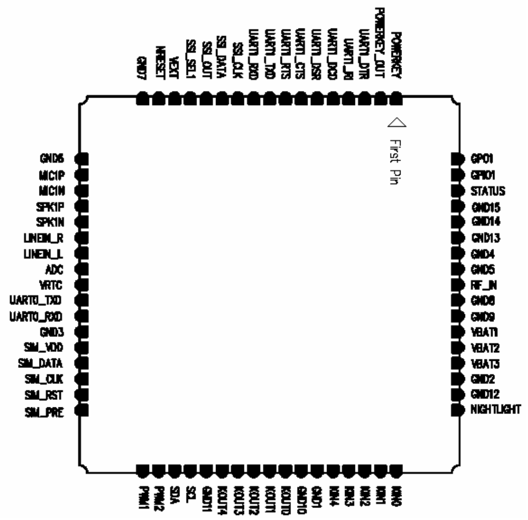
Compatibility

- AT cellular command interface

Approvals（in planning）

- CE
- FCC
- ROHS
- PTCRB
- GCF
- AT&T
- IC
- TA

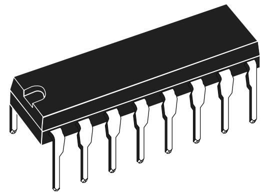
Pin Assignment



ULN2001A-ULN2002A
ULN2003A-ULN2004A

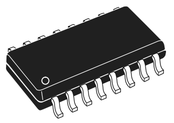
SEVEN DARLINGTON ARRAYS

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



DIP16

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



SO16

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

DESCRIPTION

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

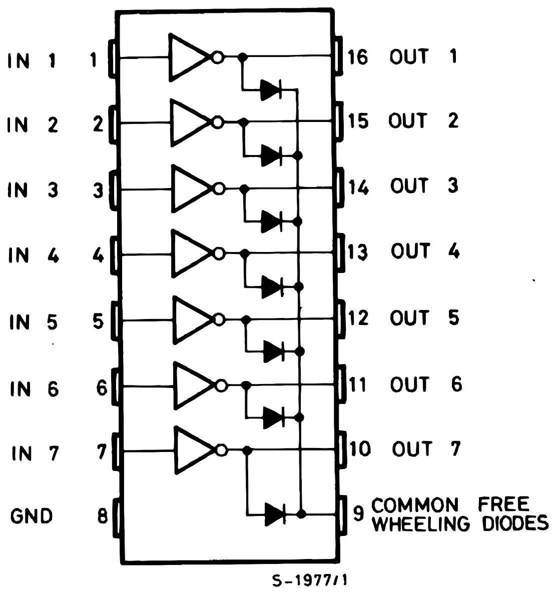
The four versions interface to all common logic families :

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

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

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

PIN CONNECTION



P89V51RD2

8-bit 80C51 5 V low power 64 kB Flash microcontroller with 1 kB RAM

Rev. 01 — 01 March 2004 Product data

1. General description

The P89V51RD2 is an 80C51 microcontroller with 64 kB Flash and 1024 bytes of data RAM.

A key feature of the P89V51RD2 is its X2 mode option. The design engineer can choose to run the application with the conventional 80C51 clock rate (12 clocks per machine cycle) or select the X2 mode (6 clocks per machine cycle) to achieve twice the throughput at the same clock frequency. Another way to benefit from this feature is to keep the same performance by reducing the clock frequency by half, thus dramatically reducing the EMI.

The Flash program memory supports both parallel programming and in serial In-System Programming (ISP). Parallel programming mode offers gang-programming at high speed, reducing programming costs and time to market. ISP allows a device to be reprogrammed in the end product under software control. The capability to field/update the application firmware makes a wide range of applications possible.

The P89V51RD2 is also In-Application Programmable (IAP), allowing the Flash program memory to be reconfigured even while the application is running.

2. Features

- 80C51 Central Processing Unit
- 5 V Operating voltage from 0 to 40 MHz
- 64 kB of on-chip Flash program memory with ISP (In-System Programming) and IAP (In-Application Programming)
- Supports 12-clock (default) or 6-clock mode selection via software or ISP
- SPI (Serial Peripheral Interface) and enhanced UART
- PCA (Programmable Counter Array) with PWM and Capture/Compare functions
- Four 8-bit I/O ports with three high-current Port 1 pins (16 mA each)
- Three 16-bit timers/counters
- Programmable Watchdog timer (WDT)
- Eight interrupt sources with four priority levels
- Second DPTR register
- Low EMI mode (ALE inhibit)
- TTL- and CMOS-compatible logic levels

4. Block diagram

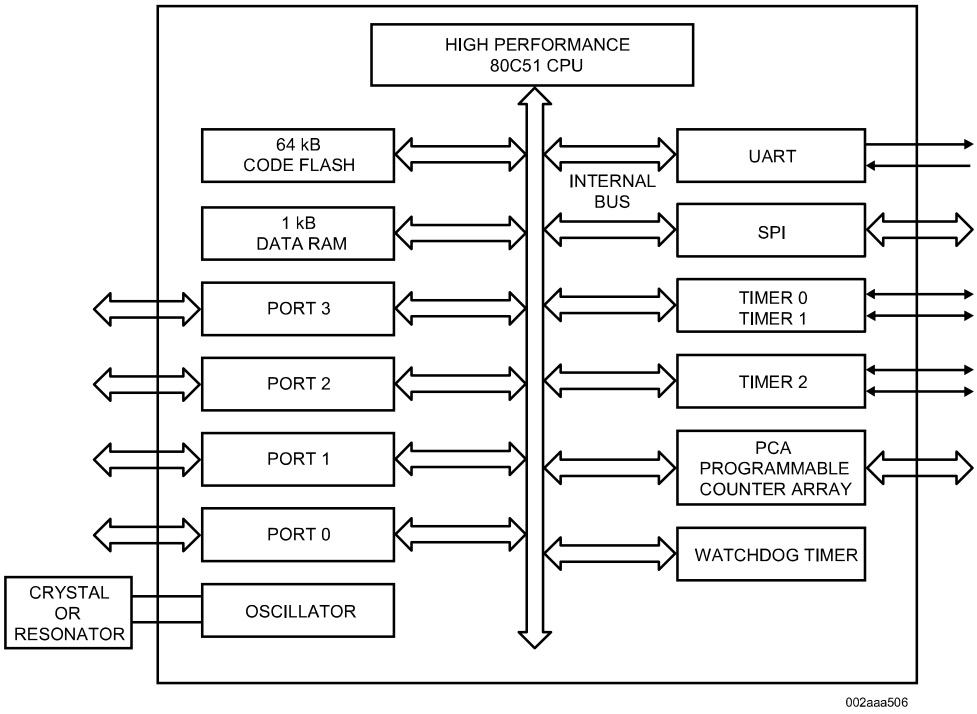


Fig 1. P89V51RD2 block diagram.

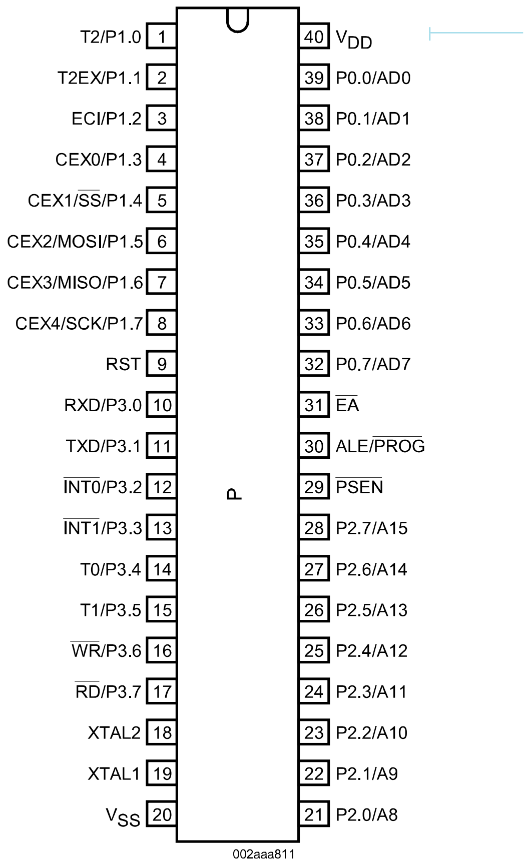


Fig 3. PDIP40 pin configuration.

5.2 Pin description

Table 3: P89V51RD2 pin description

Symbol	Pin			Type	Description
	DIP40	TQFP44	PLCC44		
P0.0 to P0.7	39-32	37-30	43-36	I/O	Port 0: Port 0 is an 8-bit open drain bi-directional I/O port. Port 0 pins that have in this state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external code and data memory. In this application, it uses strong internal pull-ups when transitioning to the code bytes during the external host mode programming, and outputs the code bytes during the external host mode verification. External pull-ups are required during program verification or as a general purpose I/O port.
P1.0 to P1.7	1-8	40-44, 1-3	2-9	I/O with internal pull-up	Port 1: Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 1 pins are pulled high by the internal pull-ups when be used as inputs in this state. As inputs, Port 1 pins that are externally pulled LOW will source current (I_{IL}) because of the internal pull-ups. P1.5, P1.6, P1.7 have high current drive of 16 mA. Port 1 also receives the low-order address bytes during the external host mode programming and verification.
P1.0	1	40	2	I/O	T2: External count input to Timer/Counter 2 or Clock-out from Timer/Counter 2
P1.1	2	41	3	I	T2EX: Timer/Counter 2 capture/reload trigger and direction control
P1.2	3	42	4	I	ECI: External clock input. This signal is the external clock input for the PCA.
P1.3	4	43	5	I/O	CEX0: Capture/compare external I/O for PCA Module 0. Each capture/compare module connects to a Port 1 pin for external I/O. When not used by the PCA, this pin can handle standard I/O.
P1.4	5	44	6	I/O	~~SS~~: Slave port select input for SPI CEX1: Capture/compare external I/O for PCA Module 1
P1.5	6	1	7	I/O	MOSI: Master Output Slave Input for SPI CEX2: Capture/compare external I/O for PCA Module 2
P1.6	7	2	8	I/O	MISO: Master Input Slave Output for SPI CEX3: Capture/compare external I/O for PCA Module 3
P1.7	8	3	9	I/O	SCK: Master Output Slave Input for SPI CEX4: Capture/compare external I/O for PCA Module 4

Table 3: P89V51RD2 pin description...continued

Symbol	Pin			Type	Description
	DIP40	TQFP44	PLCC44		
P2.0 to P2.7	21-28	18-25	24-31	I/O with internal pull-up	Port 2: Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. Port 2 pins are pulled HIGH by the internal pull-ups when be used as inputs in this state. As inputs, Port 2 pins that are externally pulled LOW will source current (I_{IL}) because of the internal pull-ups. Port 2 sends the high-order address byte during fetches from external program memory and during accesses to external Data Memory that use 16-bit address (MOVX@DPTR). In this application, it uses strong internal pull-ups when transitioning to signals and a partial of high-order address bits during the external host mode programming and verification.
P3.0 to P3.7	10-17	5, 7-13	11, 13-19	I/O with internal pull-up	Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. Port 3 pins are pulled HIGH by the internal pull-ups when be used as inputs in this state. As inputs, Port 3 pins that are externally pulled LOW will source current (I_{IL}) because of the internal pull-ups. Port 3 also receives some control signals and a partial of high-order address bits during the external host mode programming and verification.
P3.0	10	5	11	I	RXD: serial input port
P3.1	11	7	13	O	TXD: serial output port
P3.2	12	8	14	I	$\overline{\text{INT0}}$: external interrupt 0 input
P3.3	13	9	15	I	$\overline{\text{INT1}}$: external interrupt 1 input
P3.4	14	10	16	I	T0: external count input to Timer/Counter 0
P3.5	15	11	17	I	T1: external count input to Timer/Counter 1
P3.6	16	12	18	O	$\overline{\text{WR}}$: external data memory write strobe
P3.7	17	13	19	O	$\overline{\text{RD}}$: external data memory read strobe
$\overline{\text{PSEN}}$	29	26	32	I/O	Program Store Enable: $\overline{\text{PSEN}}$ is the read strobe for external program memory. When the device is executing from internal program memory, $\overline{\text{PSEN}}$ is inactive (HIGH). When the device 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. A forced HIGH-to-LOW input transition on the $\overline{\text{PSEN}}$ pin while the RST input is continually held HIGH for more than 10 machine cycles will cause the device to enter external host mode programming.
RST	9	4	10	I	Reset: While the oscillator is running, a HIGH logic state on this pin for two machine cycles will reset the device. If the $\overline{\text{PSEN}}$ pin is driven by a HIGH-to-LOW input transition while the RST input pin is held HIGH, the device will enter the external host mode, otherwise the device will enter the normal operation mode.

Table 3: P89V51RD2 pin description...continued

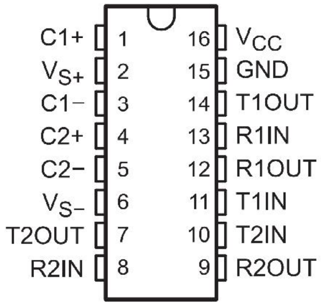
Symbol	Pin			Type	Description
	DIP40	TQFP44	PLCC44		
~~EA~~	31	29	35	I	External Access Enable: ~~EA~~ must be connected to V_{SS} in order to enable the device to fetch code from the external program memory. $\overline{EA}$ must be strapped to V_{DD} for internal program execution. However, Security lock level 4 will disable $\overline{EA}$, and program execution is only possible from internal program memory. The ~~EA~~ pin can tolerate a high voltage of 12 V.
ALE/ $\overline{PROG}$	30	27	33	I/O	Address Latch Enable: ALE is the output signal for latching the low byte of the address during an access to external memory. This pin is also the programming pulse input ($\overline{PROG}$) for flash programming. Normally the ALE[1] is emitted at a constant rate of $^1/_6$ the crystal frequency[2] and can be used for external timing and clocking. One ALE pulse is skipped during each access to external data memory. However, if AO is set to ALE is disabled.
NC	-	6, 17, 28, 39	1, 12, 23, 34	I/O	No Connect
XTAL1	19	15	21	I	Crystal 1: Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTAL2	18	14	20	O	Crystal 2: Output from the inverting oscillator amplifier.
V_{DD}	40	38	44	I	Power supply
V_{SS}	20	16	22	I	Ground

[1] ALE loading issue: When ALE pin experiences higher loading (>30 pF) during the reset, the microcontroller may accidentally enter into modes other than normal working mode. The solution is to add a pull-up resistor of 3 kΩ to 50 kΩ to V_{DD}, e.g., for ALE pin.

[2] For 6-clock mode, ALE is emitted at $^1/_3$ of crystal frequency.

- **Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28**
- **Operates From a Single 5-V Power Supply With 1.0-μF Charge-Pump Capacitors**
- **Operates Up To 120 kbit/s**
- **Two Drivers and Two Receivers**
- **±30-V Input Levels**
- **Low Supply Current . . . 8 mA Typical**
- **ESD Protection Exceeds JESD 22**
  - **2000-V Human-Body Model (A114-A)**
- **Upgrade With Improved ESD (15-kV HBM) and 0.1-μF Charge-Pump Capacitors is Available With the MAX202**
- **Applications**
  - **TIA/EIA-232-F, Battery-Powered Systems, Terminals, Modems, and Computers**

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube of 25	MAX232N	MAX232N
	SOIC (D)	Tube of 40	MAX232D	MAX232
		Reel of 2500	MAX232DR	
	SOIC (DW)	Tube of 40	MAX232DW	MAX232
		Reel of 2000	MAX232DWR	
	SOP (NS)	Reel of 2000	MAX232NSR	MAX232
−40°C to 85°C	PDIP (N)	Tube of 25	MAX232IN	MAX232IN
	SOIC (D)	Tube of 40	MAX232ID	MAX232I
		Reel of 2500	MAX232IDR	
	SOIC (DW)	Tube of 40	MAX232IDW	MAX232I
		Reel of 2000	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

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.

LinASIC is a trademark of Texas Instruments.

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

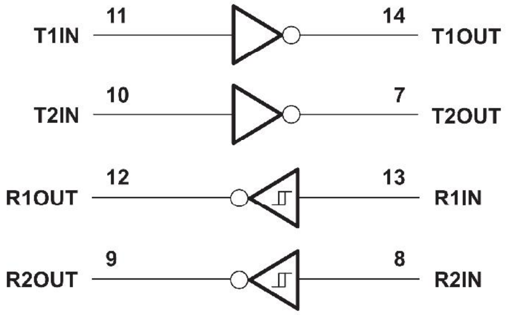
H = high level, L = low level

EACH RECEIVER

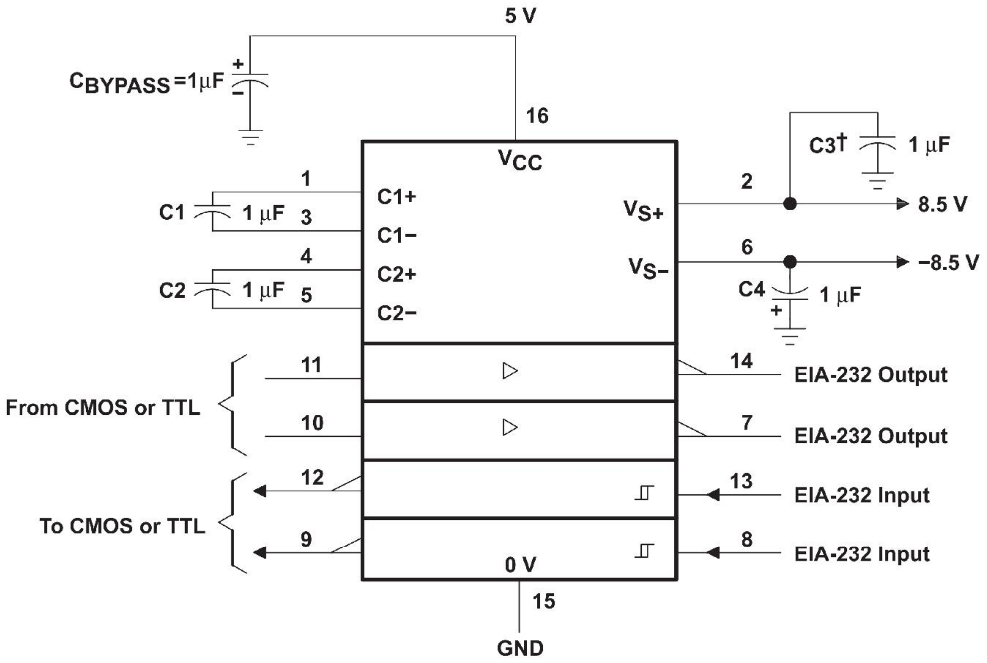
INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

logic diagram (positive logic)



APPLICATION INFORMATION



† C3 can be connected to V_{CC} or GND.

NOTES: A. Resistor values shown are nominal.

B. Nonpolarized ceramic capacitors are acceptable. If polarized tantalum or electrolytic capacitors are used, they should be connected as shown. In addition to the 1-µF capacitors shown, the MAX202 can operate with 0.1-µF capacitors.

Figure 4. Typical Operating Circuit

August 2013

LM78XX / LM78XXA
3-Terminal 1 A Positive Voltage Regulator

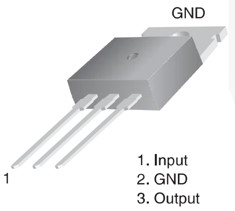
Features

- Output Current up to 1 A
- Output Voltages: 5, 6, 8, 9, 10, 12, 15, 18, 24 V
- Thermal Overload Protection
- Short-Circuit Protection
- Output Transistor Safe Operating Area Protection

Description

The LM78XX series of three-terminal positive regulators is available in the TO-220 package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut-down, and safe operating area protection. If adequate heat sinking is provided, they can deliver over 1 A output current. Although designed primarily as fixed-voltage regulators, these devices can be used with external components for adjustable voltages and currents.

TO-220 (Single Gauge)



Ordering Information[1]

Product Number	Output Voltage Tolerance	Package	Operating Temperature	Packing Method
LM7805CT	±4%	TO-220 (Single Gauge)	-40°C to +125°C	Rail
LM7806CT				
LM7808CT				
LM7809CT				
LM7810CT				
LM7812CT				
LM7815CT				
LM7818CT				
LM7824CT				
LM7805ACT	±2%		0°C to +125°C	
LM7809ACT				
LM7810ACT				
LM7812ACT				
LM7815ACT				

Note:

1. Above output voltage tolerance is available at 25°C.

Block Diagram

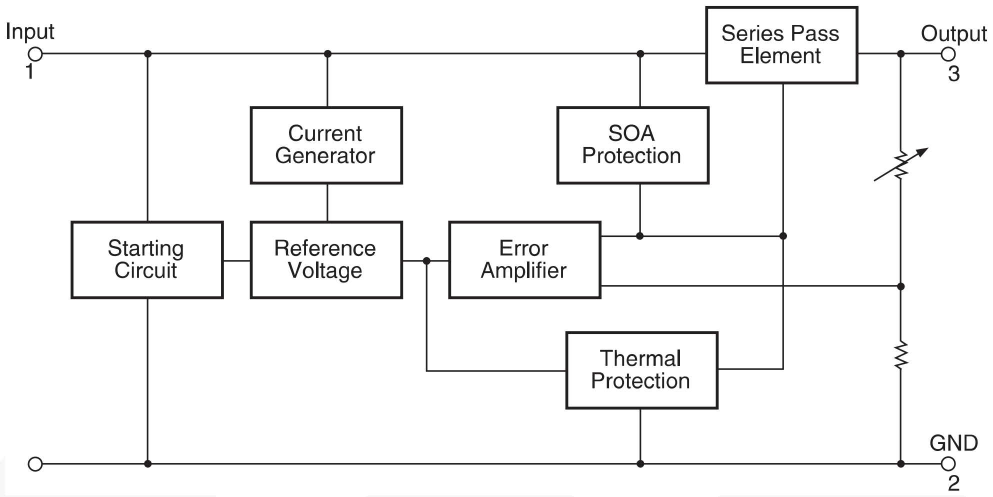


Figure 1. Block Diagram

Absolute Maximum Ratings

Stresses exceeding the absolute maximum ratings may damage the device. The device may not function or be operable above the recommended operating conditions and stressing the parts to these levels is not recommended. In addition, extended exposure to stresses above the recommended operating conditions may affect device reliability. The absolute maximum ratings are stress ratings only. Values are at T_A = 25°C unless otherwise noted.

Symbol	Parameter		Value	Unit
V_I	Input Voltage	V_O = 5 V to 18 V	35	V
		V_O = 24 V	40	
$R_{\theta JC}$	Thermal Resistance, Junction-Case (TO-220)		5	°C/W
$R_{\theta JA}$	Thermal Resistance, Junction-Air (TO-220)		65	°C/W
T_{OPR}	Operating Temperature Range	LM78xx	-40 to +125	°C
		LM78xxA	0 to +125	
T_{STG}	Storage Temperature Range		- 65 to +150	°C

SIM900
GSM/GPRS Module

The SIM900 is a complete Quad-band GSM/GPRS solution in a SMT module which can be embedded in the customer applications.

Featuring an industry-standard interface, the SIM900 delivers GSM/GPRS 850/900/1800/1900MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. With a tiny configuration of 24mm x 24mm x 3 mm, SIM900 can fit almost all the space requirements in your M2M application, especially for slim and compact demand of design.

- SIM900 is designed with a very powerful single-chip processor integrating AMR926EJ-S core
- Quad - band GSM/GPRS module with a size of 24mmx24mmx3mm
- SMT type suit for customer application
- An embedded Powerful TCP/IP protocol stack
- Based upon mature and field-proven platform, backed up by our support service, from definition to design and production

General featrues

- Quad-Band 850/ 900/ 1800/ 1900 MHz
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Compliant to GSM phase 2/2+
  – Class 4 (2 W @850/ 900 MHz)
  – Class 1 (1 W @ 1800/1900MHz)
- Dimensions: 24* 24 * 3 mm
- Weight: 3.4g
- Control via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- SIM application toolkit
- Supply voltage range 3.4 ... 4.5 V
- Low power consumption
- Operation temperature:
  -30 °C to +80 °C

Specifications for fax

- Group 3, class 1

Specifications for data

- GPRS class 10: max. 85.6 kbps (downlink)
- PBCCH support
- Coding schemes CS 1, 2, 3, 4
- CSD up to 14.4 kbps
- USSD
- Non transparent mode
- PPP-stack

Specifications for SMS via GSM / GPRS

- Point-to-point MO and MT
- SMS cell broadcast
- Text and PDU mode

Drivers

- MUX Driver

Specifications for voice

- Tricodec
  – Half rate (HR)
  – Full rate (FR)
  – Enhanced Full rate (EFR)
- Hands-free operation
  （Echo suppression）
- AMR
  Half Rate(HR)
  Full Rate(FR)

Interfaces

- Interface to external SIM 3V/ 1.8V
- analog audio interface
- RTC backup
- SPI interface
- Serial interface
- Antenna pad
- I2C
- GPIO
- PWM
- ADC

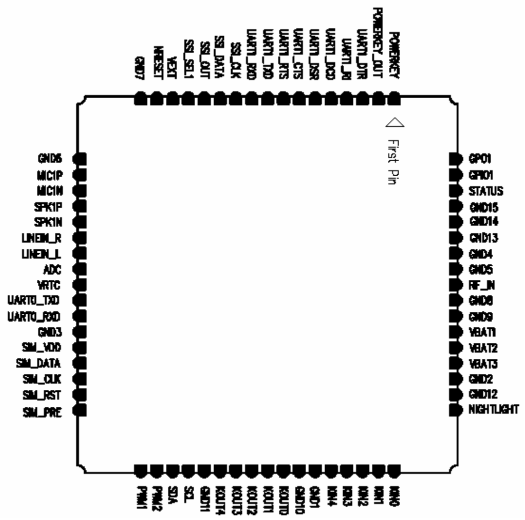
Compatibility

- AT cellular command interface

Approvals（in planning）

- CE
- FCC
- ROHS
- PTCRB
- GCF
- AT&T
- IC
- TA

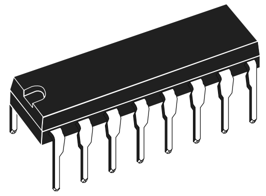
Pin Assignment



ULN2001A-ULN2002A
ULN2003A-ULN2004A

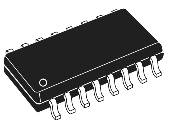
SEVEN DARLINGTON ARRAYS

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



DIP16

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



SO16

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

DESCRIPTION

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

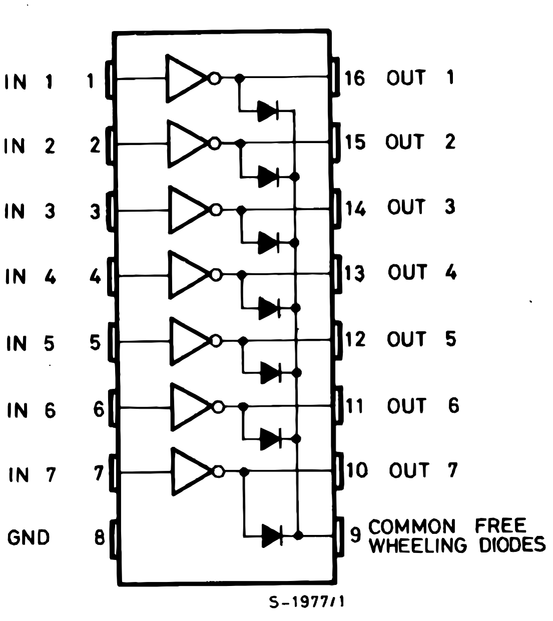
The four versions interface to all common logic families :

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

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

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

PIN CONNECTION


IN 1 1
16 OUT 1
IN 2 2
15 OUT 2
IN 3 3
14 OUT 3
IN 4 4
13 OUT 4
IN 5 5
12 OUT 5
IN 6 6
11 OUT 6
IN 7 7
10 OUT 7
GND 8
9 COMMON FREE WHEELING DIODES
S-1977/1