SMS BASED STUDENT REGISTRATION SYSTEM

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

Electronics and Communication Engineering

By

DHRUV KAPOOR – 061046

ASTITVA BHARGAVA – 061047

GAURAV GUPTA – 061052

under the Supervision of

Dr. D.S. SAINI

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

MAY 2010

Jaypee University of Information Technology, Waknaghat
Solan - 173 234, Himachal Pradesh

Certificate

This is to certify that the project report entitled "SMS Based Student Registration System", submitted by Dhruv Kapoor, Astitva Bhargava and Gaurav Gupta in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

Date: 21.05.10

(Dr. D.S. Saini)
(Assistant Professor)
(Electronics and Communication)

It is certified that this work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

1. **(Dhruv Kapoor)**
   (061046)
2. **(Astitva Bhargava)**
   (061047)
3. **(Gaurav Gupta)**
   (061052)

Acknowledgement

We express our sincere gratitude and thanks to all those who have helped us in the completion of this project.

Of all the persons who have helped us, we would first like to thank **Dr. D.S. Saini,** (Assistant Professor, Electronics and Communication Department), under whose able guidance we have completed our project and who helped us at each and every stage of our project.

We also thank all the staff members of our college for their help in making this project a successful one.

Finally, we take this opportunity to extend our deep appreciation to our **family** and **friends**, for all that they meant to us during the crucial times of the completion of our project.

Date: 21·05·10

Astitva

Gupta

Dhruv Kapoor **(061046)**

Astitva Bhargava **(061047)**

Gaurav Gupta **(061052)**

Table of Contents

List of Figures

Page

List of Tables Page

List of Abbreviations

AC – Alternating Current

ADO – ActiveX Data Objects

AT – Attention

CD – Carrier Detect

CR – Carriage Return

CTS – Clear To Send

DCE – Data Communication Equipment

DD – Demand Draft

DSR – Data Set Ready

DTE – Data Terminal Equipment

DTR – Data Terminal Ready

GPRS – Ground Packet Radio Service

GSM – Global System for Mobiles

IMEI – International Mobile Equipment Identity

IMSI – International Mobile Subscriber Identity

MMS – Multimedia Message Service

PC – Personal Computer

SCI – Serial Communications Interfaces

SIM – Subscriber Identification Module

SMS – Short Message Service

RI – Ring Indicator

RS-232 – Recommended Standard no. 232

RTS – Request To Send

RXD – Receive Data

TXD – Transmit Data

USB – Universal Serial Bus

VB – Visual Basic

Abstract

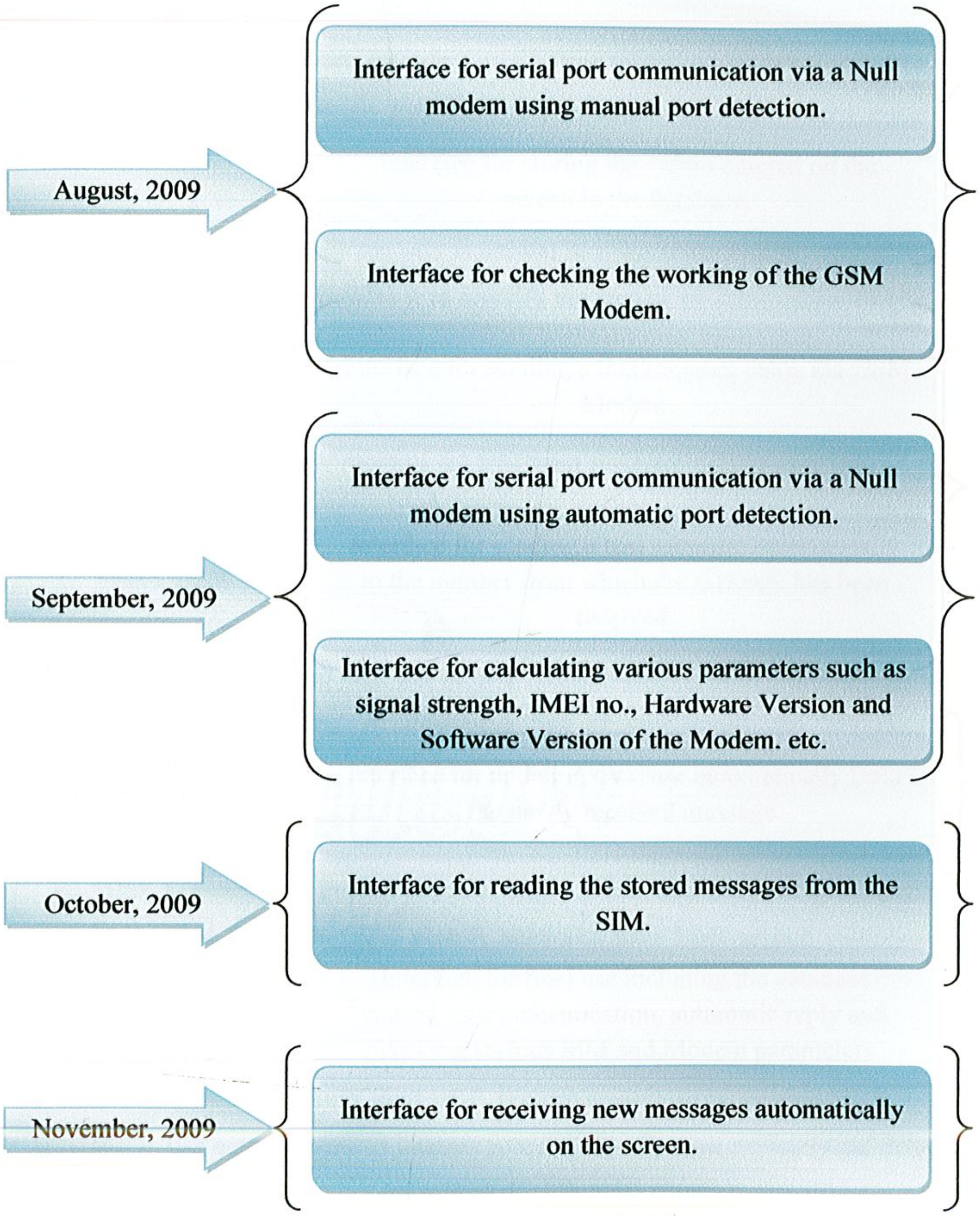
As the world goes mobile at an exponential rate, especially after the technological advancements making internet applications possible on handsets, the need to provide services on mobile through internet, SMSs and other modes of secure data and information transfer has also grown rapidly.

Therefore there is a pressing need for low cost yet flexible and secure applications for various financial and other security sensitive transactions via mobile. One of these applications related directly to us, is the *student registration system*. In this project we have successfully shown a functioning model for this system. Our model provides ideal solution to the problems faced by traditional registration system. The system is wireless, hence more adaptable and cost-effective. Also, it uses GSM technology, providing ubiquitous access to the student. The model has a futuristic scope as well.

Our model is based on the principle of SMS through GSM modem. As, the short message service (SMS) technology is one of the most stable mobile technologies around and most of the students carry mobile phones with SMS facilities, this can be exploited for registration. A text message is generated and sent to GSM modem in form of SMS. On receipt of SMS, GSM modem informs computer which performs specified task. The system alerts user in case of occurrence of any abnormal conditions like invalid format, wrong information etc.

Project Progress

This project has evolved over a period of one year and here are the various steps of the process through which the project taken its final shape. We shall be discussing the final algorithm in the sections to follow. The project timeline is shown in the following figure.



August, 2009

Interface for serial port communication via a Null modem using manual port detection.

Interface for checking the working of the GSM Modem.

September, 2009

Interface for serial port communication via a Null modem using automatic port detection.

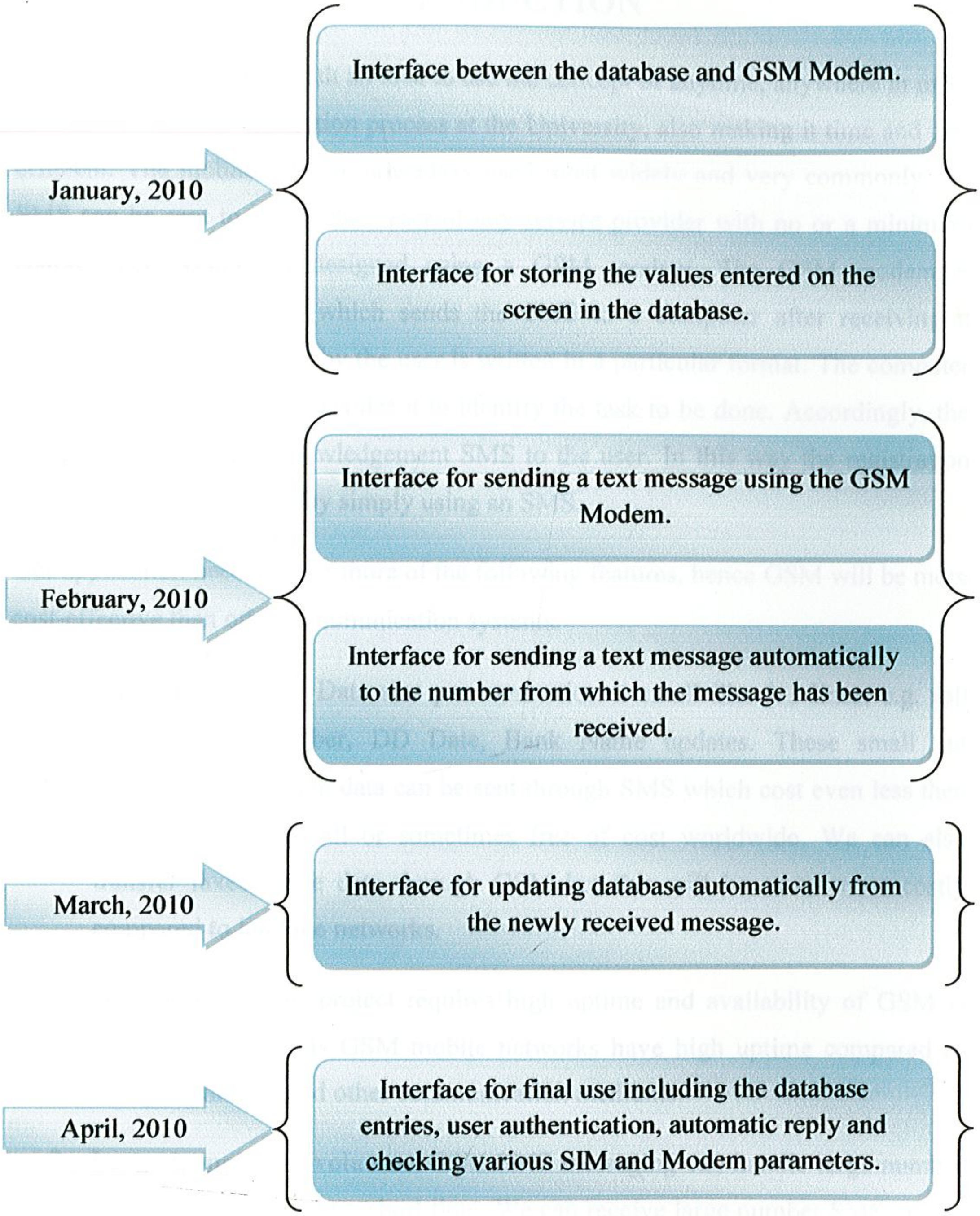
Interface for calculating various parameters such as signal strength, IMEI no., Hardware Version and Software Version of the Modem. etc.

October, 2009

Interface for reading the stored messages from the SIM.

November, 2009

Interface for receiving new messages automatically on the screen.



January, 2010

Interface between the database and GSM Modem.

Interface for storing the values entered on the screen in the database.

February, 2010

Interface for sending a text message using the GSM Modem.

Interface for sending a text message automatically to the number from which the message has been received.

March, 2010

Interface for updating database automatically from the newly received message.

April, 2010

Interface for final use including the database entries, user authentication, automatic reply and checking various SIM and Modem parameters.

CHAPTER 1
INTRODUCTION

This project is designed with an idea to use the concept of anytime, anywhere in order to ease the student registration process at the University, also making it time and cost efficient. The mobile sets are nowadays used most widely and very commonly. An SMS can be sent to any mobile user of any service provider with no or a minimum charge. This system is designed using a GSM modem. The GSM modem is configured as a receiver which sends the SMS to a computer after receiving it completely. The SMS sent by the user is written in a particular format. The computer receives the message and decodes it to identify the task to be done. Accordingly, the computer sends one acknowledgement SMS to the user. In this way the registration process can be completed by simply using an SMS.

Our application needs one or more of the following features, hence GSM will be more cost-effective than other communication systems.

1. **Short Data Size**: Data size per transaction is small like 1-3 lines, e.g. roll number, DD number, DD Date, Bank Name updates. These small but important transaction data can be sent through SMS which cost even less then a local telephone call or sometimes free of cost worldwide. We can also transfer faxes, large data through GSM but this will be as or more costly compared to landline networks.

2. **High uptime**: Our project requires high uptime and availability of GSM is best suited for us as GSM mobile networks have high uptime compared to landline, internet and other communication mediums.

3. **Large transaction volumes**: GSM SMS messaging can handle large number of transactions in a very short time. We can receive large number SMS on our server like e-mails without internet connectivity. E-mails normally get delayed a lot but SMS are almost instantaneous. Consider situation like students doing registration with GSM technology instead of conventional methods. Here GSM connection is enough to handle hundreds of transaction per minute.

4. **Mobility, Quick installation:** GSM technology allows mobility, GSM terminals, modems can be just picked and installed at any location unlike telephone lines. Also we can be mobile with GSM terminals and can also communicate with server using mobile phone. We have to just purchase the GSM hardware like modems, terminals and mobile handsets, insert SIM cards, configure software and are ready for GSM communication.

CHAPTER 2
REVIEW OF BASIC CONCEPTS

2.1 Serial Port Communication

A serial port is a serial communication physical interface through which information transfers in or out one bit at a time. The term "serial port" usually identifies hardware more or less compliant to the RS-232 standard, intended to interface with a modem or with a similar communication device.

2.1.1 Types of serial port communication

There are two basic types of serial communication, synchronous and asynchronous. With synchronous communication, the two devices initially synchronize themselves to each other, and then continually send characters to stay in sync. Even when data is not really being sent, a constant flow of bits allows each device to know where the other is at any given time. That is, each character that is sent is either actual data or an idle character. Synchronous communication allows faster data transfer rates than asynchronous methods, because additional bits to mark the beginning and end of each data byte are not required. The serial ports on IBM-style PCs are asynchronous devices and therefore only support asynchronous serial communication.

Asynchronous means "no synchronization", and thus does not require sending and receiving idle characters. However, the beginning and end of each byte of data must be identified by start and stop bits. The start bit indicates when the data byte is about to begin and the stop bit signals when it ends. The requirement to send these additional two bits cause asynchronous communication to be slightly slower than synchronous however it has the advantage that the processor does not have to deal with the additional idle characters.

An asynchronous line that is idle is identified with a value of 1, (also called a mark state). By using this value to indicate that no data is currently being sent, the devices are able to distinguish between an idle state and a disconnected line. When a character is about to be transmitted, a start bit is sent. A start bit has a value of 0, (also called a space state). Thus, when the line switches from a value of 1 to a value of 0, the receiver is alerted that a data character is about to come down the line.

2.1.2 Advantages of serial port communication

1. The serial port cable can be longer than a parallel port cable, as serial port transmits '1' as voltage from -5 to -12V and '0' as voltage from +5 to +12 V, while parallel port transmits '1' as voltage of 5 volts and '0' as voltage of 0 volts. At the same time the receiver of the serial port receives '1' as voltage from -3 to -25 V and '0' as voltage from +3 to +25 V. Thus serial port can have maximal swing up to 50 volts, while parallel port has maximal swing of 5 volts. Thus the losses in the cable when transmitting data using serial port are less substantial then losses when transmitting data using parallel port.
2. The number of wires needed when transmitting data serially is less than when the transmission is parallel. Is the external device has to be installed at a great distance from the computer, the cable with three wires is much cheaper than the cable with 19 or 25 wires if the transmission is parallel. Still one should remember that there are interface creation expenses for every receiver/transmitter.
3. Further development of serial port is usage of infrared devices which immediately proved popular. Many electronic diaries and palmtop computers have inbuilt infrared devices for connection with external devices.
4. Another proof of serial port universality is microcontrollers. Many of them have inbuilt SCI (Serial Communications Interfaces), used for communication with other devices. In this case serial interface reduces the number of outputs on the chip. Usually only 2 outputs are used: Transmit Data (TXD) and Receive Data (RXD). Just compare that to minimum of 8 outputs when using 8-bit parallel connection.

2.1.3 RS-232

RS-232 (Recommended Standard - 232) is a telecommunications standard for binary serial communications between devices. It supplies the roadmap for the way devices speak to each other using serial ports. The devices are commonly referred to as a DTE (data terminal equipment) and DCE (data communications equipment); for example, a computer and modem, respectively.

RS-232 sets acceptable voltage and signal levels, along with common pin designations, or configurations, for wiring serial connector ports. It also specifies protocols for the control information passed between devices, which include such events such as indicating the beginning or end of a data stream.

2.1.4 RS-232C

RS-232 stands for Recommend Standard number 232 and C is the latest revision of the standard. The serial ports on most computers use a subset of the RS-232C standard. The full RS-232C standard specifies a 25-pin "D" connector of which 22 pins are used. Most of these pins are not needed for normal PC communications, and indeed, most new PCs are equipped with male D type connectors having only 9 pins.

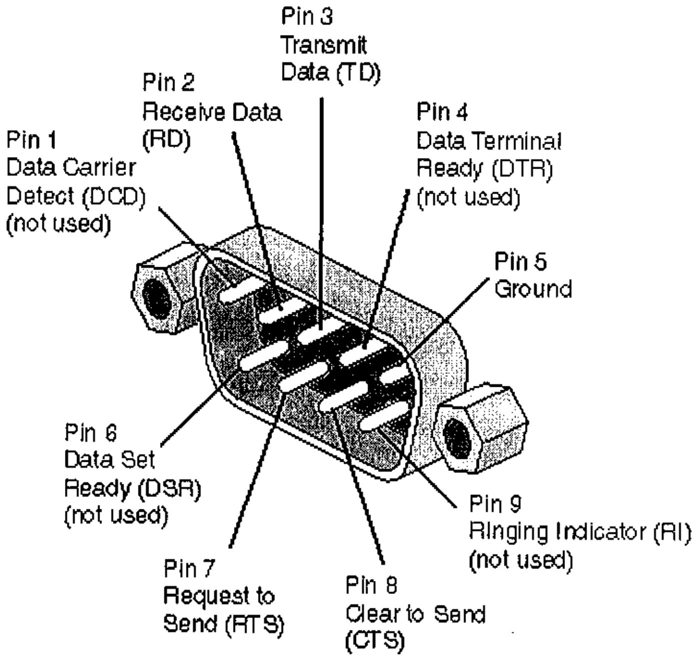


Figure 2.1 9 Pin Connector on a DTE device (PC connection)

The TD (transmit data) wire is the one through which data from a DTE device is transmitted to a DCE device. This name can be deceiving, because this wire is used by a DCE device to receive its data. The TD line is kept in a mark condition by the DTE device when it is idle. The RD (receive data) wire is the one on which data is received by a DTE device, and the DCE device keeps this line in a mark condition when idle.

RTS stands for **Request to Send**. This line and the CTS line are used when "hardware flow control" is enabled in both the DTE and DCE devices. The DTE device puts this line in a mark condition to tell the remote device that it is ready and able to receive data. If the DTE device is not able to receive data (typically because its receive buffer is almost full), it will put this line in the space condition as a signal to the DCE to stop sending data. When the DTE device is ready to receive more data (i.e. after data has been removed from its receive buffer), it will place this line back in the mark condition. The complement of the RTS wire is CTS, which stands for Clear to Send. The DCE device puts this line in a mark condition to tell the DTE device that it is ready to receive the data. Likewise, if the DCE device is unable to receive data, it will place this line in the space condition. Together, these two lines make up what is called RTS/CTS or "hardware" flow control. The Software Wedge supports this type of flow control, as well as XON/XOFF or "software" flow control. Software flow control uses special control characters transmitted from one device to another to tell the other device to stop or start sending data. With software flow control the RTS and CTS lines are not used.

DTR stands for **Data Terminal Ready**. Its intended function is very similar to the RTS line. DSR (Data Set Ready) is the companion to DTR in the same way that CTS is to RTS. Some serial devices use DTR and DSR as signals to simply confirm that a device is connected and is turned on. The Software Wedge sets DTR to the mark state when the serial port is opened and leaves it in that state until the port is closed. The DTR and DSR lines were originally designed to provide an alternate method of hardware handshaking. It would be pointless to use both RTS/CTS and DTR/DSR for flow control signals at the same time. Because of this, DTR and DSR are rarely used for flow control.

CD stands for **Carrier Detect.** Carrier Detect is used by a modem to signal that it has a made a connection with another modem, or has detected a carrier tone.

The last remaining line is **RI** or **Ring Indicator.** A modem toggles the state of this line when an incoming call rings your phone.

The Carrier Detect (CD) and the Ring Indicator (RI) lines are only available in connections to a modem. Because most modems transmit status information to a PC when either a carrier signal is detected (i.e. when a connection is made to another modem) or when the line is ringing, these two lines are rarely used.

2.2 GSM Modem

A GSM modem is a wireless modem that works with a GSM wireless network. A wireless modem behaves like a dial-up modem. The main difference between them is that a dial-up modem sends and receives data through a fixed telephone line while a wireless modem sends and receives data through radio waves.

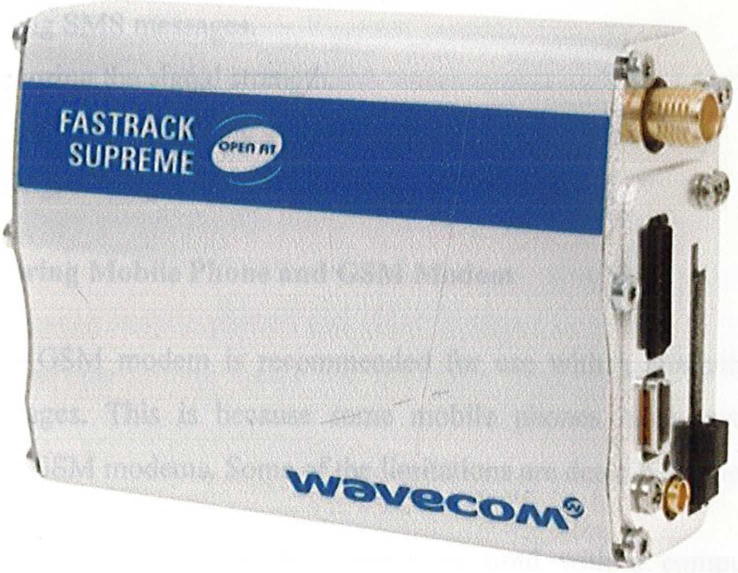


Figure 2.2 A GSM Modem

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. A GSM modem can be a dedicated modem device with a serial, USB or Bluetooth connection, or it may be a mobile phone that provides GSM modem capabilities. Both GSM modems and dial-up modems support a common set of standard AT commands. You can use a GSM modem just like a dial-up modem.

A GSM modem can be an external device or a PC Card / PCMCIA Card. Typically, an external GSM modem is connected to a computer through a serial cable or a USB cable. A GSM modem in the form of a PC Card / PCMCIA Card is designed for use with a laptop computer. It should be inserted into one of the PC Card / PCMCIA Card slots of a laptop computer.

In addition to the standard AT commands, GSM modems support an extended set of AT commands. These extended AT commands are defined in the GSM standards. These extended AT commands provide features like:

- Reading, writing and deleting SMS messages.
- Sending SMS messages.
- Monitoring the signal strength.
- Monitoring the charging status and charge level of the battery.
- Reading, writing and searching phone book entries.

2.2.1 Comparing Mobile Phone and GSM Modem

In general, a GSM modem is recommended for use with a computer to send and receive messages. This is because some mobile phones have certain limitations comparing to GSM modems. Some of the limitations are described below:

- Some mobile phone models cannot be used with a computer to receive concatenated SMS messages. A concatenated SMS message is a message that contains more than 140 bytes. (A normal SMS message can only contain at most 140 bytes.) Concatenated SMS works like this: the sender's mobile device breaks a message longer than 140 bytes into smaller parts. Each of these parts are then fitted in a single SMS message and sent to the recipient. When these SMS messages reach the destination, the recipient's mobile device will combine them back to one message. When the mobile phone receives the SMS messages that are parts of a concatenated SMS message, it combines them to one message automatically. The correct behavior should be: when the mobile phone receives the SMS messages that are parts of a concatenated SMS message, it forwards them to the computer without combining them.

- Many mobile phone models cannot be used with a computer to receive MMS messages. Because when they receive a MMS notification, they handle it automatically instead of forwarding it to the computer.

- A mobile phone may not support some AT commands, command parameters and parameter values. Usually GSM modems support a more complete set of AT commands than mobile phones.

- Most SMS messaging applications have to be available 24 hours a day. If such SMS messaging applications use mobile phones to send and receive SMS messages, the mobile phones have to be switched on all the time. However, some mobile phone models cannot operate with the battery removed even when an AC adaptor is connected, which means the battery will be charged 24 hours a day.

Besides the above issues, mobile phones and GSM modems are more or less the same for sending and receiving SMS messages from a computer. There is not much difference between mobile phones and GSM modems in terms of SMS transmission rate, since the determining factor for the SMS transmission rate is the wireless network.

2.3 Introduction to AT Commands

AT commands are instructions used to control a modem. AT is the abbreviation of ATtention. Every command line starts with "AT" or "at". That's why modem commands are called AT commands. Many of the commands that are used to control wired dial-up modems, such as ATD (Dial), ATA (Answer), ATH (Hook control) and ATO (Return to online data state), are also supported by GSM/GPRS modems and mobile phones. Besides this common AT command set, GSM/GPRS modems and mobile phones support an AT command set that is specific to the GSM technology, which includes SMS-related commands like AT+CMGS (Send SMS message), AT+CMSS (Send SMS message from storage), AT+CMGL (List SMS messages) and AT+CMGR (Read SMS messages).

Note that the starting "AT" is the prefix that informs the modem about the start of a command line. It is not part of the AT command name. For example, D is the actual AT command name in ATD and +CMGS is the actual AT command name in AT+CMGS. However, some books and web sites use them interchangeably as the name of an AT command.

Here are some of the tasks that can be done using AT commands with a GSM/GPRS modem or mobile phone:

- Get basic information about the mobile phone or GSM/GPRS modem. For example, name of manufacturer (AT+CGMI), model number (AT+CGMM), IMEI number (International Mobile Equipment Identity) (AT+CGSN) and software version (AT+CGMR).
- Get basic information about the subscriber. For example, MSISDN (AT+CNUM) and IMSI number (International Mobile Subscriber Identity) (AT+CIMI).
- Get the current status of the mobile phone or GSM/GPRS modem. For example, mobile phone activity status (AT+CPAS), mobile network registration status (AT+CREG), radio signal strength (AT+CSQ), battery charge level and battery charging status (AT+CBC).
- Establish a data connection or voice connection to a remote modem (ATD, ATA, etc).
- Send and receive fax (ATD, ATA, AT+F*).
- Send (AT+CMGS, AT+CMSS), read (AT+CMGR, AT+CMGL), write (AT+CMGW) or delete (AT+CMGD) SMS messages and obtain notifications of newly received SMS messages (AT+CNMI).
- Read (AT+CPBR), write (AT+CPBW) or search (AT+CPBF) phonebook entries.
- Perform security-related tasks, such as opening or closing facility locks (AT+CLCK), checking whether a facility is locked (AT+CLCK) and changing passwords (AT+CPWD). (Facility lock examples: SIM lock [a password must

be given to the SIM card every time the mobile phone is switched on] and PH-SIM lock [a certain SIM card is associated with the mobile phone. To use other SIM cards with the mobile phone, a password must be entered.])

- Control the presentation of result codes / error messages of AT commands. For example, you can control whether to enable certain error messages (AT+CMEE) and whether error messages should be displayed in numeric format or verbose format (AT+CMEE=1 or AT+CMEE=2).
- Get or change the configurations of the mobile phone or GSM/GPRS modem. For example, change the GSM network (AT+COPS), bearer service type (AT+CBST), radio link protocol parameters (AT+CRLP), SMS center address (AT+CSCA) and storage of SMS messages (AT+CPMS).
- Save and restore configurations of the mobile phone or GSM/GPRS modem. For example, save (AT+CSAS) and restore (AT+CRES) settings related to SMS messaging such as the SMS center address.

Note that mobile phone manufacturers usually do not implement all AT commands, command parameters and parameter values in their mobile phones. Also, the behavior of the implemented AT commands may be different from that defined in the standard. In general, GSM/GPRS modems designed for wireless applications have better support of AT commands than ordinary mobile phones.

In addition, some AT commands require the support of mobile network operators. For example, SMS over GPRS can be enabled on some GPRS mobile phones and GPRS modems with the +CGSMS command (command name in text: Select Service for MO SMS Messages). But if the mobile network operator does not support the transmission of SMS over GPRS, you cannot use this feature.

2.3.1 Basic Commands and Extended Commands

There are two types of AT commands: basic commands and extended commands.

Basic commands are AT commands that do not start with "+". For example, D (Dial), A (Answer), H (Hook control) and O (Return to online data state) are basic commands.

Extended commands are AT commands that start with "+". All GSM AT commands are extended commands. For example, +CMGS (Send SMS message), +CMSS (Send SMS message from storage), +CMGL (List SMS messages) and +CMGR (Read SMS messages) are extended commands.

2.3.2 General Syntax of Extended AT Commands

The general syntax of extended AT commands is straightforward. The syntax rules are provided below. The syntax of basic AT commands is slightly different. All SMS messaging commands are extended AT commands.

Syntax rule 1: All command lines must start with "AT" and end with a carriage return character (<*CR*>). In a terminal program like HyperTerminal of Microsoft Windows, we can press the Enter key on the keyboard to output a carriage return character.

Example: To list all unread inbound SMS messages stored in the message storage area, type "AT", then the extended AT command "+CMGL", and finally a carriage return character, like this:

AT+CMGL<*CR*>

Syntax rule 2: A command line can contain more than one AT command. Only the first AT command should be prefixed with "AT". AT commands in the same command-line string should be separated with semicolons.

Example: To list all unread inbound SMS messages stored in the message storage area and obtain the manufacturer name of the mobile device, type "AT", then the extended AT command "+CMGL", followed by a semicolon and the next extended AT command "+CGMI":

AT+CMGL; +CGMI<*CR*>

An error will occur if both AT commands are prefixed with "AT", like this:

AT+CMGL; AT+CGMI<*CR*>

Syntax rule 3: A string is enclosed between double quotes.

Example: To read all SMS messages from message storage in SMS text mode, we need to assign the string "ALL" to the extended AT command +CMGL, like this:

```
AT+CMGL="ALL"<CR>
```

Syntax rule 4: Information responses and result codes always start and end with a carriage return character and a linefeed character.

Example: After sending the command line "AT+CGMI<*CR*>" to the mobile device, the mobile device should return a response similar to this:

```
<CR><LF>Nokia<CR><LF>
<CR><LF>OK<CR><LF>
```

The first line is the information response of the AT command +CGMI and the second line is the final result code. <*CR*> and <*LF*> represent a carriage return character and a line feed character respectively. The final result code "OK" marks the end of the response. It indicates no more data will be sent from the mobile device to the computer / PC.

When a terminal program such as HyperTerminal of Microsoft Windows sees a carriage return character, it moves the cursor to the beginning of the current line. When it sees a line feed character, it moves the cursor to the same position on the next line. Hence, the command line "AT+CGMI<*CR*>" that we entered and the corresponding response will be displayed like this in a terminal program such as HyperTerminal of Microsoft Windows:

```
AT+CGMI
Nokia
OK
```

The meanings of information response and final result code stated above are:

AT+CGMI	← Command line entered
Nokia	← Information response
OK	← Final result code

2.3.3 Case Sensitivity of AT Commands

In the SMS specification, all AT commands are in uppercase letters. However, many GSM/GPRS modems and mobile phones allow us to type AT commands in either uppercase or lowercase letters.

2.4 Serial Communication in Visual Basic

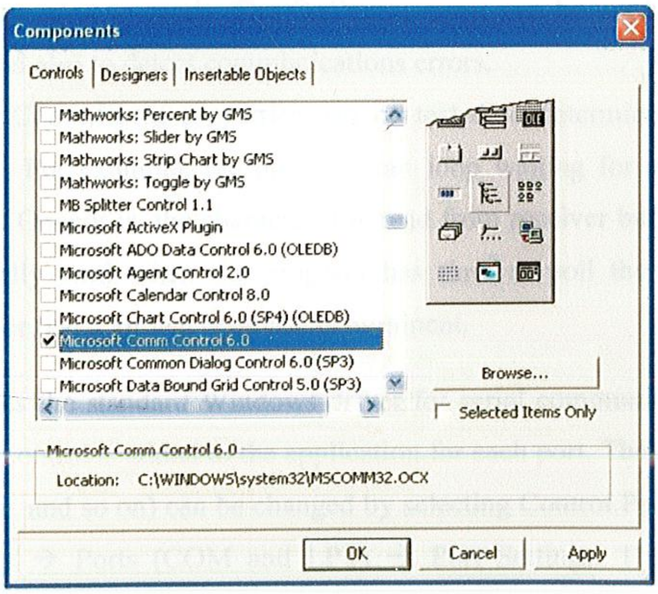
Visual Basic (VB) can be used to access serial communication functions. Windows hides much of the complexity of serial communications and automatically puts any received characters in a receive buffer and characters sent into a transmission buffer. The receive buffer can be read by the program whenever it has time and transmit buffer is emptied when it is free to send characters.

2.4.1 Communications Control

Visual Basic allows many additional components to be added to toolbox. The Microsoft Comm component is used to add a serial communication facility.

This is added to toolbox with: **Project→ Components (Ctrl-T)**

Figure 2.3 shows how a component is added in VB6. This then adds a Comms Component ☎ into the toolbox, as shown in Figure 2.4.



Components

Controls | Designers | Insertable Objects

□ Mathworks: Percent by GMS
□ Mathworks: Slider by GMS
□ Mathworks: Strip Chart by GMS
□ Mathworks: Toggle by GMS
□ MB Splitter Control 1.1
□ Microsoft ActiveX Plugin
□ Microsoft ADO Data Control 6.0 (OLEDB)
□ Microsoft Agent Control 2.0
□ Microsoft Calendar Control 8.0
□ Microsoft Chart Control 6.0 (SP4) (OLEDB)
☑ Microsoft Comm Control 6.0
□ Microsoft Common Dialog Control 6.0 (SP3)
□ Microsoft Data Bound Grid Control 5.0 (SP3)

□ Selected Items Only

Microsoft Comm Control 6.0

Location: C:\WINDOWS\system32\MSCOMM32.OCX

Figure 2.3 Adding Microsoft Comm component with Visual Basic 6

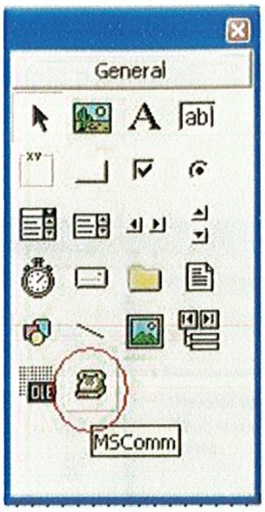


Figure 2.4 Toolbox showing Comms components

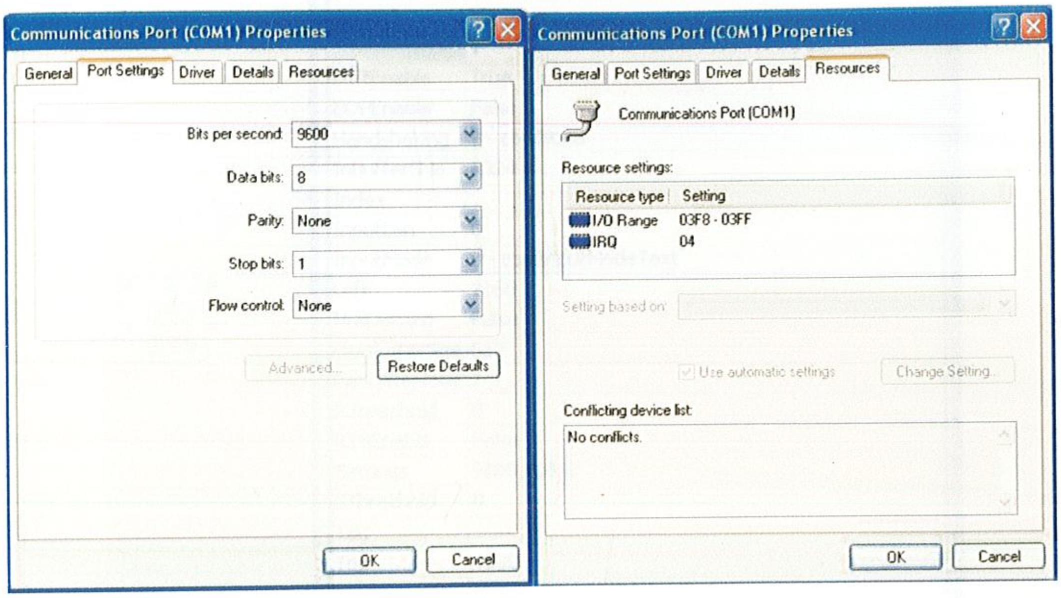
In order to use the Comms component the files MSCOMM32.OCX (for a 32-bit module) must be present in WINDOWS\SYSTEM directory. The class name is MSComm.

The communications control provides the following ways for handling communications:

1. **Event-driven:** Event-driven communications is the best method of serial communication as it frees the computer to do other things. The event can be defined as reception of a character, a change in CD (carrier detect) or a change in RTS (request to send). The OnComm event can be used to capture these events and also to detect communications errors.
2. **Polling:** CommEvent properties can be tested to determine if an error has occurred. For example, the program can loop waiting for a character to be received. Once it is, the character it is read from receiver buffer. This method is normally used when the program has time to poll the communications receiver or that a known response is imminent.

Visual Basic uses the standard Windows driver for serial communication ports. The communication control is added to the application for each port. The parameters (such as bit rate, parity, and so on) can be changed by selecting Control Panel → System → Device Manager → Ports (COM and LPT) → Port Settings. The settings of the communications port (the IRQ and the port address) can be changed by selecting

Control Panel → System → Device Manager → Ports → Resources for IRQ and Addresses. Figure 2.5 shows example parameters and settings.



Communications Port (COM1) Properties

Bits per second: 9600

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

Communications Port (COM1) Properties

Communications Port (COM1)

Resource settings:

Resource type	Setting
I/O Range	03F8 - 03FF
IRQ	04

Setting based on:

☑ Use automatic settings

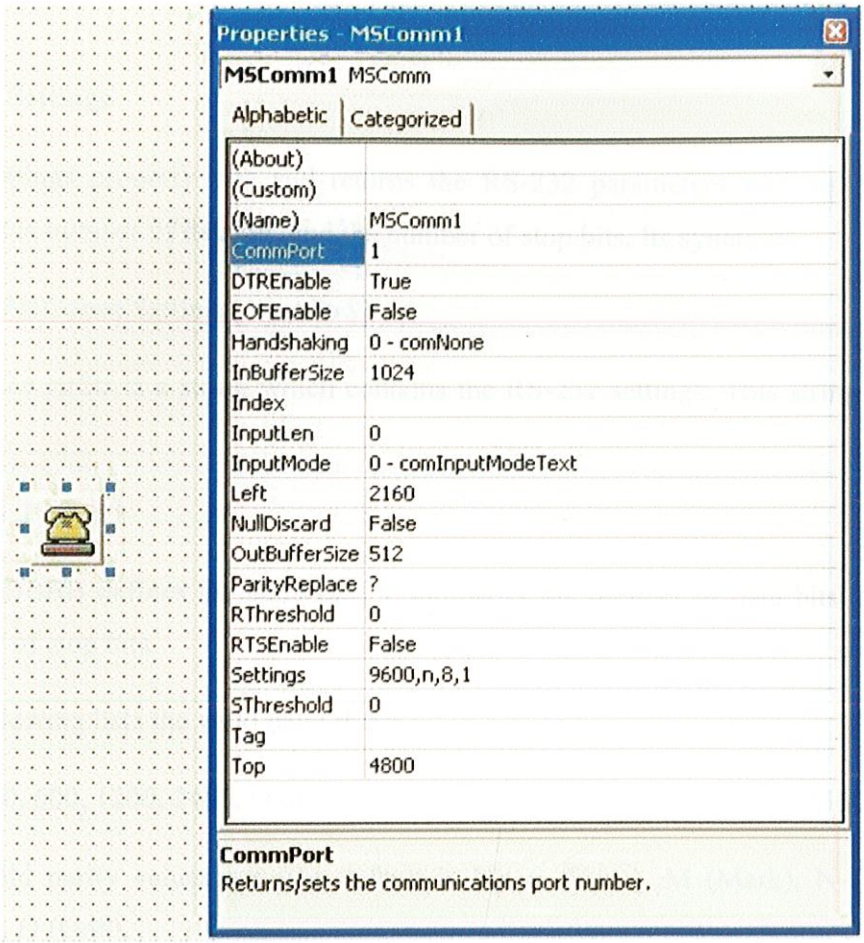
Conflicting device list:

No conflicts.

Figure 2.5 Changing port setting and parameters

2.4.2 Properties

The Comm Component is added to a form whenever serial communications are required (as shown in the left-hand side of figure 2.6). The right-hand side of Figure 2.6 shows its properties. By default the first created object is named as MSComm1 (the second is named MSComm2, and so on). It can be seen that main properties of the object are: CommPort, DTREnable, EOFEnable, Handshaking, InBufferSize, OutBufferSize, Index, InputLen, InputMode, Left, Name, NullDiscard, OutBufferSize, ParityReplace, RThershold, RTSEnable, Settings, SThreshold, Tag and Top. The main properties are defined in Table 2.1.



(About)	
(Custom)	
(Name)	MSComm1
CommPort	1
DTREnable	True
EOFEnable	False
Handshaking	0 - comNone
InBufferSize	1024
Index	
InputLen	0
InputMode	0 - comInputModeText
Left	2160
NullDiscard	False
OutBufferSize	512
ParityReplace	?
RThreshold	0
RTSEnable	False
Settings	9600,n,8,1
SThreshold	0
Tag	
Top	4800

CommPort
Returns/sets the communications port number.

Figure 2.6 Communications control and MS Comm Properties

Table 2.1 The main communications control properties

Properties	***Description***
CommPort	Sets and returns the communications port number
Input	Returns and removes characters from the receive buffer
Output	Writes a string of characters to the transmit buffer
PortOpen	Opens and closes a port, and gets port settings
Settings	Sets and returns port parameters, such as bit rate, parity number, number of data bits and so on

2.4.2.1 Settings

The Settings property sets and returns the RS-232 parameters such as baud rate, parity, the number of data bit, and the number of stop bits. Its syntax is:

[form.]*MSComm*.Settings[=*set Str*$]

where the strStr is a string which contains the RS-232 settings. This string takes the form:

"BBBB ,P, D, S"

Where BBBB defines the baud rate, P the parity the number of data bits, and S the number of stop bits.

The following lists the valid baud rates:

110, 300, 600, 1200, 2400, 9600, 14400, 19200, 38400, 56000, 128000, 256000.

The valid parity values are (the default is N): E (Even), M (Mark), N (None), S (Space), O (Odd).

The valid data bit values are (default is 8): 4, 5, 6, 7 or 8.

The valid stop bit values are (default is 1): 1, 1.5 or 2.

An example of setting a control port to 4800 baud, even parity, 7 data bits and 1 stop bit is:

Com1.settings = "4800, 7, E, 1"

2.4.2.2 CommPort

This property sets and returns the communication port number. Its syntax is:

[form.]*MSComm*.CommPort[=*portNumber*%]

which defines the port number value between 1 and 99. The value of 68 is returned if port does not exist.

2.4.2.3 PortOpen

The PortOpen property sets and returns the state of communication port. Its syntax is:

[form.]*MSComm*.PortOpen[={*True*|*False*}]

A True setting opens the port, while a False closes the port and clears the receive and transmit buffers (this happens automatically when an application is closed).

The following example opens communication port number 1(COM1) at 4800 baud, even parity, 7 data bits and 1 stop bit:

Com1.settings = "4800, 7, E, 1"

Com1CommPort = 1

Com1.PortOpen = True

2.4.2.4 Inputting Data

The three main properties used to read data from the receiver buffer are Input, InBufferCount and InBufferSize.

Input

This property returns and removes a string of characters from the receive buffer. Its syntax is:

[form.]*MSComm*.Input

To determine the number of characters in the buffer the InBufferCount property is tested. Setting InputLen to 0 causes the Input property to read the entire contents of the receive buffer.

InBufferSize

This property sets and returns the maximum number of characters that can be received in the receiver buffer (by default it is 1024 bytes). Its syntax is:

[form.]*MSComm*.InBufferSize[=*numBytes*%]

The size of buffer should be set so that it can store the maximum number of characters that can be received before the application program can read them from buffer.

InBufferCount

This property returns the number of characters in the receiver buffer. It can also be used to clear the buffer by setting the number of characters to 0. Its syntax is:

[form.]*MSComm*.InBufferCount[=*count*%]

2.4.2.5 Outputting data

The three main properties used to write data in transmit buffer are Output, OutBufferCount and OutBufferSize.

Output

The output property writes a string of characters to transmit buffer. Its syntax is:

[form.]*MSComm*.Output[=*outString*$]

OutBufferSize

This property sets and returns the number of characters in transmit buffer (default size is 512 bytes). Its syntax is:

[form.]*MSComm*.OutBufferSize[=*numBytes*%]

OutBufferCount

This property returns the number of characters in the transmit buffer. It can also be used to clear the buffer by setting the number of characters to 0. Its syntax is:

[form.]*MSComm*.OutBufferCount[=*0*]

2.4.2.6 Other Properties

Some other important properties are:

- **Break.** Sets or clears the break signal. A true sets the break signal while a False clears the break signal. When True character transmission is suspended and a break level is set on line. This continues until break signal is set to False. Its syntax is:

  [form.]*MSComm*.Break[={*True*|*False*}]

- **CDTiemout.** Sets and returns the maximum amount of time that the control waits for CD (Carrier Detect) signal in milliseconds, before a timeout. Its syntax is:

  [form.]*MSComm*.CDTimeout[=*milliseconds&*]

- **CTSHolding.** Determines whether CTS line should be detected. CTS is typically used for hardware handshaking. Its syntax is:

  [form.]*MSComm*.CTSHolding[={*True*|*False*}]

- **DSRHolding.** Determines the DSR line rate. DSR is typically used to indicate the modem. IF is a True then DSR line is high, else it is low. Its syntax is:

  [form.]*MSComm*.DSRHolding[=*setting*]

- **DSRTimeout.** Sets and returns the number of milliseconds to wait for DSR signal before an ONComm event occurs. Its syntax is:

  [form.]*MSComm*.DSRTimeout[=*milliseconds&*]

- **DTREnable.** Determines whether DTR signal is enabled. It is typically is send from computer to modem to indicate that it is ready to receive data. A True signal enables the DTR line (output level high). Its syntax is:

  [form.]*MSComm*.DTREnable[={*True*|*False*}]

- **RTSEnable.** Determines whether RTS signal is enabled. Normally used to handshake incoming data and is controlled by computer. Its syntax is:

  [form.]*MSComm*.RTSEnable[={*True*|*False*}]

- **NullDiscard.** Determines whether null characters are received into the receive buffer.A True setting does not transfer the characters. Its syntax is:

  [form.]*MSComm*.NullDiscard[={*True*|*False*}]

- **SThreshold.** Sets and returns the minimum number of characters allowable in the transmit buffer before the OnComm event. A 0 value disables generating the ONComm event for all transmission events, while a value of 1 causes the OnComm event to be called when transmit buffer is empty. Its syntax is:

  [form.]*MSComm*.SThreshold[*=numchars%*]

- **Handshaking.** Sets and returns the handshaking protocol. It can be set to no handshaking, hardware handshaking using RTS/CTS or software handshaking using XON/XOFF. Valid settings are given in Table 2.2. Its syntax is:

  [form.]*MSComm*.Handshaking[*=protocol%*]

- **CommEvent.** Returns the most recent error message. Its syntax is:

  [form.]*MSComm*.CommEvent

Table 2.2 Settings for handshaking

Setting	*Value*	*Description*
comNone	0	No handshaking (Default)
comXOnXOff	1	XON/XOFF handshaking
comRTS	2	RTS/CTS handshaking
comRTSXOnXOff	3	RTS/CTS and XON/XOFF handshaking

When a serial communication event (OnComm) occurs then the event (error or change) can be determined by testing the CommEvent property. Table 2.3 lists the error values and Table 2.4 lists the communication events.

Table 2.3 CommEvent properties

Setting	*Value*	*Description*
comBreak	1001	Break signal received
comCTSTO	1002	CTS Timeout
comDSRTO	1003	DSR Timeout
comFrame	1004	Framing error
comOverrun	1006	Port Overrun
comCDTO	1007	CD Timeout
comRxOver	1008	Receive buffer overflow
comRxParity	1009	Parity error
comTxFull	1010	Transmit buffer full

Table 2.4 Communications events

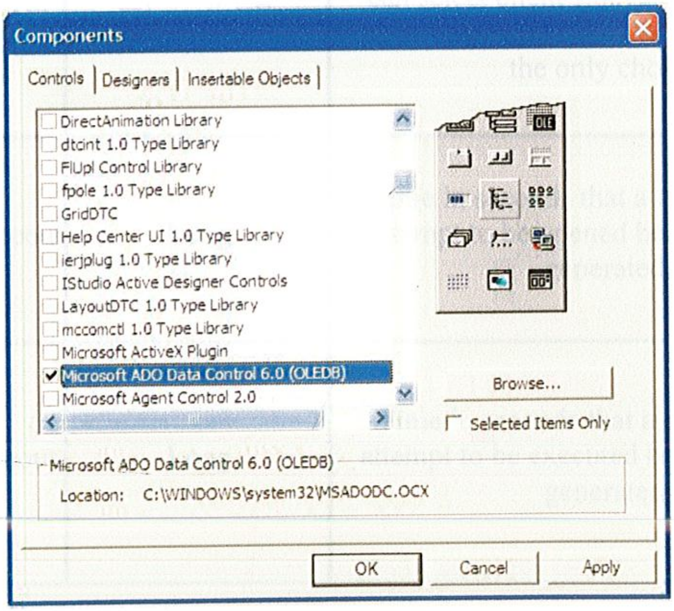
Setting	*Value*	*Description*
comEvSend	1	Character has been sent
comEvReceive	2	Character has been received
comEvCTS	3	Change in CTS line
comEvDSR	4	Change in DSR line from a high to a low
comEvCD	5	Change in CD line
comEvRing	6	Ring detected
comEvEOF	7	EOF character received

2.5 Microsoft Access and VB

Microsoft Access is an application used to create computer databases that can be used on a Microsoft Windows operating system, on a web site, or on a portable medium.

A database is a list of items stored somewhere to make their values easy to access or retrieve. This means that a database can exist anywhere, including human or non-human memory. A computer database is a list or a group of lists created as a project. There are various ways and various types of applications used to create such a list. To make it more useful, special computer applications are formally developed to help create and manage computer databases. In VB there are various libraries available for accessing database. Here, in our application we will use the following:

Microsoft ActiveX Data Objects: Microsoft ActiveX Data Objects, also called ADO, is a library that was developed to allow programmers with other environments to create and manage Microsoft Access databases. To support this, it provides a driver that allows these other programming environments to "attach" their applications to a Microsoft Access database. Like Microsoft Access' own library, you can use ADO inside of Microsoft Access to fully create and manage a database. With this library, ADODC component is available for accessing the database. Figure 2.7 shows how to add ADODC component.



□ DirectAnimation Library
□ dtcint 1.0 Type Library
□ FlUpl Control Library
□ fpole 1.0 Type Library
□ GridDTC
□ Help Center UI 1.0 Type Library
□ ierjplug 1.0 Type Library
□ IStudio Active Designer Controls
□ LayoutDTC 1.0 Type Library
□ mccomctl 1.0 Type Library
□ Microsoft ActiveX Plugin
☑ Microsoft ADO Data Control 6.0 (OLEDB)
□ Microsoft Agent Control 2.0

□ Selected Items Only

Microsoft ADO Data Control 6.0 (OLEDB)
Location: C:\WINDOWS\system32\MSADODC.OCX

Figure 2.7 Adding ADODC with Visual Basic 6

Once we have added the ADODC component in our form we need to set various properties. Table 2.5 shows some of the important properties:

Table 2.5 Some properties of ADODC

Name	Data Type	Description
ConnectionString	String	Defines in string form the location of the data source you wish to connect to. Key fields in the string you will use are the "Provider=" and the "Data Source=" fields. You may also use the "Mode=" field.
Provider	String	A string defining the provider of a connection object.
Mode	connectModeEnum	Sets (or returns) the permissions for modifying data across the open connection. Available permissions include Read, Write, Read/Write, and various Share/Deny types.
CursorLocation	cursorLocationEnum	Sets the location of the cursor engine. You can select client side or server side, for simpler databases (like Access) client side is the only choice.
ConnectionTimeout	Long	Time in seconds that a connection will attempt to be opened before an error is generated.
CommandTimeout	Long	Time in seconds that a command will attempt to be executed before an error is generated.

Apart from these properties there are also some methods associated with ADODC. Table 2.6 describes the important methods.

Table 2.6 Methods of ADODC

Name	Description
Close	Closes an open connection.
Open	Opens a connection with the settings in the ConnectionString property.

CHAPTER 3

BLOCK DIAGRAM AND DESCRIPTION

3.1 Block Diagram

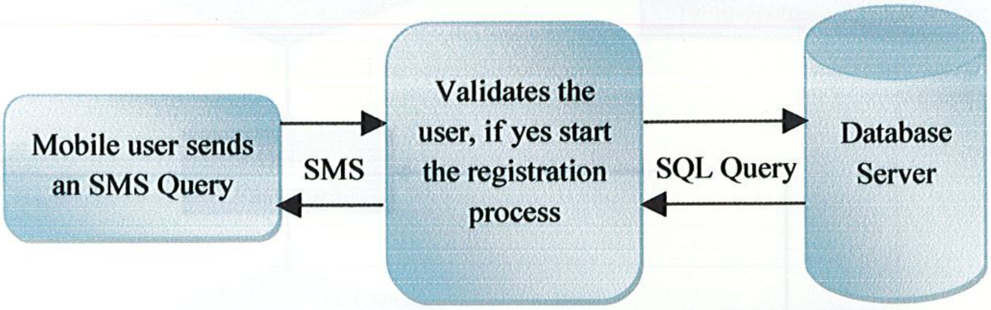
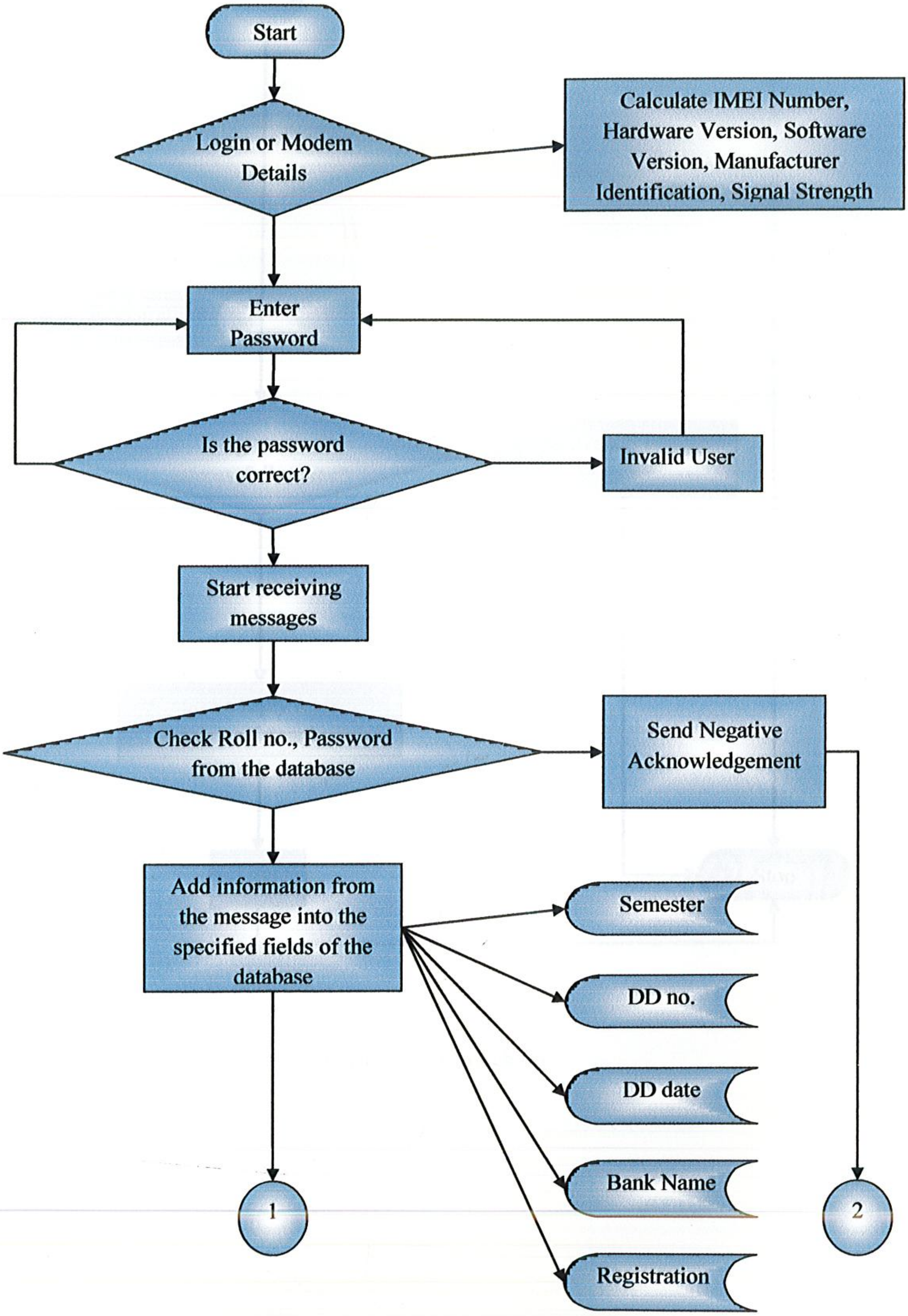

Mobile user sends an SMS Query
SMS
Validates the user, if yes start the registration process
SQL Query
Database Server

Figure 3.1 Functional Block Diagram of the system

Here is the functional block diagram of the system. The process begins with the student sending the request for the registration in the form of an SMS. The SMS has to be sent in a certain predefined format, which contains various details such as the student's enrollment number, password, semester, demand draft number, demand draft date, etc.

On receiving the SMS, the system first verifies the user's password. If the password is correct, then the registration process begins, else a negative acknowledgement is sent to the user. If the password is correct, the system breaks up the message and extracts various details from the message such as the enrollment number, demand draft number, demand draft date, etc. and sends them to the database server in the form of queries in order to update the required fields of the database. As soon as this process is completed, the system sends an acknowledgement to the user that his registration has been successfully completed.

3.2 Flowchart


Start
Login or Modem Details
Calculate IMEI Number, Hardware Version, Software Version, Manufacturer Identification, Signal Strength
Enter Password
Is the password correct?
Invalid User
Start receiving messages
Check Roll no., Password from the database
Send Negative Acknowledgement
Add information from the message into the specified fields of the database
Semester
DD no.
DD date
Bank Name
Registration
1
2

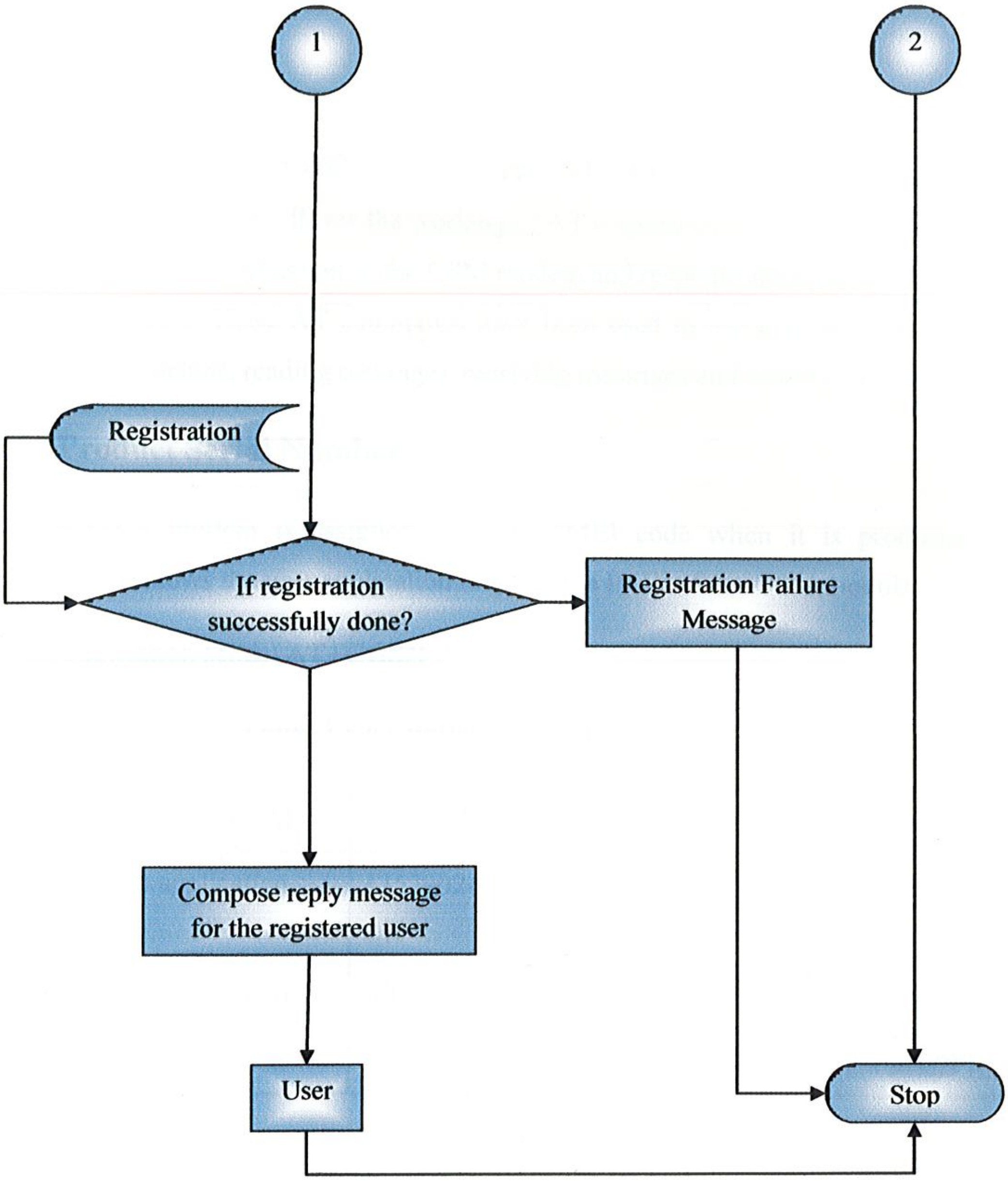

1
2
Registration
If registration successfully done?
Registration Failure Message
Compose reply message for the registered user
User
Stop

Figure 3.2 Flowchart

3.3 Operation

In order to control the GSM Modem with a PC, AT commands are required. In the following section we will see the working of AT commands in detail which will include request from the application to the GSM modem and response from the GSM modem to the application. These AT commands have been used in our application for checking the modem details, reading messages, receiving messages and sending messages.

- **Product Serial Number**

Each GSM modem is assigned a unique IMEI code when it is produced. This command allows the user application to know the IMEI of the GSM module.

The application sends: **AT+CGSN**

Table 3.1 Example for working of AT+CGSN

Application to GSM	AT+CGSN	request IMEI
GSM to application	135790248939 OK	IMEI present in E2PROM
Application to GSM	AT+CGSN	request IMEI
GSM to application	+CME ERROR: 22	IMEI present in E2PROM

- **Request model identification**

This command is used to get the hardware version of GSM Modem.

The application sends: **AT+CGMM**

Table 3.2 Example for working of AT+CGMM

Application to GSM	AT+CGMM	get hardware version
GSM to application	GSM P 900 OK	command valid

- **Request revision identification**

This command is used to get the software version of GSM Modem.

The application sends: **AT+CGMR**

Table 3.3 Example for working of AT+CGMR

Application to GSM	AT+CGMR	get software version
GSM to application	V2.74 OK	command valid

- **Manufacturer identification**

This command gives the manufacturer identification.

The application sends: **AT+CGMI**

Table 3.4 Example for working of AT+CGMI

Application to GSM	AT+CGMI	get manufacturer identification
GSM to application	WELCOME OK	command valid (« WELCOME » is not a manufacturer identification !)

- **Read message**

This command allows the application to read incoming stored messages. In our application we have made an interface in which user can read a particular message stored in the SIM by entering the message number.

The application sends: **AT+CMGR=<index>**

+CMGR=<stat>,<oa>,<scts>[,<tooa>,<fo>,<pid>,<dcs>,<sca>,<tosca>,<length >]<CR><LF><data>

Table 3.5 Example for working of AT+CGMR

GSM to application	+CMTI: "SM",1	New message received
Application to GSM	AT+CMGR=1	read the message
GSM to application	+CMGR: "REC UNREAD", "43322449"<CR> To be or not to be! OK	
Application to GSM	AT+CMGR=1	read again the message
GSM to application	+CMGR: "REC READ", "43322449",20<CR> To be or not to be! OK	
Application to GSM	AT+CMGR=2	read + bad index
GSM to application	+CMS ERROR: 32	error : invalid index
Application to GSM	AT+CMGR=1	in PDU mode
GSM to application	+CMGR: "REC READ", <length><CR><LF><pdu> OK	

- **New message indication**

This command selects the procedure of receiving the message from the network. In our application to start the registration process and in order to receive message from the students this command is used.

The application must send the following command:

AT+CNMI=<mode>,<mt>,<bm>,<ds>,<bfr>

<mode> (controls the processing of unsolicited result codes):

Table 3.6 Various options for AT+CNMI=<mode>

0	Buffer unsolicited result codes in the TA. If TA result code buffer is full, indications can be buffered in some other place or the oldest indications may be discarded and replaced with the new received indications.
1	Discard indication and reject new received message unsolicited result codes when TA-TE link is reserved. Otherwise forward them directly to the TE.
2	Buffer unsolicited result codes in the TA when TA-TE link is reserved and flush them to the TE after reservation. Otherwise forward them directly to the TE.
3	Forward unsolicited result codes directly to the TE. TA-TE link specific inband used to embed result codes and data when TA is in on-line data mode.

<mt> (sets the result code indication routing for SMS-DELIVERs):

Table 3.7 Various options for AT+CNMI=<mt>

0	no SMS-DELIVER indications are routed.
1	SMS-DELIVERs are routed using unsolicited code : +CMTI : « SM », <index>
2	SMS-DELIVERs (except class 2 messages) are routed using unsolicited code : +CMT : <pdu> (if PDU mode chosen) or +CMT : <oa>,<scts>[,<tooa>,<fo>,<pid>,<dcs>,<sca>,<tosca>,<length>]<CR><LF><data>
3	class 3 SMS-DELIVERS are routed directly using code in <mt>=2 ; message of other classes result in indication <mt>=1

<bm> (the rules for storing received CBMs Types depend on its coding scheme, the setting of Select CBM Types (+CSCB) and this value):

Table 3.8 Various options for AT+CNMI=<bm>

0	no CBM indications are routed to the TE.
1	If CBM is stored into ME/TA, indication of the memory location is routed to the TE using unsolicited result code : +CBMI : <mem>, <index>
2	New CBMs are routed directly to the TE using unsolicited result code. +CBM : <length><CR><LF><pdu> (if PDU mode chosen) or +CBM :<sn>,<mid>, <dcs>,<page>,<pages><CR><LF><data>(text mode-enabled)
3	class 3 SMS-DELIVERS are routed directly using code in <mt>=2 ; message of other classes result in indication <mt>=1

<ds> (for SMS-STATUS-REPORTs):

Table 3.9 Various options for AT+CNMI=<ds>

0	no SMS-STATUS-REPORTs are routed.
1	SMS-STATUS-REPORTs are routed using unsolicited code : +CDS : <pdu> (if PDU mode chosen) or +CDS : <fo>,<mr>[,<ra>,<tora>],<scts>,<dt>,<st>

<bfr>

Table 3.10 Various options for AT+CNMI=<bfr>

0	TA buffer of unsolicited result codes defined within this command is flushed to the TE when <mode> 1...3 is entered (OK response shall be given before flushing the codes)
1	TA buffer of unsolicited result codes defined within this command is cleared when <mode> 1...3 is entered.

Table 3.11 Example for working of AT+CNMI

Application to GSM	AT+CNMI=0,1,0,0,0	<mt>=1
GSM to application	OK	
Application to GSM	+CMTI : « SM », 1	Message received
GSM to application	AT+CNMI=0,2,0,0,0	
Application to GSM	OK	
GSM to application	+CMT :"123456","13/01/96 12h30m00s",129,4,32,240,"15379",129,5<CR><LF> HELLO	Message received
Application to GSM	AT+CNMI=0,0,0,1,0	<ds>=1
GSM to application	OK	

- **Send message**

This AT command is used for sending messages from PC to the user. In our application this command is used to send the acknowledgment message to the user. For this two cases are possible:

1) If the registration is completed then the message "Registration Successfully Completed" is sent.

2) If for any reason the registration cannot be done then the message "Registration Failed" is sent.

The <address> field is the address of the terminal network to which the message is sent; <exitmethod> is ^Z: ASCII 26. The text can contain all existing character except ^Z.

The application sends: **AT+CMGS="+33146290800"<CR>**

Table 3.12 Example for working of AT+CMGS

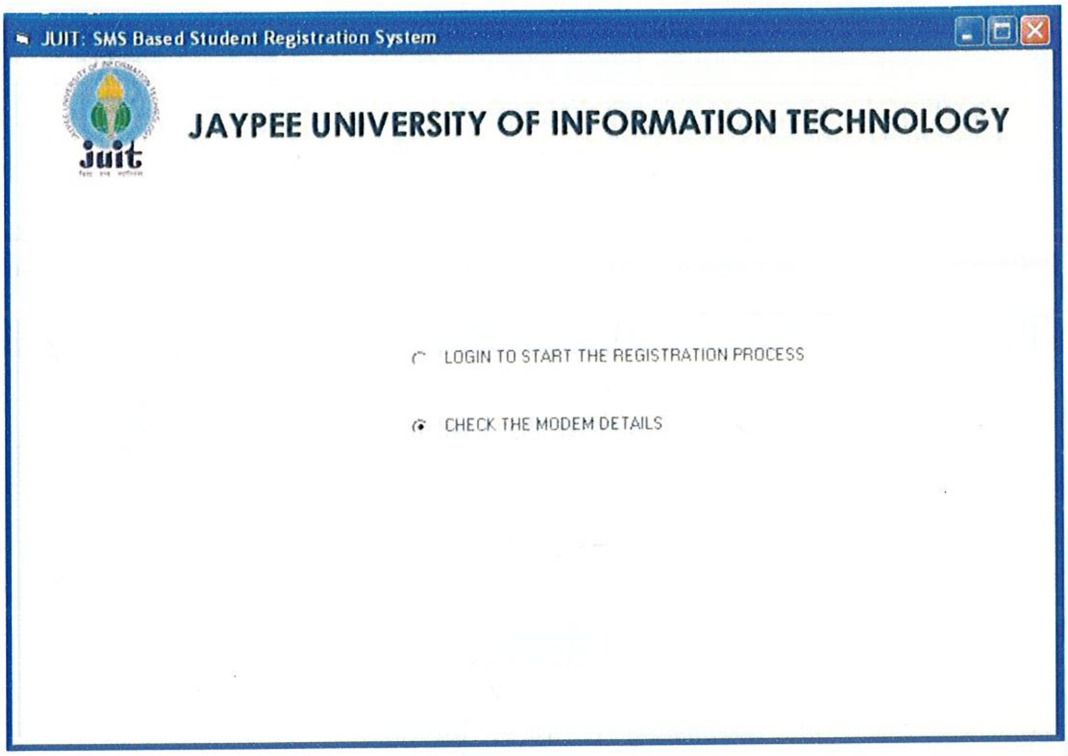
Application to GSM	AT+CMGS="+33146290800"<CR> Hello, how are you ?^Z	Send a message in text mode
GSM to application	+CMGS : <mr> OK	Successful transmission
Application to GSM	AT+CMGS=<length><CR> <pdu>^Z	Send a message in PDU mode
GSM to application	+CMGS : <mr> OK	Successful transmission

The message reference <mr> which is returned back to the application is allocated by the GSM module. This number begins with 0 and is incremented by one for each outgoing message (successful and failure case); it is cyclic on one byte (0 follows 255).

CHAPTER 4

RESULTS

When the user will start the application, he will see the following window. He would need to choose whether he wants to check the modem details, its various parameters like the signal strength, IMEI number, manufacturer identification, hardware and software version, etc, or whether he wants to begin the registration process.


JUIT: SMS Based Student Registration System

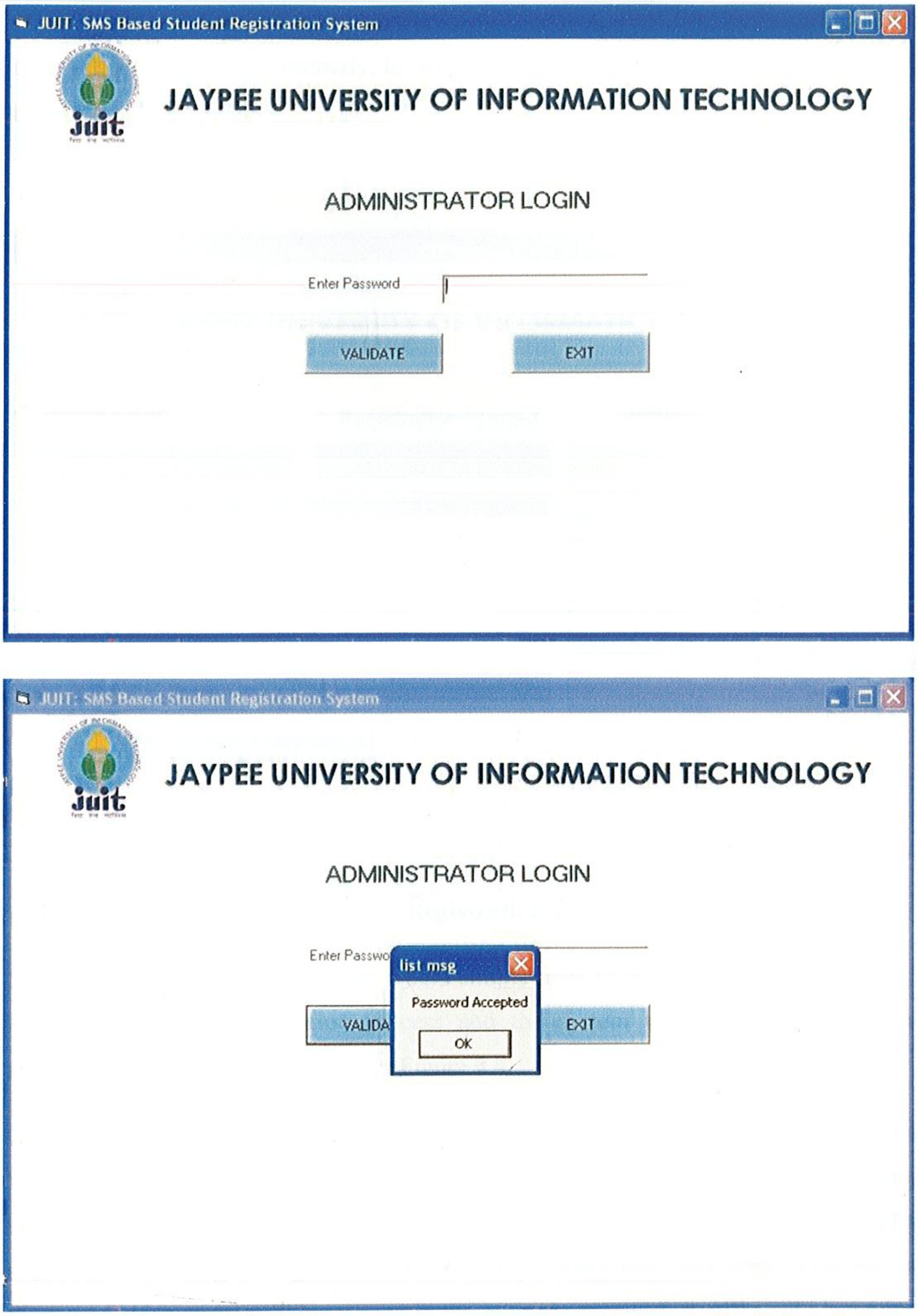
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

○ LOGIN TO START THE REGISTRATION PROCESS

● CHECK THE MODEM DETAILS

Figure 4.1 Start-up window

On clicking the option to login and start the registration process, he would be asked to enter the administrator password for the application, as shown in Figure 4.2. On entering the correct password, the registration window will open. If the password entered is wrong, the process will not start.



JUIT: SMS Based Student Registration System
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
ADMINISTRATOR LOGIN
Enter Password
VALIDATE
EXIT
JUIT: SMS Based Student Registration System
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
ADMINISTRATOR LOGIN
Enter Passwo
VALIDA
EXIT

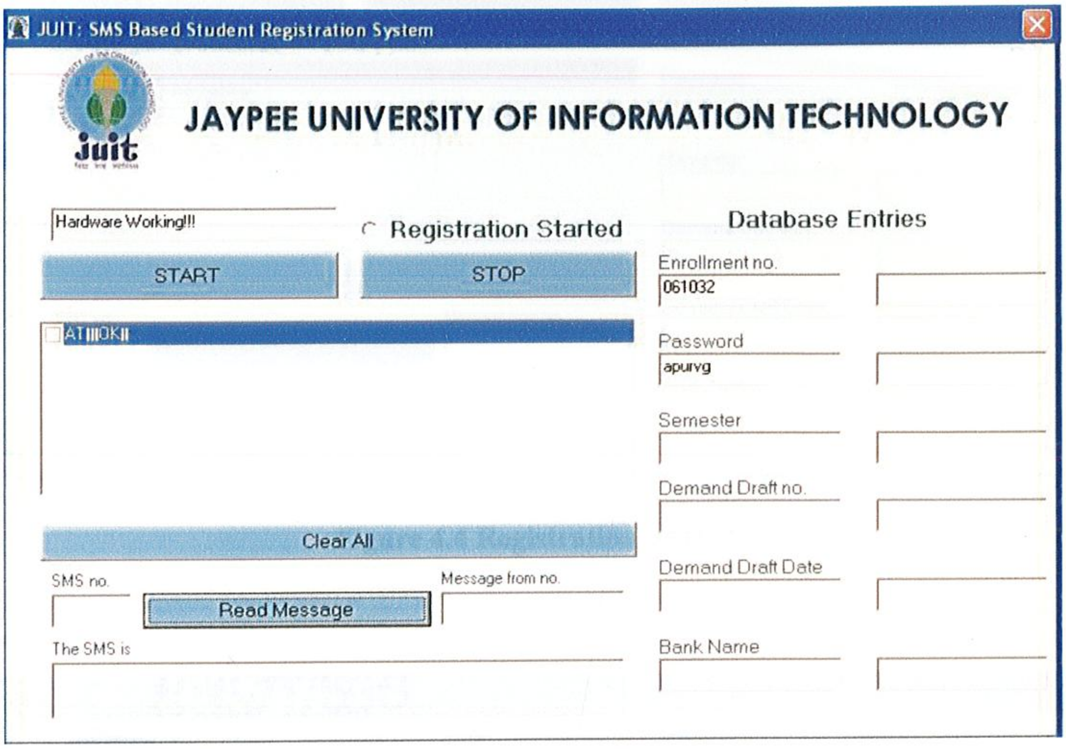
list msg

Password Accepted

OK

Figure 4.2 Administrator login window

Once the user has entered the correct password, he would see the following window. If the modem is working properly, he would see a message, "Hardware Working!!!", as shown in Figure 4.3.



JUIT: SMS Based Student Registration System

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

Hardware Working!!!

○ Registration Started

AT|||OK||

Database Entries

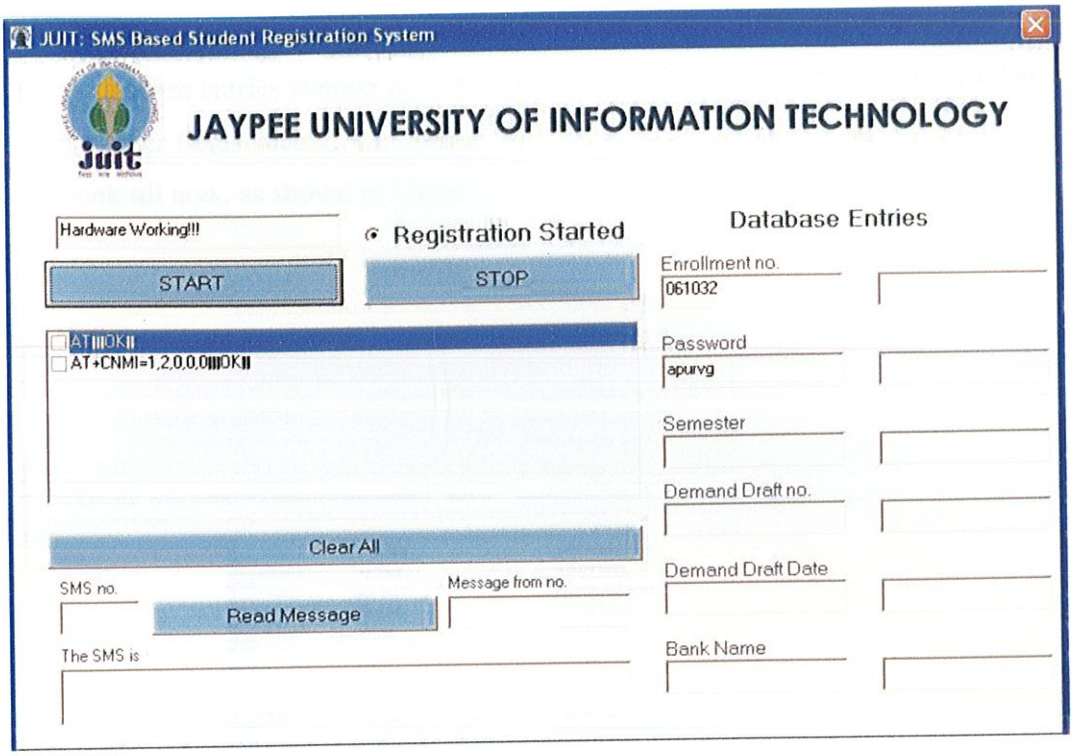
Enrollment no. 061032

Password apurvg

Figure 4.3 Registration window

To start the registration process, he would simply need to click on the START button. This will start the registration process and the system will start receiving new messages automatically, as shown in Figure 4.4.

On receiving a new message, the system will read the contents of the SMS. If the SMS is in the correct syntax, i.e., "Roll no., Password, Semester, DD no., DD date, bank name", the system will make the appropriate changes in the database, and send a confirmation message, "Registration Successfully Completed", as a reply, as shown in Figure 4.5. If there is anything wrong in the message received, it will send a negative acknowledgement, "Registration Failed", as a reply. To quit the application, the user has to simply press the STOP button.


JUIT: SMS Based Student Registration System
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
START
STOP
Clear All
SMS no.
Read Message
Message from no.
The SMS is
Database Entries

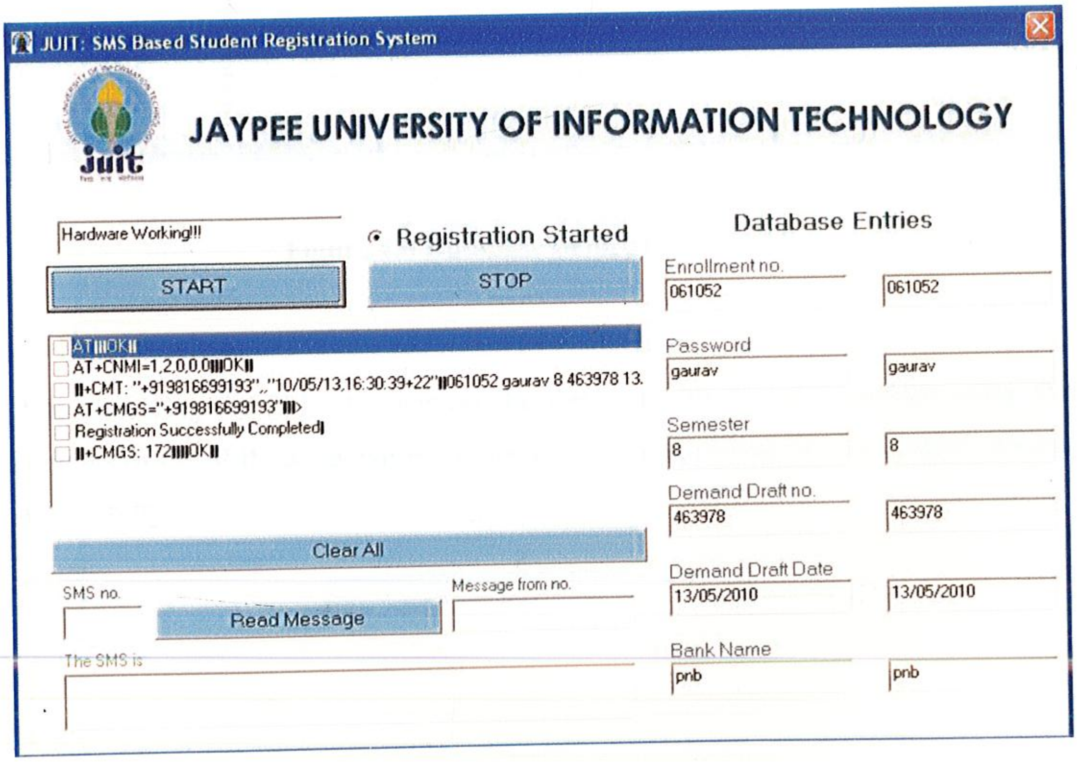
Hardware Working!!!

● Registration Started

```
AT▮▮▮OK▮▮
AT+CNMI=1,2,0,0,0▮▮▮OK▮▮
```

Enrollment no.	061032
Password	apurvg
Semester	
Demand Draft no.	
Demand Draft Date	
Bank Name	

Figure 4.4 Registration started


JUIT: SMS Based Student Registration System
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
START
STOP
Clear All
SMS no.
Read Message
Message from no.
The SMS is
Database Entries

Hardware Working!!!

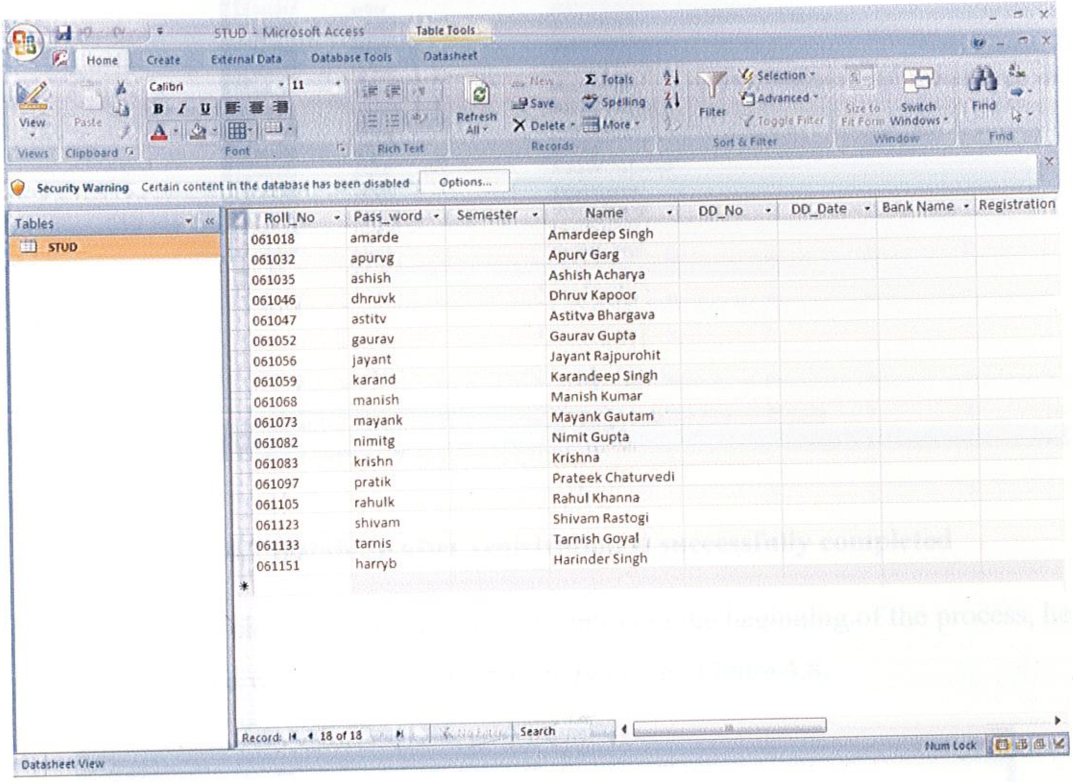
● Registration Started

```
AT▮▮▮OK▮▮
AT+CNMI=1,2,0,0,0▮▮▮OK▮▮
▮▮+CMT: "+919816699193",,"10/05/13,16:30:39+22"▮▮061052 gaurav 8 463978 13.
AT+CMGS="+919816699193"▮▮▮>
Registration Successfully Completed▮
▮▮+CMGS: 172▮▮▮▮OK▮▮
```

Enrollment no.	061052	061052
Password	gaurav	gaurav
Semester	8	8
Demand Draft no.	463978	463978
Demand Draft Date	13/05/2010	13/05/2010
Bank Name	pnb	pnb

Figure 4.5 Registration successfully completed

Here is a preview of the database before the registration process has started. The initial database entries contain the roll number, name and password of all the students. All the other fields such as DD number, DD date, bank name, registration status, etc. are blank till now, as shown in Figure 4.6.

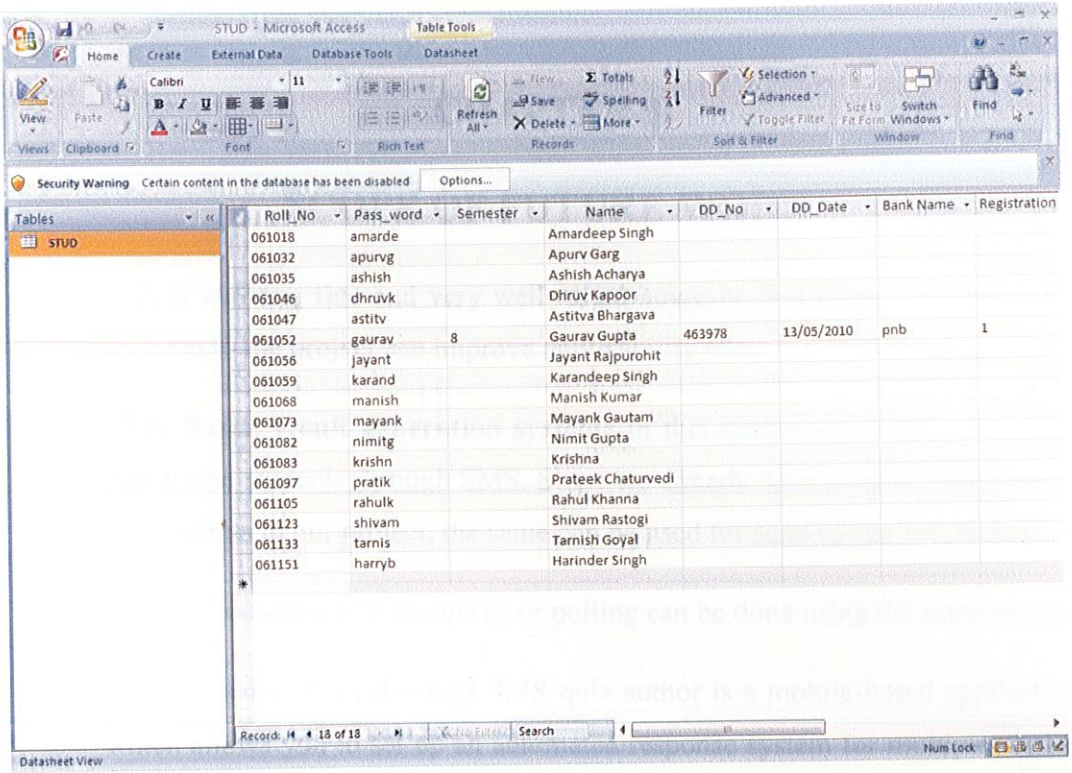


Security Warning Certain content in the database has been disabled Options...

Roll_No	Pass_word	Semester	Name	DD_No	DD_Date	Bank Name	Registration
061018	amarde		Amardeep Singh				
061032	apurvg		Apurv Garg				
061035	ashish		Ashish Acharya				
061046	dhruvk		Dhruv Kapoor				
061047	astitv		Astitva Bhargava				
061052	gaurav		Gaurav Gupta				
061056	jayant		Jayant Rajpurohit				
061059	karand		Karandeep Singh				
061068	manish		Manish Kumar				
061073	mayank		Mayank Gautam				
061082	nimitg		Nimit Gupta				
061083	krishn		Krishna				
061097	pratik		Prateek Chaturvedi				
061105	rahulk		Rahul Khanna				
061123	shivam		Shivam Rastogi				
061133	tarnis		Tarnish Goyal				
061151	harryb		Harinder Singh				

Figure 4.6 Database before registration

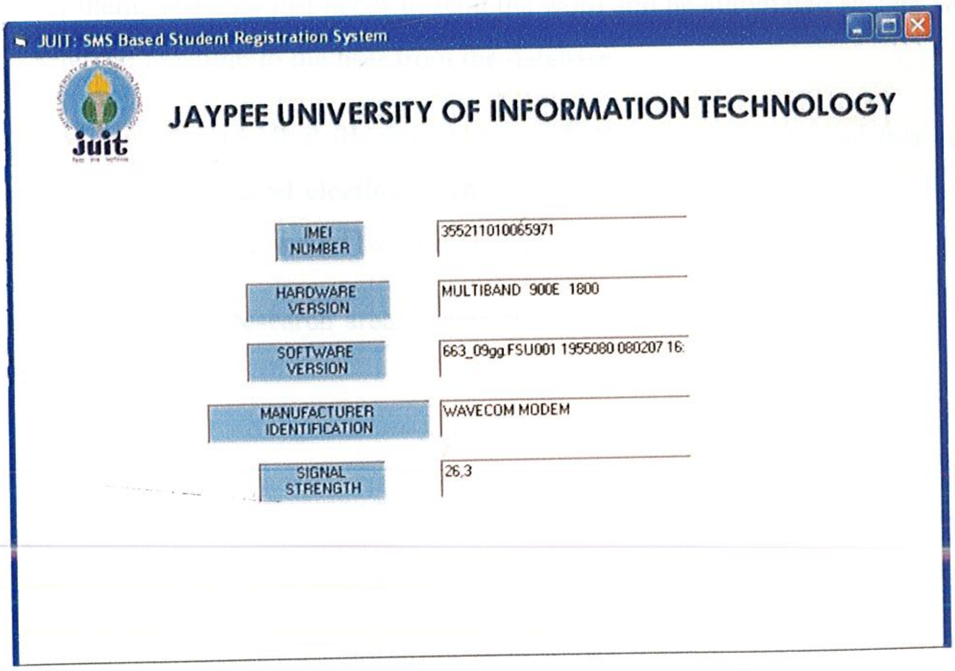
After receiving the SMS, the appropriate fields are filled in the database using the contents of the SMS, as shown in Figure 4.7, thus completing the registration process successfully.



Roll_No	Pass_word	Semester	Name	DD_No	DD_Date	Bank Name	Registration
061018	amarde		Amardeep Singh				
061032	apurvg		Apurv Garg				
061035	ashish		Ashish Acharya				
061046	dhruvk		Dhruv Kapoor				
061047	astitv		Astitva Bhargava				
061052	gaurav	8	Gaurav Gupta	463978	13/05/2010	pnb	1
061056	jayant		Jayant Rajpurohit				
061059	karand		Karandeep Singh				
061068	manish		Manish Kumar				
061073	mayank		Mayank Gautam				
061082	nimitg		Nimit Gupta				
061083	krishn		Krishna				
061097	pratik		Prateek Chaturvedi				
061105	rahulk		Rahul Khanna				
061123	shivam		Shivam Rastogi				
061133	tarnis		Tarnish Goyal				
061151	harryb		Harinder Singh				

Figure 4.7 Database after registration is successfully completed

Had the user had clicked the modem details option in the beginning of the process, he would have seen a window similar to that shown in the Figure 4.8.


JUIT: SMS Based Student Registration System
juit
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

IMEI NUMBER: 355211010065971

HARDWARE VERSION: MULTIBAND 900E 1800

SOFTWARE VERSION: 663_09gg FSU001 1955080 080207 16:

MANUFACTURER IDENTIFICATION: WAVECOM MODEM

SIGNAL STRENGTH: 26,3

Figure 4.8 Modem details

CHAPTER 5

SCOPE OF FUTURE WORK

The project is working fine and very well tested however following additions at the university level to the project can improve applicability of it:

- **SMS Based result generation system:** In this system the results would be sent to the students through SMS. Since we already have used the database of the students in our project, the same can be used for sending the results also.
- **SMS Based survey**: Any survey or polling can be done using the same model.
- **SMS Based quiz authoring:** SMS quiz author is a mobile-based application which allows you to set up an automated response system for multiple-choice quizzes on the learner's mobile phone. Participants answer questions by text and receive instant feedback.
- **SMS Based dictionary system:** Since in the university there are many aspiring candidates for competitive exams, this system would be very helpful for them. The user just needs to send the word and he automated system would send the meaning to the user from the database.

These applications were all at the university level. Now at national level this model can be used for **SMS based election contesting,** which would replace the traditional voting system, and in turn would be more transparent, reliable, and secure.

Hence there are many research areas where the same model can be used to ease the process.

CHAPTER 6
CONCLUSION

The main objective of our project was to design an efficacious model for the registration process at our university. In this project a low cost, secure, ubiquitously accessible, auto-configurable, remotely controlled solution for registration has been introduced. The approach discussed and implemented in the project is novel and has successfully achieved the target to enable the student to remotely register using the SMS based system thus satisfying the student's needs and requirements.

During tests, the full functionality of the system was checked and after going through the various modules, we found that our system comprises of the following four features, namely **security, cost, time,** and **ease of access.**

The following benefits are offered by our system:

- **Security** → The users are expected to acquire login and password to access the system. This adds protection from unauthorized access.
- **Cost efficient** → In the present day SMS is cheapest method of communication. Most network providers are continuously providing schemes for free SMS or at a very reasonable rate.
- **Time efficient** → It is highly time efficient as the student just has to send the SMS request and the rest of the process is fully automated.
- **Accessibility** → All students have access to services without being physically present at the university just with the help of a mobile phone.
- **Reliability** → The system performs dependably, accurately, and consistently.
- **Accuracy** → The implemented system is more accurate, minimal error possible.
- **Usability** → The system is easier and more convenient to use.

It can be seen that the project is developed in such a way that it is also advantageous for the university. With the present traditional system a lot of manpower is needed to register the students and a lot of time is consumed each year during the registration. But now the students and the university can both save a lot of time by completing the registration process using the given system.

References

Books:

- Steven Holzner, *Visual Basic 6 Black Book,* Paraglyph Press, August 1998

- Building Wireless Sensor Networks by Robert Faludi, O'Reilly Media, Inc, April 2010

Research Papers:

- SMS Based Wireless Home Appliance Control System(HACS) for Automating Appliances and Security by *Malik Sikandar Hayat Khiyal, Aihab Khan, and Erum Shehzadi Software Engineering Dept., Fatima Jinnah Women University, Rawalpindi, Pakistan*

- A Six-Level Model of SMS-based eGovernment by *Tony Dwi Susanto, Dr. Robert Goodwin, A/Prof. Paul Calder School of Informatics and Engineering-Flinders University, South Australia*

- The Sms Based Forecasting System Of Public Transport Arrival Time by *Fong Yit Meng Yee Chee Hong Yong Huey Yi*

- An SMS Based Querying System for Mobile Learning by *Dunwei Xiong Ally Lin.*

- Secure Mobile Based Voting System by *Manish Kumar1*, T.V.Suresh Kumar1, M. Hanumanthappa2, D Evangelin Geetha1*

- SMS based systems for monitoring of Polling Stations: Toward improving Electoral System while considering the range of technical and societal challenges by Sanjeev Ranjan, Chief Electoral Officer, Tripura.

- A Six-Level Model of SMS-based eGovernment by *Tony Dwi Susanto, Dr. Robert Goodwin, A/Prof. Paul Calder*

Web Pages:

- http://www.bitwisemag.com/index.html
- http://www.ozeki.hu
- http://www.activexperts.com/files/mmserver/manual.htm#intro
- http://www.wavecom.com/modules/movie/scenes/products
- http://www.developershome.com/sms/
- http://www.ravirajtech.com/product.html
- http://msdn.microsoft.com/en-in/library/aa231237(v=VS.60).aspx
- http://www.ringtones-central.com/what-is-sms.htm
- http://www.dynamax.com/GSM.htm
- http://www.nowsms.com
- http://www.nextbus.com

Appendix A

Program 1: Program for selecting the appropriate option

```
Private Sub Label1_Click()
        Form3.Show
        Option1.Value = True
        Unload Me
End Sub
Private Sub Label2_Click()
        Form4.Show
        Option1.Value = True
        Unload Me
End Sub
Private Sub Label3_Click()
        Form4.Show
        Option1.Value = True
        Unload Me
End Sub
```

Program 2: Program for validating the administrator login

```
Private Sub Command1_Click()
        Dim strpass$
        strpass = "juit"
        Dim intctr As Integer
        intctr = 1
        Dim blnvalid As Boolean
        Do While Not blnvalid
        If Text1.Text < > strpass Then
                MsgBox "INVALID PASSWORD"
                intctr = intctr + 1
                        If intctr = 3 Then
                        MsgBox "UNAUTHORIZED USER... Please enter a valid
                        Password"
                Exit Do
                Text1.SetFocus
        End If
        Else
        blnvalid = True
        MsgBox "Password Accepted"
        Unload Me
        Form1.Show
End If
Loop
End Sub

Private Sub Command2_Click()
        Unload Me
End Sub
```

Program 3: Program for the Student Registration Process

```
Dim prtData, Msg, Emsg, Nmsg As String
Dim PrtDetect, i, l As Integer
Dim HrdChk As Boolean
Dim NewMsg As Boolean
Dim ExtMsg As Boolean
Dim sendmsg As Boolean
Dim sendnsg As Boolean
Dim smsg As Boolean
Dim auto As Boolean
Dim adod As Boolean
Dim loc As Integer
Dim x As Integer
Dim y As String
Dim d As Long

Private Sub Command2_Click()
        List1.Clear
        Text1.Text = ""
        Text2.Text = ""
        Text3.Text = ""
        Text4.Text = ""
        Text5.Text = ""
        Text6.Text = ""
        Text7.Text = ""
        Text8.Text = ""
        Text9.Text = ""
        Text10.Text = ""
        Text11.Text = ""
        Text12.Text = ""
        Text13.Text = ""
        Text14.Text = ""
        Text15.Text = ""
        Text16.Text = ""
        Text17.Text = ""
        Text18.Text = ""
        Text19.Text = ""
        Text20.Text = ""
End Sub
Private Sub Command3_Click()
        ReadData "AT+CMGR=" & Text2.Text
        ExtMsg = True
End Sub
Private Sub Command4_Click()
        WriteData "AT"
        HrdChk = True
```

```
End Sub
Private Sub Command5_Click()
        'SendData "AT+CMGS=" & "+91" & Text10.Text
        sendmsg = True
End Sub
Private Sub Command6_Click()
        Unload Me
End Sub
Private Sub Form_Load()
        Adodc1.ConnectionString = "Provider=Microsoft.jet.oledb.4.0;persist security
        info=false;data source=" & App.Path & "\Payroll.mdb"
        Adodc1.RecordSource = "select * from STUD"
        Adodc1.Refresh
        PrtDetect = 1
        sel:
        Select Case PrtDetect
                Case 1:
                        MSComm.CommPort = 1
                        On Error Resume Next
                        MSComm.PortOpen = True
                        GoTo main
                Case 2:
                        MSComm.CommPort = 2
                        On Error Resume Next
                        MSComm.PortOpen = True
                        GoTo main
                Case 3:
                        MSComm.CommPort = 3
                        On Error Resume Next
                        MSComm.PortOpen = True
                        GoTo main
                Case 4:
                        MSComm.CommPort = 4
                        On Error Resume Next
        MSComm.PortOpen = True
        GoTo main
Case 5:
        MSComm.CommPort = 5
        On Error Resume Next
        MSComm.PortOpen = True
        GoTo main
Case 6:
        MSComm.CommPort = 6
        On Error Resume Next
        MSComm.PortOpen = True
        GoTo main
Case 7:
        MSComm.CommPort = 7
        On Error Resume Next
        MSComm.PortOpen = True
        GoTo main
Case 8:
        MSComm.CommPort = 8
        On Error Resume Next
        MSComm.PortOpen = True
        GoTo main
Case 9:
        MSComm.CommPort = 9
        On Error Resume Next
        MSComm.PortOpen = True
        GoTo main
Case Else:
        End Select
        WriteData "AT"
        Exit Sub
main:
PrtDetect = PrtDetect + 1
GoTo sel
```

```
End Sub
Private Sub Timer1_Timer()
        If MSComm.PortOpen = True Then
        prtData = MSComm.Input
        If Mid(prtData, 6, 2) = "OK" Then
                Text1.Text = "Hardware Working!!!"
                HrdChk = False
        End If
        If Len(prtData) > 0 Then
                If Right(prtData, 2) = Chr(62) & Chr(32) Then
                        If sendmsg Then
                        MSComm.Output = "Registration Successfully Completed" &
Chr(26)
                        sendmsg = False
                        End If
                End If
                If Right(prtData, 2) = Chr(62) & Chr(32) Then
                        If sendnsg Then
                        MSComm.Output = "Registration Failed" & Chr(26)
                        sendnsg = False
                        End If
                End If
        List1.AddItem prtData
        ' Text14.Text = Len(prtData)

        End If
End If
'Timer1.Enabled = False
        If HrdChk = True Then
                If Mid(prtData, 6, 2) = "OK" Then
                        Text1.Text = "Hardware Working!!!"
                        HrdChk = False
                Else
                Text1.Text = "Please Check Hardware!"
                HrdChk = False
                End If
        End If
'Timer1.Enabled = True
        If NewMsg = True Then
                On Error Resume Next
                MsgBox ("Message from number" & Mid(prtData, 33, 13))
                NewMsg = False
        End If
        If ExtMsg = True Then
                i = Len(prtData) - 80
                On Error Resume Next
                Text6.Text = Mid(prtData, 70, i)
                On Error Resume Next
                Text3.Text = Mid(prtData, 33, 12)
                ExtMsg = False
                Exit Sub
        End If
        If auto = True Then
                'prtData = x
                If Len(prtData) > 24 Then
                Text4.Text = Mid(prtData, 13, 10)
                Text10.Text = Text4.Text
                ElseIf Len(prtData) = 0 Then
                Text4.Text = Text4.Text
                Else
                Text4.Text = ""
                End If


        If Len(prtData) = 88 And Mid(prtData, 3, 5) = "+CMT:" Then
                Text15.Text = Mid(prtData, 50, 6)
                Text16.Text = Mid(prtData, 57, 6)
                Text17.Text = Mid(prtData, 64, 1)
```

```
    Text18.Text = Mid(prtData, 66, 6)
    Text19.Text = Mid(prtData, 73, 10)
    Text20.Text = Mid(prtData, 84, 3)
If Adodc1.Recordset.RecordCount <> 0 Then
    Adodc1.Recordset.MoveLast
    Adodc1.Recordset.MoveFirst
    Adodc1.Recordset.Find "Roll_No='" & Text15.Text & "'"
    If Adodc1.Recordset!Roll_No = Text15.Text And
        Adodc1.Recordset!Pass_word = Text16.Text And
        IsNull(Adodc1.Recordset!Registration) = True Then
        If Adodc1.Recordset.EOF = False Then
            Adodc1.Recordset!Semester = Text17.Text
            Adodc1.Recordset!DD_No = Text18.Text
            Adodc1.Recordset!DD_Date = Text19.Text
            Adodc1.Recordset![Bank Name] = Text20.Text
            Adodc1.Recordset!Registration = "1"
            Adodc1.Recordset.Update
            sendmsg = True
        End If
    Else
        sendnsg = True
    End If

 End If
l = Len(prtData) - 49
On Error Resume Next
Text5.Text = "received msg is" & Mid(prtData, 47, l)
ElseIf Len(prtData) > 24 And Len(prtData) <> 88 Then
x = Len(prtData)
'y = 88 - x
Text14.Text = Text14.Text & Mid(prtData, 1, 88 - x)
y = Text14.Text
If Len(y) = 88 And Mid(y, 3, 5) = "+CMT:" Then
Text15.Text = Mid(y, 50, 6)
```

```
    Text16.Text = Mid(y, 57, 6)
    Text17.Text = Mid(y, 64, 1)
    Text18.Text = Mid(y, 66, 6)
    Text19.Text = Mid(y, 73, 10)
    Text20.Text = Mid(y, 84, 3)
    If Adodc1.Recordset.RecordCount <> 0 Then
        Adodc1.Recordset.MoveLast
        Adodc1.Recordset.MoveFirst
        Adodc1.Recordset.Find "Roll_No='" & Text15.Text & "'"
        If Adodc1.Recordset!Roll_No = Text15.Text And
        Adodc1.Recordset!Pass_word = Text16.Text And
        IsNull(Adodc1.Recordset!Registration) = True Then
        If Adodc1.Recordset.EOF = False Then
            Adodc1.Recordset!Semester = Text17.Text
            Adodc1.Recordset!DD_No = Text18.Text
            Adodc1.Recordset!DD_Date = Text19.Text
            Adodc1.Recordset![Bank Name] = Text20.Text
            Adodc1.Recordset!Registration = "1"
            Adodc1.Recordset.Update
            sendmsg = True
        End If
        Else
    End If
End If
End If
End If
End If
If sendnsg = True Then
MSComm.Output = "AT+CMGS=" & Chr(34) & "+91" & Text10.Text & Chr(34) &
Chr(13)
Exit Sub
End If
If sendmsg = True Then
```

```
MSComm.Output = "AT+CMGS=" & Chr(34) & "+91" & Text10.Text & Chr(34) &
Chr(13)
Exit Sub
End If
End Sub

Private Sub Command1_Click()
        WriteData "AT+CNMI=1,2,0,0,0"
        auto = True
        Option1.Value = True
End Sub

Private Sub WriteData(Dta As String)
        MSComm.Output = Dta
        MSComm.Output = Chr(13)
End Sub

Private Sub ReadData(Rdt As String)
        MSComm.Output = Rdt
        MSComm.Output = Chr(13)
End Sub

Private Sub SendData(Sdt As String)
        MSComm.Output = Sdt
        MSComm.Output = Chr(13)
End Sub
```

Program 4: Program for checking the GSM modem details

```
Dim PrtDetect As Integer
Dim prtData As String
Dim ss As Boolean
Dim imei As Boolean
Dim hrd As Boolean
Dim soft As Boolean
Dim mi As Boolean
Dim HrdChk As Boolean
Dim adod As Boolean
Private Sub Form_Load()
        PrtDetect = 1
        sel:
        Select Case PrtDetect
                Case 1:
                        MSComm.CommPort = 1
                        On Error Resume Next
                        MSComm.PortOpen = True
                        GoTo main
                Case 2:
                        MSComm.CommPort = 2
                        On Error Resume Next
                        MSComm.PortOpen = True
                        GoTo main
                Case 3:
                        MSComm.CommPort = 3
                        On Error Resume Next
                        MSComm.PortOpen = True
                        GoTo main
                Case 4:
                        MSComm.CommPort = 4
                        On Error Resume Next
                        MSComm.PortOpen = True
```

```
            GoTo main
        Case 5:
            MSComm.CommPort = 5
            On Error Resume Next
            MSComm.PortOpen = True
            GoTo main
        Case 6:
            MSComm.CommPort = 6
            On Error Resume Next
            MSComm.PortOpen = True
            GoTo main
        Case 7:
            MSComm.CommPort = 7
            On Error Resume Next
            MSComm.PortOpen = True
            GoTo main
        Case 8:
            MSComm.CommPort = 8
            On Error Resume Next
            MSComm.PortOpen = True
            GoTo main
        Case 9:
            MSComm.CommPort = 9
            On Error Resume Next
            MSComm.PortOpen = True
            GoTo main
        Case Else:
            End Select
            Exit Sub
        main:
        PrtDetect = PrtDetect + 1
        GoTo sel
End Sub
```

```
Private Sub Label2_Click()
        WriteData "AT+CGMM"
        hrd = True
End Sub

Private Sub Label3_Click()
        WriteData "AT+CGMR"
        soft = True
End Sub
Private Sub Label4_Click()
        WriteData "AT+CGMI"
        mi = True
End Sub

Private Sub Label5_Click()
        WriteData "AT+CSQ"
        ss = True
End Sub

Private Sub Label1_Click()
        WriteData "AT+CGSN"
        imei = True
End Sub

Private Sub Label6_Click()
        WriteData "AT+CNUM"
        msisdn = True
End Sub

Private Sub Timer1_Timer()
        If MSComm.PortOpen = True Then
                prtData = MSComm.Input
                If Len(prtData) > 0 Then
                        If ss = True Then
                                Text6.Text = Mid(prtData, 16, 4)
                                ss = False
                        End If
                        If imei = True Then
                                Text1.Text = Mid(prtData, 11, 15)
                                imei = False
                        End If
                        If hrd = True Then
                                Text3.Text = Mid(prtData, 12, 21)
                                hrd = False
                        End If
                        If soft = True Then
                                Text4.Text = Mid(prtData, 11, 36)
                                soft = False
                        End If
                        If mi = True Then
                                Text5.Text = Mid(prtData, 12, 13)
                                mi = False
                        End If
                End If
        End If
End Sub

Private Sub WriteData(Dta As String)
        MSComm.Output = Dta
        MSComm.Output = Chr(13)
End Sub
```