REMOTE MONITORING AND CONTROL THROUGH SMS

By

GC JAWAD YAQUB

GC SAAD ABBASI

CAPT AAQIB NASEER

GC AHSAN ASHFAQ

PROJECT SUPERVISOR: GP.CAPT® MUZAFFAR ALI

Dissertation submitted for partial fulfilment of requirement of MCS/NUST for award of the B.E. degree in Telecommunication Engineering.

Department of Electrical Engineering
Military College of Signals
Rawalpindi
APRIL 2006

ABSTRACT

REMOTE MONITORING AND CONTROL SYSTEM USING SMS

By

GC JAWAD YAQUB

GC SAAD ABBASI

CAPT AAQIB NASEER

GC AHSAN ASHFAQ

Mobile communication is becoming more and more popular day by day. Not only the uses are over growing, but rates are also getting cheaper. GSM technology not only allows voice communication but data services like SMS, MMS, and GPRS are also its salient features. Most mobile users love to send and receive SMS. It is not only a way of having fun; rather it can be useful in many 'monitoring and control applications. You have a system at your home or office, which can understand your SMS and act accordingly (switch your PC off, turn off lights etc) can be an exciting stuff. All you need is a GSM terminal capable of receiving your SMS and a microcontroller host card, to execute your command. We will implement in this project a demonstration of this remote control system.

LIST OF FIGURES

LIST OF TABLES

1.1 Brief Project Concept and Description

As the name of project *"REMOTE MONITORING AND CONTROL THROUGH SMS"* implies, the project aims at fulfilling the requirements of control and monitoring remotely through a SMS enabled GSM device.

The control task to be performed can be as simple as switching a light bulb; or as complicated as monitoring the premises status at the specified events and reporting back to the user. Since there is no limit of how far the user is (GSM network allows you to communicate with host card even if you are on international roaming). We call this as 'remote control through SMS'. We first have to send the correct code and the requested task will be performed.

1.2 Block Diagram

The concept is illustrated by following diagram:-

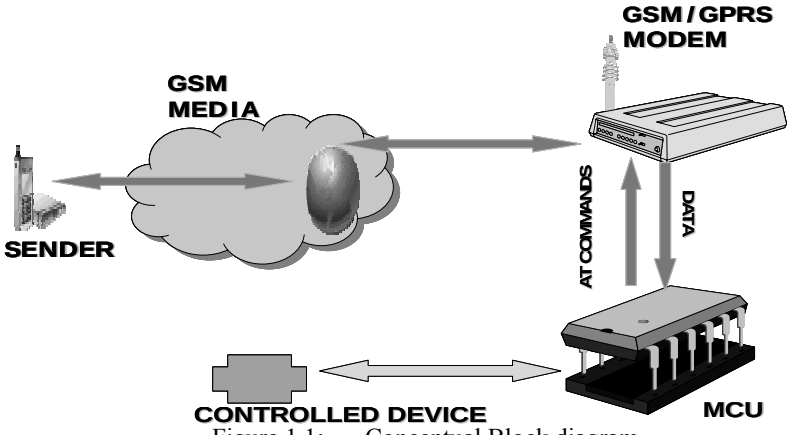


Figure 1.1: Conceptual Block diagram

1.3 Strategy to carry out project

Since our project demanded a careful implementation of embedded control system, we decided to proceed with task as following:-

1. Identification of main components. This included the major components recognition/ selection as the concept dictated .e.g. a modem and microcontroller. Later it included small components required to complete relay and interface circuitry, between major components.

2. Interfacing various components. This required a comprehensive knowledge of internal working of major components. E.g. knowing that every modem is controlled by special commands, called AT commands, was crucial to interface our GSM modem to microcontroller. Similarly knowledge of programming microcontroller. Hardwiring various components in complete circuit form required understanding the pin configuration of various IC's and their voltage levels, as well as communication protocols like RS-232 followed by them.

3. Final adjustments as the need arises, once main framework has been decided upon.

1.3.1 **Identification of Main Components**:-

We identified following main components to meet the requirement of project:-

1. A GSM device (a mobile station)-to originate an SMS based command

2. A GSM terminal modem – a GSM terminal (modem) will be needed, capable of receiving (& sending) an SMS text message. This GSM terminal will communicate with the host micro-controller board

3. An intelligent microcontroller – to interpret the command received and guide the necessary sequence of action

4. An interface and relay circuitry- to implement the commands generated by microcontroller, through peripheral devices.

1.4 **Main Components Used**:-

1. Siemens MC-35 terminal modem
2. Atmel AT 89C52 microcontroller chip.
3. Interface and relay circuitry:
    a. DS-232A-N Line driver IC
    b. DS-1302 RTC chip
    c. ULN-2003
    d. AT 24C1024
    e. Hitachi LCD

2.1 GSM Modem Siemens MC-35[1]

2.1.1 Modem description

Figure 2.1: MC-35 Modem

Siemens MC-35 GSM modem offers the advantage of being equipped with both GPRS and dual-band GSM .The MC35 allows a PC or other device to control a GSM phone via standard and extended AT commands. (For detailed Electrical characteristics, refer to Appendix A: GSM Modem MC-35 Characteristics)

2.1.2 MC-35 Features

Table 2.1 MC-35 Modem features

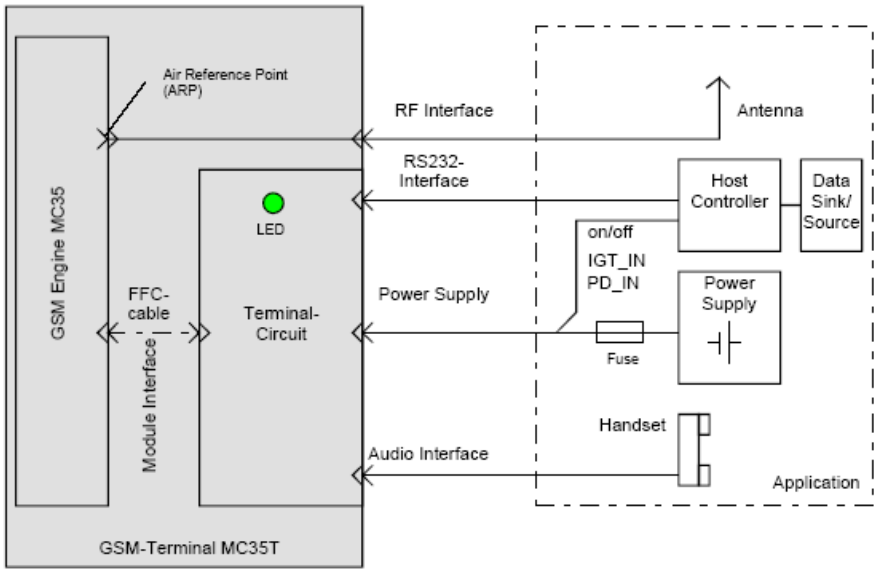
Feature	**Implementation**
Transmission	Voice, data, SMS, fax
Power supply	Single supply voltage 8V to 30V
Frequency bands	Dual Band GSM900 and GSM1800
Transmit power	Class 4 (2W) for GSM900 Class 1 (1W) for GSM1800
SIM card reader	internal
External antenna	Connected via antenna FME connector
Current consumption @12V	TALK mode (peak) 1,2A (pulsed 577ms at T=4,615ms) TALK mode GSM: 170mA (typical average) DATA mode GPRS: 180mA (typical average) IDLE mode: 35mA (typical) SLEEP mode: 30mA (typical) Power Down mode: 700µA (typical)
Audio interface	Analog (microphone, earpiece)
Serial interface	RS-232 (CMOS level) bi-directional bus for commands / data using AT commands
Selectable baud rate	300bps...115kbps (AT interface)
Autobauding range	Supported baud rates: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps

[1] Source: MC-35 datasheet.pdf available at www.siemens.com/wm

2.1.3 **Block diagram of an MC-35 GSM/GPRS modem based application**[2]

The place and of GSM modem in our project can be conceptually well understood by following block diagram. The various interfaces along with the application part are shown in the next diagram.

Figure 2.2: Block diagram for MC-35 application



2.1.3.1 **MC-35 GSM/GPRS engine**

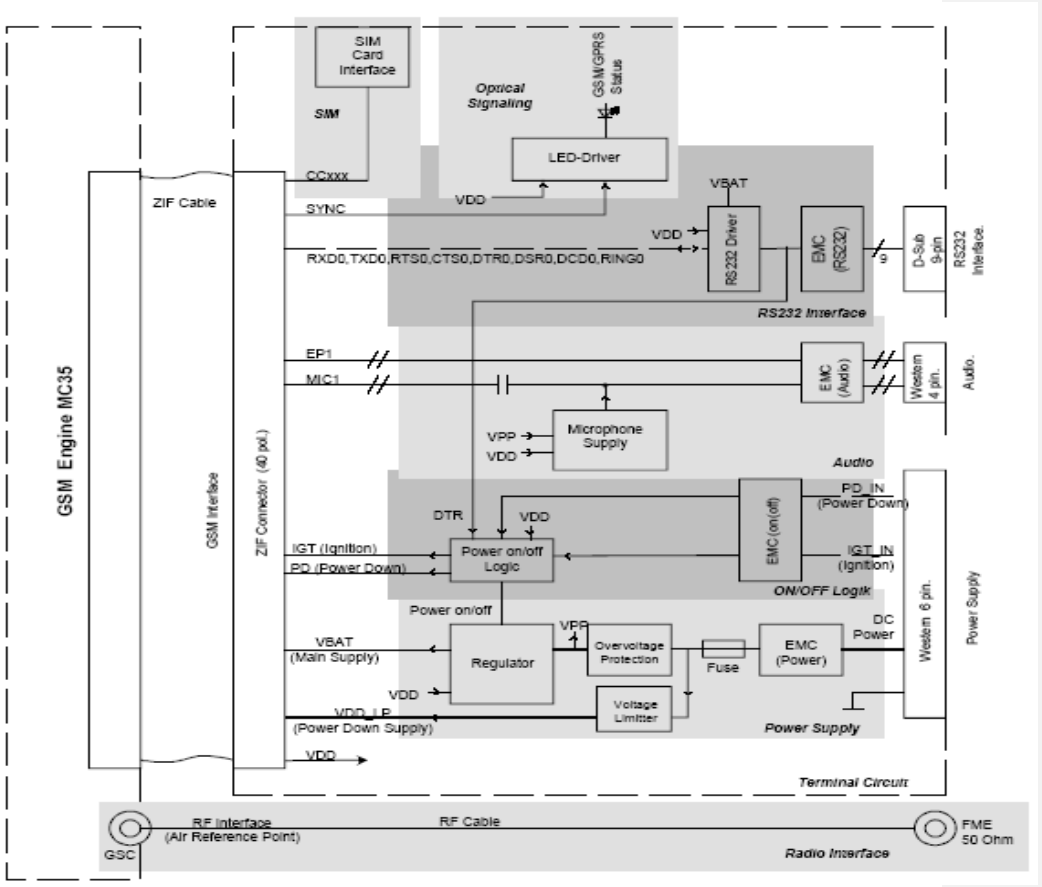
The MC35 GSM/GPRS Engine is a major functional component of the MC35 Terminal that handles all the processing for audio, signal and data within a GSM/GPRS cellular device. Internal software runs the application interface and the whole GSM and GPRS protocol stack. A UART forms the interface to the Terminal Circuit.

[2] Source: hardware interface description MC-35.pdf available at www.siemens.com/wm

2.1.3.2 **Terminal circuit**

The block diagram of the terminal circuit of the modem is shown below. The various interfaces of modem to peripheral devices, along with their description come next.

Figure 2.3: Block diagram of terminal circuit



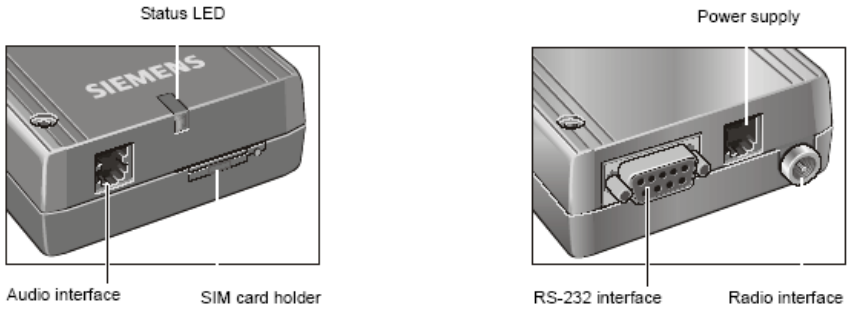
2.1.4 **Modem interfaces and their description**

MC35 Terminal provides the following connectors for power supply, interfacing

and antenna:

- 6-pole Western plug (female) for power supply, ignition, power down signal
- 4-pole Western plug (female) for audio accessory, such as a handset
- 9-pole (female) SUB-D plug for RS-232 serial interface
- FME Jack (male) for antenna (Radio Interface)
- SIM card holder

Figure 2.4: MC-35 interfaces



2.1.4.1 **RS-232 Interface**

Via RS-232 Interface, the host controller controls the MC35 Terminal and transports data.

The details of the pin assignment is given below in the table

Table 2.2: Pin assignment-RS232 interface of MC-35

Pin no.	Signal name	I/O	Function
1	DCD	O	Data Carrier Detected
2	RXD	O	Receive Data
3	TXD	I	Transmit Data
4	DTR	I	Data Terminal Ready Attention: The ignition of MC35 Terminal is activated via a rising edge of high potential (+5 ... +15 V)
5	GND	-	Ground
6	DSR	O	Data Set Ready
7	RTS	I	Request To Send
8	CTS	O	Clear To Send
9	RI	O	Ring Indication

MC35T is designed for use as a DCE. Based on the conventions for DCE-DTE connections, it communicates with the customer application (DTE) using the following signals:

- Port TxD @ application sends data to TXD of MC35T
- Port RxD @ application receives data from RD of MC35T

The RS-232 interface is implemented as a serial asynchronous transmitter and receiver conforming to ITU-T RS-232 Interchange Circuits DCE. It is configured for 8 data bits, no Parity and 1 stop bit, and can be operated at bit rates from 300bps to 115kbps. Auto-bauding supports the following bit rates: 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200 bps. Hardware handshake using the RTS and CTS signals and XON/XOFF software flow control are supported.

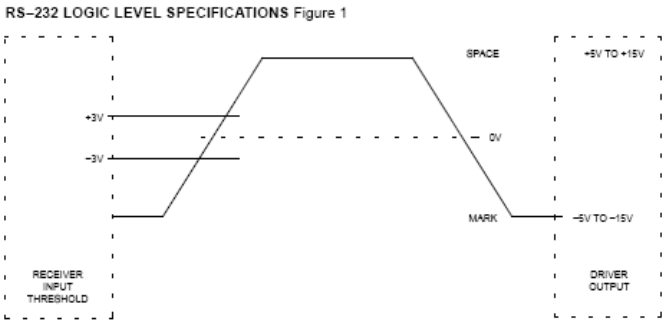
2.1.4.2 **RS-232 communication fundamentals**[3]

RS-232 means recommended standard approved by ITU for serial data communication. RS-232 is an asynchronous (a marching band must be "in sync" with each other so that when one steps they all step. They are asynchronous in that they follow the band leader to keep their timing) communications method.

2.1.4.2.1 **Voltage levels-RS-232**

RS-232 has its own predefined voltage levels, which must be adhered to while communicating data serially. Many devices such as Microcontrollers use CMOS logic levels of 0-5 volts, which must be converted to RS-232 voltage levels using UART device (or simply serial port).A positive voltage is called a MARK and a negative voltage is called a SPACE. Typically, the plc works with +/- 15volts. The voltage between +/- 3 volts is generally not used and is considered noise

Figure 2.5: RS-232 logic level diagram



2.1.4.2.2 **Types of RS-232 devices**

[3] Source: Fundamentals of RS–232 Serial Communications.pdf at Dallas semiconductors website

There are 2 types of RS-232 devices. The first is called a DTE device. This means Data Terminal Equipment and a common example is a computer. The other type is called a DCE device. DCE means Data Communications Equipment and a common example is a modem.

2.1.4.3 **SIM interface**

The SIM interface is intended for 3V SIM.

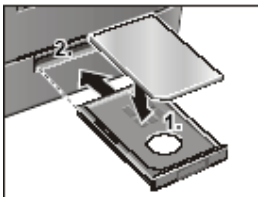
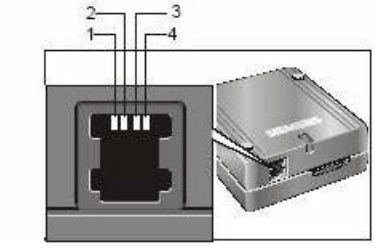


Figure 2.6: SIM interface-MC35

2.1.4.4 **Audio interface**

The audio interface provides an analog input for a microphone and an analog output for an earpiece.



Pin assignment:

1	MICN (Microphone -)
2	EPN (Earpiece)
3	EPP (Earpiece)
4	MICP (Microphone +)

Figure 2.7: Audio interface-MC35

2.1.4.5 **Radio interface**

A 50Ω internal antenna cable adapts the Air Reference Point (ARP) to the FME

(male) connector.

2.1.5 **AT Commands**

The modem isn't dealt with directly in most communications applications. Instead, protocols which abstract the modem are used. The standard and guaranteed protocol for performing basic communication with a modem is the Hayes AT command set, developed originally by Hayes corp. for their Smart modem.

When a modem isn't connected or connecting to another modem, it's in command mode. When a modem is in command mode, the modem can accept commands, in the form of strings written to the serial port to which the modem is attached. When the modem isn't in command mode, it is in online mode. In online mode, everything written to a modem's serial port is sent over the phone line to another modem.

2.1.5.1 **AT command syntax**

The general syntax of the AT commands is given below:

1. The "AT" or "at" prefix must be set at the beginning of each command line.
2. To terminate a command Line enter <CR>.
3. Commands are usually followed by a response that includes "<CR><LF><response><CR><LF>".

2.1.5.2 **AT commands used**

We have used AT commands to control and communicate with siemens MC-35 GSM modem. (Details at Appendix B: AT commands used)

2.2 **8052 Microcontroller (Atmel AT89C52 microcontroller)**[4]

2.2.1 **What is Microcontroller**

[4] 8052 Tutorial & Reference.pdf available at http://www.8052.com

A microcontroller (often abbreviated MCU) is a single computer chip (integrated circuit) that executes a user program, normally for the purpose of controlling some device—hence the name microcontroller.

The program is normally contained either in a second chip, called an EPROM, or within the same chip as the microcontroller itself. A microcontroller is normally found in devices such as microwave ovens, automobiles, keyboards, CD players, etc.

Microcontrollers are used in devices that require some amount of computing power but don't require as much computing power as that provided by a complex (and expensive) 486 or Pentium system which generally requires a large amount of supporting circuitry

Microcontroller-based systems are generally smaller, more reliable, and cheaper. They are ideal for the types of applications described above where cost and unit size are very important considerations.

2.2.1 **What is an 8052**

The 8052 is an 8-bit microcontroller originally developed by Intel in the late 1970s. It included an instruction set of 255 operation codes (op codes), 32 input/output lines, three user-controllable timers, an integrated and automatic serial port, and 256 bytes of on-chip RAM.

The 8052 was designed such that control of the MCU and all input/output between the MCU and external devices is accomplished via Special Function Registers (SFRs).

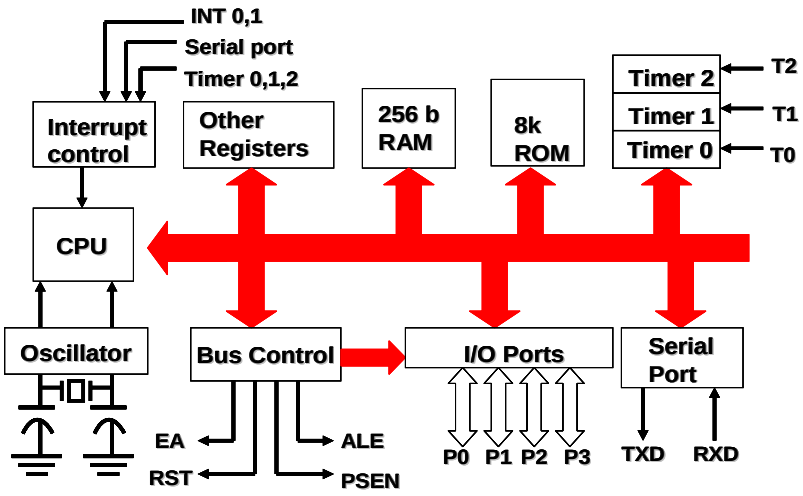


Figure 2.8: 8052 Block diagram

2.2.2 **Features** (For details please refer to Appendix E: Atmel AT89C52 IC)

1. 8kb ROM
2. 256 byte on-chip RAM
3. Four 8-bit I/O ports
4. Three 16 bit timers
5. Serial interface
6. 64 k external ROM
7. 64 k external RAM
8. Boolean processor

9 210 bit address able locations

10 8-bit data bus

11 16-bit address bus

12 34 general purpose registers each of 8 bits

13 Internal and 2 external interrupts.

14 Bit as well as byte addressable RAM area of 16K

2.2.3 **Register Banks**

The 8052 uses 8 "R" registers which are used in many of its instructions. These "R" registers are numbered from 0 through 7 (R0, R1, R2, R3, R4, R5, R6, and R7) and are generally used to assist in manipulating values and moving data from one memory location to another.

2.2.4 **Bit memory**

The 8052 is a communications and control-oriented microcontroller, which often has to deal with on and off situations. Bit memory gives the developer the ability to access a number of bit variables directly with simple instructions to set, clear, and compare these bits. These variables may be either 1 or 0.

2.2.5 **Special function Register memory**

Special Function Registers (SFRs) are areas of memory that control specific functionality of the 8052 MCU. For example, four SFRs permit access to the 8052.s 32 input/output lines (8 lines per SFR). Another SFR allows a program to read or write to the 8052.s serial port. Other SFRs allow the user to set the serial baud rate, control and access timers, and configure the 8052.s interrupt system. There are only 26 SFRs in a standard 8052.

2.2.6 **Memory map**

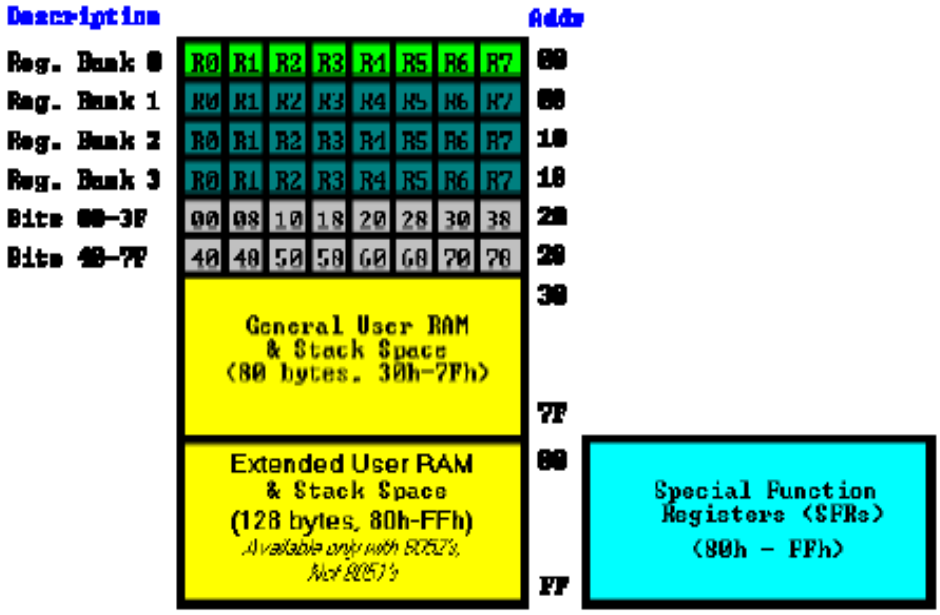


Figure 2.9: Memory map-8052

2.2.7 **Microcontroller Program Storage**

The program for a microcontroller is normally stored on a memory integrated circuit (IC), called an EPROM, or on the microcontroller chip itself. Once you've developed software for a microcontroller it is normally programmed (or .burned.) into an EPROM chip, and that chip is subsequently physically inserted into the circuitry of your hardware. The microcontroller accesses the program stored in the EPROM and executes it.

2.2.8 **Timing**

The 8052 operates with timing derived from an external crystal or a clock.

2.2.9 **Timers**

The 8052 comes equipped with three timers, all of which may be controlled, set, read, and configured individually. The timers have three general functions: 1) Keeping time and/or calculating the amount of time between events, 2) Counting the events themselves, or 3) Generating baud rates for the serial port.

2.2.10 **Serial communication**

One of the 8052.s many powerful features is its integrated UART, otherwise known as a serial port. The fact that the 8052 has an integrated serial port means that you may very easily read and write values to the serial port. We simply need to configure the serial port's operation mode and baud rate. Once configured, all we have to do is write to an SFR to write a value to the serial port or read the same SFR to read a value from the serial port.

2.3 **Minor components details**

2.3.1 **Atmel 24C1024 EEPROM**[5]

[5] Source: AT89C52.pdf available at Atmel semiconductors website

It is an optional module whose general purpose is to save the status of the system. It provides 1048576 bits of serial electrically programmable memory organized as 131072 words of 8 bits each.

2.3.1.1 **Features**

1. Low voltage operation 2.7
2. Internally organized 131072 x 8 bit memory
3. wire serial interface
4. Schmitt trigger filtered inputs for noise suppression
5. Bi directional data transfer protocol
6. 400 kHz and 1 MHz clock rates
7. Write protect key for hardware and software data protection
8. Random and sequential read modes
9. High reliability and self timed write cycle

2.3.2 **DS 1302 Real time clock (The Trickle Charge Time Keeping Chip)**[6]

DS-1302 real time clock has been used in our design whose main purpose was to display the time. Interfacing the DS1302 with a microprocessor is simplified by using synchronous serial communication. Only three wires are required to communicate with the clock/RAM: CE, I/O (data line), and SCLK (serial clock). Data can be transferred to and from the clock/RAM 1 byte at a time or in a burst of up to 31 bytes. The DS1302 is

[6] Source: DS1302 Trickle Charge Timekeeping Chip. pdf available at Dallas semiconductors website

designed to operate on very low power and retain data and clock information on less than 1μW.

2.3.2.1 **Key features**

1. Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year with Leap-Year Compensation Valid Up to 2100
2. 31 x 8 RAM for Scratchpad Data Storage +
3. Serial I/O for Minimum Pin Count
4. 2.0V to 5.5V Full Operation
5. Uses Less than 300nA at 2.0V
6. Single-Byte or Multiple-Byte (Burst Mode) Data Transfer for Read or Write of Clock or RAM Data
7. 8-Pin DIP or Optional 8-Pin SO for Surface Mount
8. Simple 3-Wire Interface

2.3.3 **Max family of Line drivers**[7]

The standard mode MAX families ICs are used basically for the level conversion from CMOS logic level to RS-232 logic level, for interfacing purpose. (For technical details, please refer to Appendix C: Max 232 Line driver IC).

2.3.4 **FLOW CONTROL ULN 2003 IC** [8]

The output from the microcontroller can not be used in order to control a device .For this purpose we required the power regulators Darlington drivers normally termed as the flow control ICs were used in this regard. The ULN2003 is 5V TTL, CMOS

[7] Source: Max 232 line driver IC. Pdf available at Maxim semiconductors website
[8] Source: ULN-2003.pdf available at STMicroelectronics website

high voltage, high current Darlington arrays each containing seven open collector Darlington pairs with common emitters. These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays, filament lamps, thermal print- heads and high power buffers.(For details, please refer to Appendix D: ULN 2003 IC)

2.3.4.1 **Characteristics**

1. Seven Darlington's per package output current 500ma per driver (600ma peak).
2. Output voltage 50v integrated suppression diodes for Inductive loads outputs can be paralleled for higher current TTL/ CMOS/ PMOS /DTL compatible inputs pinned opposite outputs to simplify layout

2.3.5 **Hitachi LCD HD44780**

A liquid crystal display has been interfaced with the microcontroller. It was required to show the status of MODEM as it executes commands given by microcontroller, and resulting responses. LCD requires a code for its proper usage, this code is vendor made and we have not incorporated for this code.

2.3.5.1 **Characteristics**

1. 40 character x 2 lines
2. Built-in LSI HD44780 controller
3. +5volt single power supply

4. Display Colour: Grey

3.1 Hard board design , Component selection

After going through the algorithm phase, successful running of the code in the simulated environment, the next step was to design hardware for the successful implementation of our hypothesis. The steps involved in the hardware design, the details of the various components used in the hardware design have been explained in this chapter of our report.

3.2 Why the board was designed

The main purpose of designing the hardware on a PCB instead of a brad board was to reduce the complexity of the design and to minimize the noise and chattering. We have designed a two-layered board in order to minimize these effects and to make our design more compact and easy to understand.

3.3 Tool used for board design

There were several tools that were available for the board designing and the simulation purposes such as PROTEL, ORCAD, PCAD, PS-PICE .we selected PCAD [9]because of its user-friendly nature, easy to understand user guide and due to its availability in the local markets. We would like to mention here that this tool can be learnt easily by going through its user help manual that is supplied on along with the product on its purchase.

3.4 Steps for PCB design

Several steps were involved in the designing of the PCB using the PCAD

[9] Details for PCAD software are available at www.cadx-pcb.com/PCAD.html

software. The details of the steps involved in the designing have been explained below.

3.4.1 **Schematic entry**

PCAD has got a schematic editor tool, which is used, for designing the schematics. Our intention in this report is not to give the details regarding the schematic editor tool but what we want is that a person reading the report must get familiarized with the steps and the techniques that are involved in the designing of the PCB using PCAD.

In the schematic entry we have to bind together all the components. These components are available in the standard library of the software; we can design those not available like some connectors and the ICs ourselves. The diagram for the schematic entry is given as Figure 3.1 (appended at last of chapter).

3.4.2 **Net list Generation**

PCAD generates the netlist file. This netlist file is basically the output of the schematic tool. The netlist file is used in the second phase of the PCB design. The steps involved in the netlist generation are as follows:

1. Import the netlist file from the schematic tool
2. Footprint the circuit in PCAD PCB
3. Board area determination and outline
4. Placement of the components: The diagram after placing the components on the PCAD PCB is shown as Figure 3.2 (appended at last of chapter)
5. Setting of the net attributes

3.4.3 **Auto-Routing phase**

The GEBRER FILES that are used for the PCB manufacture are generated in this

phase and then the final circuit can be imported from the output for realistic manufacture. The diagrammatical view of the PCB after the auto router phase is shown as Figure 3.3 (appended at last of chapter)

3.5 **Component design**

The details of the various components used in the design along with their interfaces are given below:

1. **Atmel AT89C528-bit Microcontroller** (with 8K Bytes Flash). It is an 8052 Microcontroller Derivative chip, being the heart of the system is the most important and distinct part of our design. It controls the entire system with the relevant commands. The details of several other components interfaced with

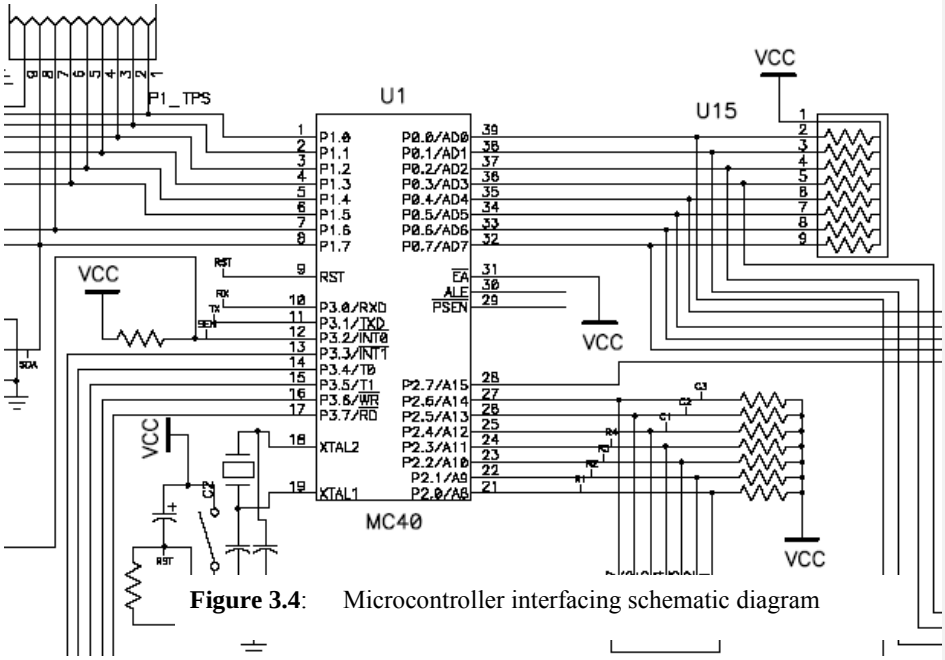


Figure 3.4: Microcontroller interfacing schematic diagram

the microcontroller can be seen from the schematic (Figure 3.4). In the schematic diagram U1 interface shows the microcontroller.

2. MAX 232 Line Drivers

The standard mode MAX family ICs are used basically for the level conversion.RS-232 level conversion for the interfacing with the personal computer is done with the help of this IC. The schematic view of the MAX drivers is given as Figure 3.5.

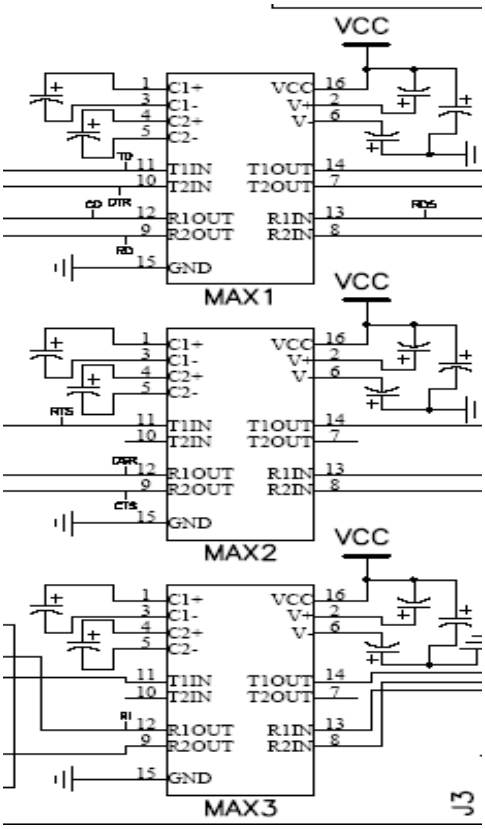


Figure 3.5: Max 232 Line drivers schematic diagram

3. **Miscellaneous Components**. Besides the major components of the design whose details and interfaces are given above, other minor parts and components along with their details are given below:

a. Negative voltage generator. LCD requires a negative voltage generator for the purpose of contrast in the display, for which specifications are provided by HITACHI LCD vendor.

b. Light emitting diodes. LEDs have been placed on the PCB in order to show in which mode the modem is operating. Green and red color LEDs have been used for the purpose.

c. Sensor. A Through beam sensor has been purchased from the market. A transmit and receive connection has been made via infrared and it has been interfaced with the microcontroller through its interrupt pin.

d. Optional ringer circuit. There has been placed an optional circuit on the board that can be used for ringing purposes. The main aim of the circuit was to generate a bell if any intruder interrupts the infrared communication between the transmitter and the receiver module of the sensor.

e. Power Supply Circuit. The diagram for the power supply circuit along with its explanation is given as Figure 3.6. The power is supplied through the 12 volts adapter. The 7805 regulator regulates the input 12 volts and generates the reference voltage. The instruction LEDs will show the status of on and off .DE-coupling

capacitors have been used in order to reduce the noise on the board. The power supply circuit has also got optional connectors for different power usage.

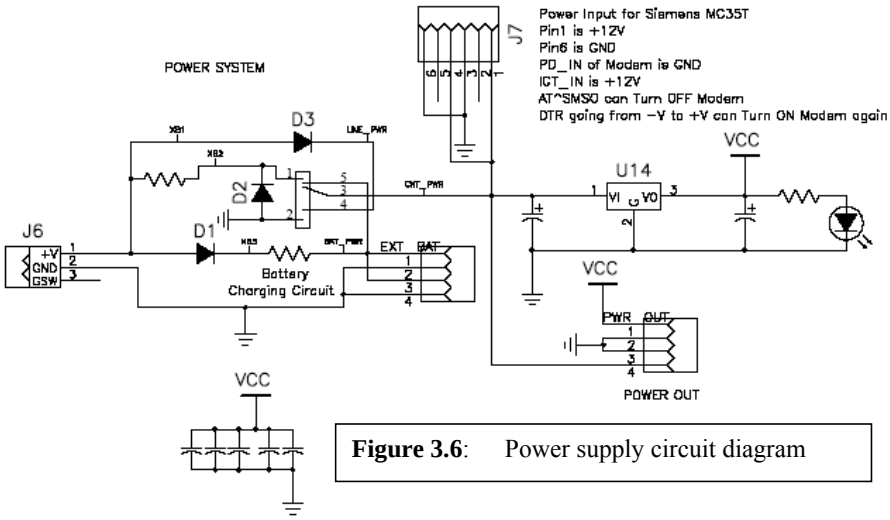
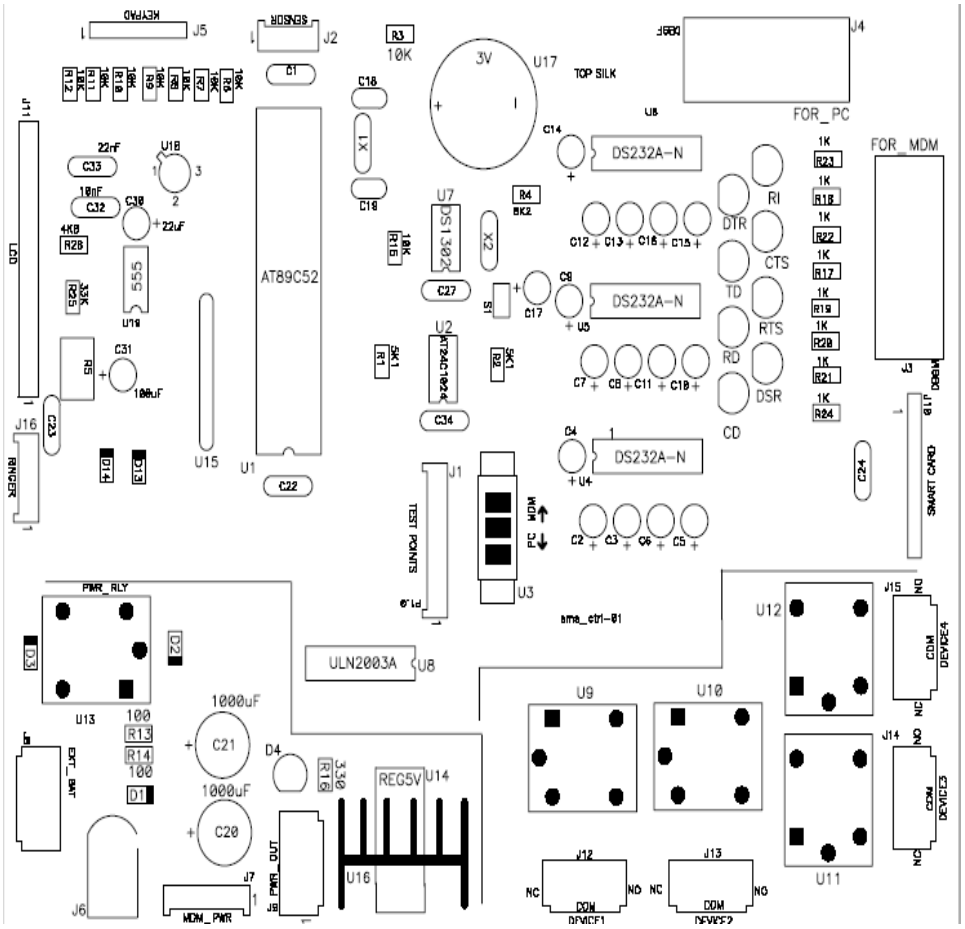


Figure 3.6: Power supply circuit diagram

Figure 3.1: Schematic entry diagram


KEYPAD
SENSOR
J5
J2
R3
10K
3V
U17
TOP SILK
J4
FOR_PC
FOR_MDM
C1
C18
X1
C19
C14
DS232A-N
U8
J11
LCD
22nF
C33
U18
18nF
C32
C30
22uF
4KB
R28
555
U19
33K
R25
AT89C52
U7
DS1302
X2
R4
BK2
C27
S1
C17
C9
DS232A-N
U5
U2
MAX240/1024
SK1
R1
SK1
R2
C34
C12 C13 C16 C15
C7 C8 C11 C10
C4
DS232A-N
U4
C2 C3 C6 C5
DTR
RI
CTS
TD
RTS
RD
DSR
CD
1K
R23
1K
R18
1K
R22
1K
R17
1K
R19
1K
R20
1K
R21
1K
R24
J3
DB9M
SMART CARD
J16
RINGER
R5
C31
100uF
C23
D14
D13
U15
U1
C22
J1
TEST POINTS
MDM
PC
U3
ama_ctrl-01
C24
PWR_RLY
D3
D2
U13
ULN2003A
U8
U12
J15
NO
COM
DEVICE4
NC
U9
U10
100
R13
R14
100
D1
1000uF
C21
1000uF
C20
D4
330
R19
REG5V
U14
EXT_BAT
J6
J7
MDM_PWR
PWR_OUT
J8
U16
J12
NC
NO
COM
DEVICE1
J13
NC
NO
COM
DEVICE2
U11
J14
NO
COM
DEVICE3
NC

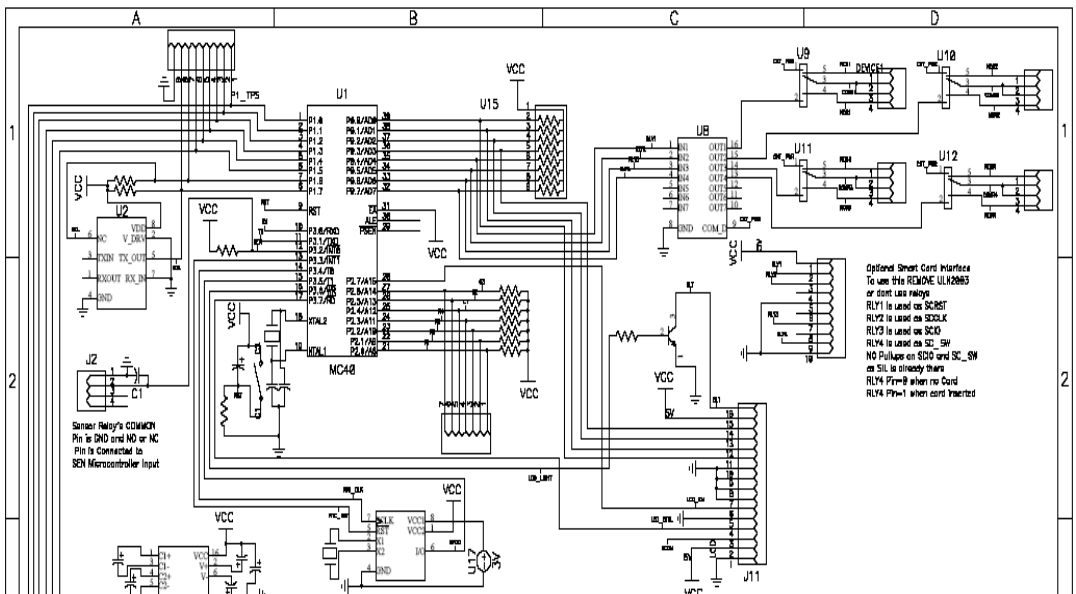
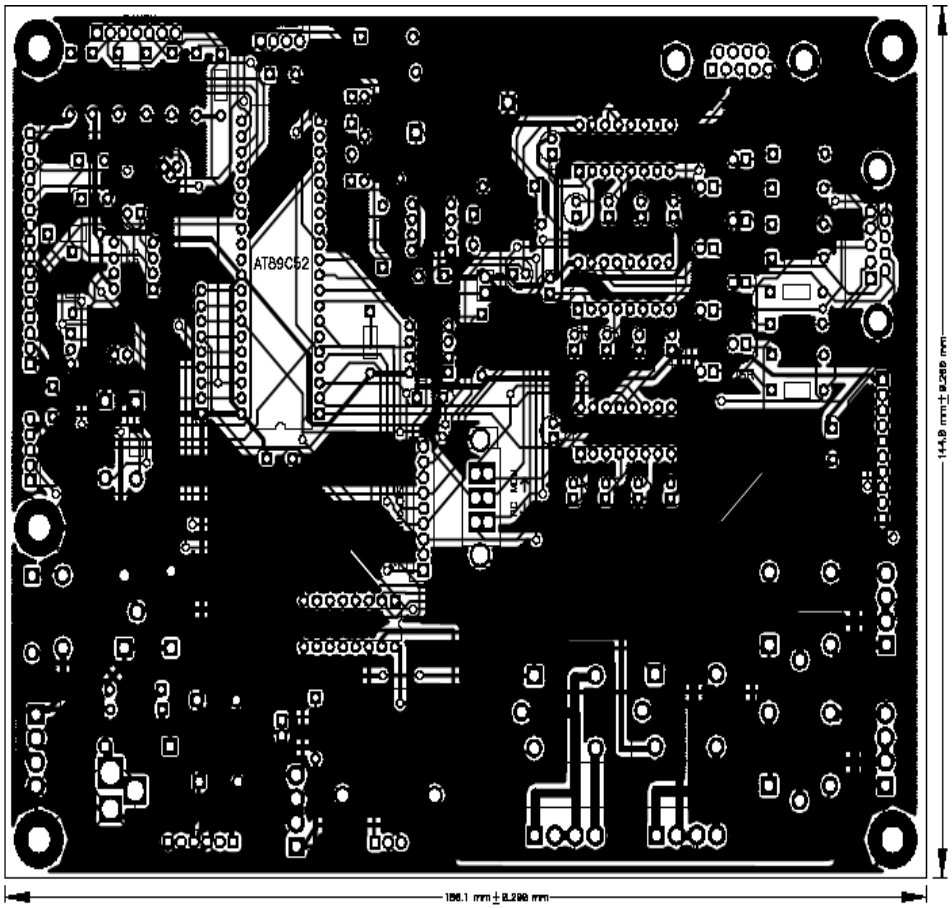

A
B
C
D
1
2
U1
P1_TPS
MC48
U15
U8
U9
U10
U11
U12
VCC
J2
C1
Sensor Relay's COMMON
Pin is GND and NO or NC
Pin is Connected to
SEN Microcontroller Input
Optional Smart Card Interface
To use this REMOVE ULN2003
or dont use relays
RLY1 is used as SCRST
RLY2 is used as SCCLK
RLY3 is used as SCIO
RLY4 is used as SC_SW
NO Pullups on SCIO and SC_SW
as SIL is already there
RLY4 Pin=0 when no Card
RLY4 Pin=1 when card inserted
U17
J11

Figure 3.2: Components placement diagram

Figure 3.3: The routed diagram



4.1 **Program flow chart**

The main flowchart of the program along with the explanation of the various blocks is given below

Figure 4.1: Program flow chart

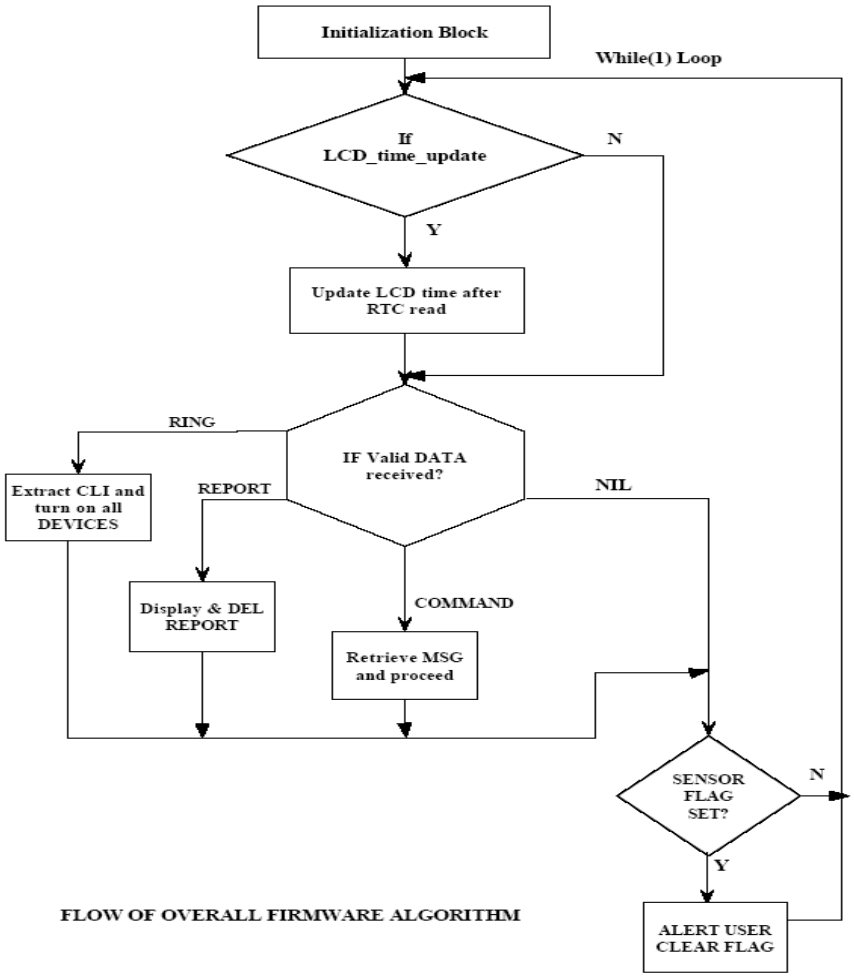
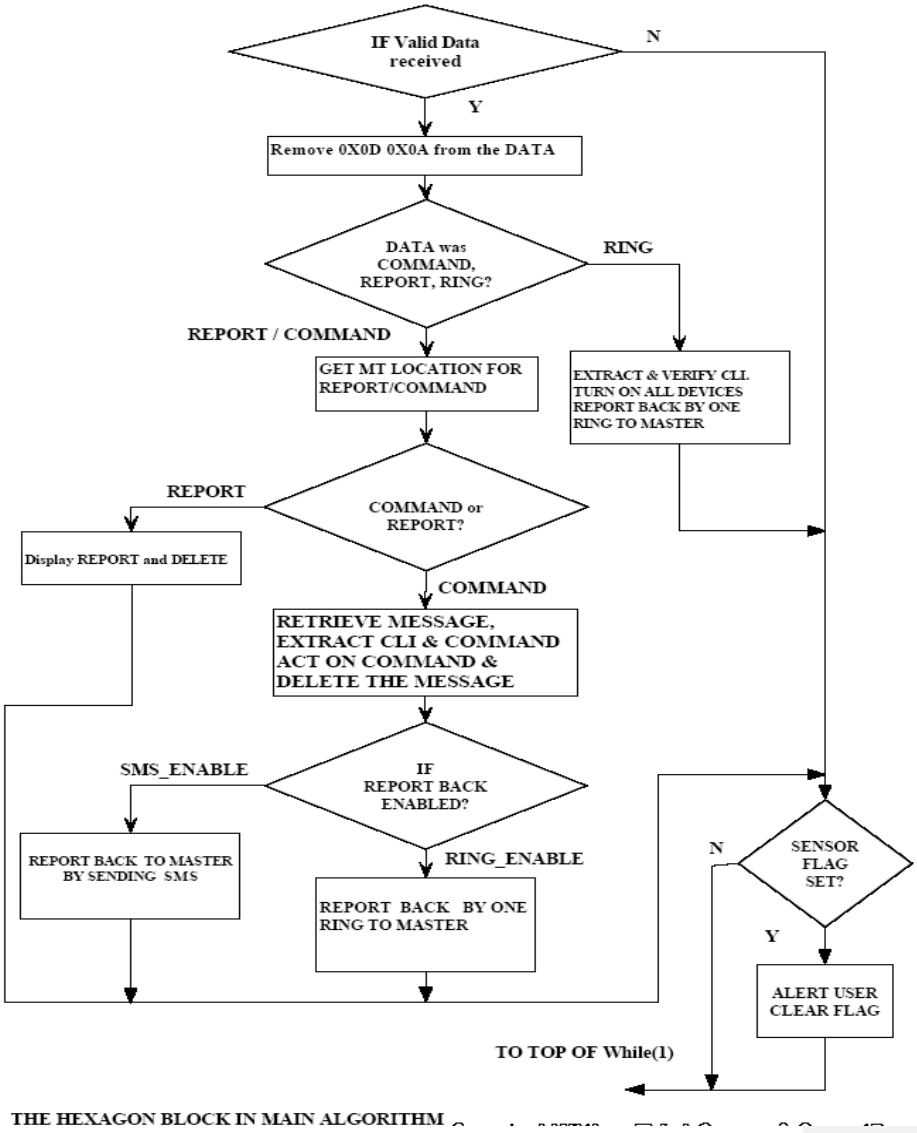


Figure 4.2: Blowup of Hexagonal block in main program flowchart



THE HEXAGON BLOCK IN MAIN ALGORITHM

4.2 **Explanation of flow chart**

1. **Initialization Block**: Initialize all Microcontroller SFRs e.g. for Timers, Baud rate generation, Serial Port Settings, Timer/Serial Port/External Interrupts etc. LCD is also initialized. Modem & RTC are also initiated in this Step. This Initialization takes place only upon power up or pressing of Reset switch. After Initialization, the code enters a while (1) loop.

2. **LCD_ time _update**: When the unit is IDLE, it is always displaying time, so this block is inside the while (1) loop. New Time is read from the RTC (through the proprietary code) and is updated on the LCD. Time update is suspended when other tasks are being carried out like acting on a command etc.

3. **Valid _ DATA _ Reception**: This is the most important Block and is responsible for the handling of the Master Rings and SMS commands. The Hexagon block in flowchart1 is explained and elaborated further in the flowchart figure 4.2. At the heart of the firmware is the Interrupt based Character reception from the 89C52 UART. The Serial port interrupts the main program whenever there is a character available on the port. The Interrupt Service Routine (ISR) [refernce void SerialP(void) interrupt 4 using 2 in SerialPort.c] of the Serial port does some key things like buffering the character into the receive array, incrementing the data counter, resetting the timer2_counter and serial_timeout flags etc. There is another ISR running and it's purpose is to signal a Serial Port time out (i.e. all characters have been received on the port for a particular serial

packet sent by MC35) so we can begin processing the data. This ISR is of Timer 2[reference void Timer2(void) interrupt 5 using 2 in GenFunctions.c]. Upon every

Interrupt on Serial Port, the variable timer2_counter and flag serial_timeout is reset to 0. Upon every entry into Timer 2 overflow interrupt, timer2_counter is incremented and at a threshold value of timer2_counter, the serial_timeout flag is set to 1. This can only occur if there was no interrupt on Serial Port for a considerable time, and hence time out occurs.

This time-out checking is performed inside the Hexagon block as well. After the timeout has occurred further processing begins which is shown in the flowcharts. This interrupt driven programming has the advantage that the data on serial port is not missed and controller doesn't have to waste time in polling the Serial Port. When there is no interrupt on Serial Port, controller is doing other functions. Like update time or process the received data etc.

4. The main code is attached at the end of the documentation as Appendix F: Project code.

4.3 What is firm ware

A microcontroller is not an intelligent device unless or until it is made intelligent by its software which in the terms of computer hardware is termed as the firmware. Just like any hardware has got its specific software a microcontroller has got a firmware. The firmware has to be programmed by the user itself. This chapter of our write-up deals with the firmware programming, the testing and the debugging phase of the microcontroller.

4.4 Basic information about the code

After going through the programming features of the microcontroller, having an in depth analysis of the memory and its limitations, going through the flowchart and algorithm phase in detail, enabled us to write the main program of our microcontroller which will control the entire system we have designed. The complete code has been written in the ASSEMBLY LANGUAGE had we have used the KEIL COMPILER for compiling the program. The complete code of the program has been given in the end of the write up .The details of the program are not required here as because our main aim is not to explain the code but to give just an overview of the language and compiler we have used for writing our program.

4.5 **Keil compiler**[10]

The 8051 Family is one of the fastest growing Microcontroller Architectures. More than 400 device variants from various silicon vendors are today available. New extended 8051 Devices, like the Philips 80C51MX architecture are dedicated for large application with several Mbytes code and data space. For optimum support of these different 8051 variants, Keil provides the several Development tools that are listed in the table below. The CX51 Compiler is a variant of the C51 compiler that is design for the new Philips 80C51MX architecture.

Table 4.1: Keil development tools

[10] Keil compiler available at www.keil.com

Development Tools	Support Microcontrollers, Description
C51 Compiler A51 Macro Assembler BL51 Linker/Locater	Development Tools for **classic 8051**. Includes support for 32 x 64KB code banks.
C51 Compiler (with OMF2 Output) AX51 Macro Assembler LX51 Linker/Locater	Development Tools for **classic 8051** and **extended 8051 variants** (like Dallas 390). Includes support for code banking and up to 16MB code and xdata memory.
CX51 Compiler AX51 Macro Assembler LX51 Extended Linker/Locater	Development Tools for **Philips 80C51MX** Supports up to 16MB code and xdata memory.

4.6 **Testing and debugging**

After writing the code and getting it compiled in the KEIL COMPILER, the next step was to get the code tested and debug it if any problems were arising. The code writing phase and its execution in a proper way was one of the most major tasks that were at our hand. The basic problem that we had to cater for the network of the cellular company we were using. We have got a sim in our modem, once we send a sms from any number it is bound to reach the destined number in few seconds but if the network is down, the sms is not delivered and the basic input to the microcontroller is not available so this was the major problem that we encountered while writing the programme. we have tried to debug this problem as much as we can by using different techniques and we can state here that so far we have been successful.

Putting it all together, we were able to write a source code of the heart of our system that is the microcontroller which controls the entire system that we have manufactured in our degree project.

When we build our target in the Keil, all code files are combined into a single executable (HEX file) for the required Microcontroller. This HEX file is then supplied to

the Device Programmer to download it into the FLASH of the Microcontroller. The Microcontroller is thus said to be "Programmed" with a firmware.

5.1 Learning through the project

This project provided us with the opportunity to learn skills related to following fields.

1) Communicating with the MODEM on AT command level.
2) RS232 Communication of modem and the host MCU.
3) Development of embedded code for the host MCU.
4) Design of logic and other control circuitry residing on the MCU host card.
5) PCB design entry and board level designing.

5.2 Future recommendations

~~1)~~ The unit can be made capable of handling Multiple SMS commands and RINGS at a time. This will require a better controller (probably a fast variant of 8051 family, like 89C420) and a firmware written as a Real Time OS. A better GSM modem may also be required. Also Status report Handling can be improved.

~~2)~~ The EEPROM on the board can be used to store the events in the form of a LOG and transferred to the PC for reporting purposes. In more advance case, the unit can upload the LOG file to a remote location through GPRS or data call. The EEPROM can also be used to store current Device Status (like which device is ON or OFF) and hence the Master can also check back his device status at anytime through a command. The basic

Formatted: Bullets and Numbering

EEPROM code has been developed, but due to lack of enough Program Memory (just 8k) this feature has not been incorporated.

~~3)~~ The Problems related to GSM network like delayed deliveries should be handled with some suitable Algorithm.

REFERENCES

1. MC-35 datasheet. pdf available at www.siemens.com/wm
2. Hardware interface description MC-35.pdf available at www.siemens.com/wm
3. Fundamentals of RS–232 Serial Communications. pdf at Dallas semiconductors website
4. 8052 Tutorial & Reference. pdf available at http://www.8052.com
5. AT89C52.pdf available at Atmel semiconductors website
6. DS1302. pdf available at Dallas semiconductors website
7. Max 232 line driver IC. Pdf available at www.maxim-ic.com
8. ULN-2003.pdf available at STMicroelectronics website
9. www.cadx-pcb.com/PCAD.html
10. www.keil.com
11. Seimens MC-35 AT commands.pdf, at http://www.siemens.com/wm

GENERAL REFERENCES

1. Draft EN (GSM 03.40) V 6.0.0 (1998-2003); technical realization of short message and point to point communication.
2. Digital cellular telecommunications system (Phase 2+); Use of (DTE - DCE) interface for Short Message Service

3. Digital cellular telecommunications system (Phase 2+); AT command set for GSM Mobile Equipment

4. The 8051 microcontroller (third edition) by I. Scott Mackenzie