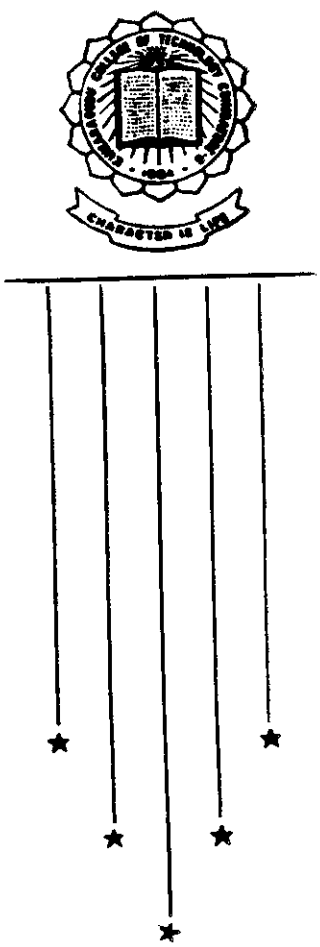
MEDIATALK

PROJECT REPORT



Submitted by

M. NAVIN SAMUEL

SUJARITHA JAYACHANDRAN

N. RADHIKA

V. SRIRAM

Guided by

Prof. M. RAMASAMY

M.E., MIEEE (U.S.A.), MIE., MISTE., C.Eng (I).

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING IN

ELECTRONICS & COMMUNICATOIN ENGINEERING

OF THE BHARATHIAR UNIVERSITY

1997 - 98

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Kumaraguru College of Technology

COIMBATORE - 641 006.

Department of Electronics and Communication Engineering
Kumaraguru College of Technology
Coimbatore - 641 006

CERTIFICATE

This is to certify that this project titled

MEDIATALK

has been submitted by

Mr/Ms _____ P-1341

In partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF ENGINEERING

in **Electronics and Communication Engineering**

of Bharathiar University, Coimbatore - 641 046

during the academic year 1997-1998

_____ GUIDE

_____ HEAD OF THE DEPARTMENT

certified that the candidate was examined by us in the project work

Viva-voce held on _____ and the

University Register Number is _____

_____ INTERNAL EXAMINER

_____ EXTERNAL EXAMINER

Kovai TDI Software Private Limited

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Mr.M.Navin Samuel, Ms.N.Radhika, Mr.V.Sriram, Ms.Sujaritha Jayachandran of B.E. (Electronics and Communication Engineering), Kumaraguru College of Technology have undergone their project work in our organisation from July '97 to March '98. The title of the project is "Mediatalk". The system has been developed using Visual C++ under Windows 95 operating system. Due to proprietory reasons, the candidates are not allowed to use the source code in any form outside KovaiTDI Software Pvt Ltd premises. Also, the technical ideas and design information of this project should be strictly used only for academic purposes with the prior permission from Kovai TDI Software Pvt Ltd, Coimbatore.

We are pleased to certify that they have successfully completed their project on time to meet our requirements. During the period of the project work, their conduct was good and we wish them all success in future endeavours.

Date: April 9, 1998

Place: Coimbatore

SIEMA Building, 8/4, Race Course Road, Coimbatore - 641 018.
INDIA. Tel : +91 (422) 200849 Fax : +91 (422) 214074
e-mail : kovaitdi@giasmd01.vsnl.net.in

TABLE OF CONTENTS

ACKNOWLEDGMENT

This project work has helped us to realize how massive an effort it really is and how much we had to rely on the selfless effort and goodwill of others. There were many who helped us with this project, and we would like to thank them in print. Without their timely help and effort, this project might still be in preparation.

Our sincere thanks goes to **Prof. Muthuraman Ramasamy** , M.E., MISTE., MIEEE(USA)., MIE., C ENG (I)., Professor and Head of the Department of Electronics and Communication Engineering for his valuable advice and timely guidance. We express our deep sense of gratitude to him, for all the help he has rendered throughout our studies and also in this project work.

We owe a debt of thanks to **Mr. V.Paramasivam** M.E., Managing Director, Kovai TDI Software Private Limited, Coimbatore , for his guidance in developing, shaping and reshaping this project work. It was his support and analytical planning schedule, that enabled this project to proceed within the stipulated schedule without much hiccups. I once again thank him for allowing us to take up a project in their esteemed organization.

We are also grateful to **Mr.D.Easwaramoorthy** our project guide, **Ms. G. Sangeetha** and **Ms. N. Uma Maheswarri** of Kovai TDI software for all the support that they had provided throughout the completion of this project.

Words are not enough to express our gratitude to all the faculty members , non-teaching staff and our friends, especially **B.R. Balaji** , who have been invaluable to us.

SYNOPSIS

This project titled " Mediatalk " enables telephone conversation through the Multimedia microphone and speaker and thereby completely eliminates the telephone handset from the user's desk. The application has been developed using Microsoft Visual C++ with the support of the Telephony Application Programmer's Interface (TAPI) . The project has been sponsored by Kovai TDI Softwares (P) Ltd., Coimbatore.

This application is a complete single-line telephone device thats runs on a Windows workstation. The application can run with minimum hardware requirements - a voice-data modem , sound card and a multimedia kit.

The project has been divided into four phases based on common functions and requirements . The First phase comprises of linking phone lines with the PC. The Second phase involves linking of the telephone lines with the sound card and the modem. The Third phase consists of creating and maintaining a database. The Fourth and final phase entails the user interface part.

1.1 EXISTING TELEPHONE SYSTEMS

The Terminal

The terminals of the present telephone network are the familiar telephone instruments. There are many different styles of instruments throughout the world although they are basically similar. A simplified circuit of the transmission elements of a telephone terminal is shown in the adjoining figure (fig.1). The microphone is almost always a carbon microphone and therefore needs a direct current to power it. This is normally provided by a d.c. power supply ("central battery" in the common switching centre). The d.c. is fed through a feeding bridge (shown in fig.2). The feeding bridge is a filter which separates the d.c. or a zero hertz signal from a.c. The d.c. is fed to the terminal but the audio frequencies are passed in both directions.

A transformer is used in the terminal to match the low impedance of the microphone to the line and to reduce "sidetone". Sidetone is the user's own voice caused by speech energy from the transmitter reaching the receiver of the same instrument.

Local Signaling

The terminal is usually connected to the local switching centre by a pair of wires which is electrically balanced about earth and carries the speech in both directions. The balancing prevents interference between telephones whose pairs are adjacent to one another in cables or on telegraph poles. The effect of induced currents and other forms of interference are also minimized. This pair of wires is called the subscriber's loop or local

line.

The control signals required for the remote control of the switching system are commonly transmitted on the same pairs of wires as the information. This saves transmission cost but requires suitable machinery to extract the signaling information received from the terminal and insert the signaling information to be sent to the terminal.

The basic signals of seize, answer and clear are provided by changes of the d.c. resistance connected across the pair of wires at the terminal. When the terminal is not in use , the handset rests on the cradle which operates a switch. This is called the on-hook condition. Removing the handset from the cradle operates the switch and connects the transmission circuit across the line. The change from open circuit to closed circuit represents a seize signal or if the telephone is being called, the change represents the accept signal. In both cases replacement of the handset at the end of a call is the clear signal.

These routing signals can be detected at the feeding bridge in the local switching centre if the bridge incorporates a relay which operates whenever a line current begins to flow. An a.c. signal is generally used to call a terminal (sending a select signal) . A bell , in series with a capacitor, is connected across the line at the terminal, so that it rings when a.c. is supplied at the switching centre.

Dialing

In a manual telephone system, the routing signal consists of verbal instructions to an operator who operates the necessary switches and gives a verbal report of the progress of the call .In an automatic system, the routing signal must be machine recognizable.

Pulse Dialing

The earliest and still the most common form of signaling is based on the dial . This is an extremely ingenious and cheap device . The basic idea is to interrupt the d.c. path of the subscriber's loop for a specific number of short periods to indicate the number dialed. This is called loop-disconnect or rotary signaling. In most countries the dial operates at about ten impulses per second with a break of about 66 2/3ms and a make of about 33 1/3ms. It is necessary to indicate the end of a pulse train so that a decoding circuit can know where one digit ends and the next one begins. This is achieved by the mechanical design of the dial which ensures a minimum make period between any two consecutive digits . This period is called the inter-digit pause and is typically 200ms. The clear signal is produced when the user replaces his handset, breaking the d.c. path. This break is distinguished from the break produced by dialing by its duration. Any break lasting longer than a few milliseconds is taken as a clear signal.

Tone Dialing

The use of a dial to send routing signals is fairly slow and inconvenient when ten or more digits are needed, as with nationwide and international dialing. A more rapid method is the tone dialing, designed to be used with push-buttons. The principal method uses a pair of tones to signal each digit. These two tones are selected as follows:

one from 697,770,852,941 Hz and

one from 1209,1336,1477,1633 Hz.

There is a total of sixteen possible tones. Only ten are needed for ordinary telephone numbers. The other six may be used for control signals. When only ten or twelve buttons are used, the 1633 Hz tone is omitted. In practice it is possible to build tone receivers that reliably recognize a pair of tones within 40ms. Allowing 40ms inter-digit pause , a maximum signaling rate of about twelve digits is feasible. This method is advantageous because the user can signal more complicated requests as there are extra buttons. Further, it is possible for a user to make a call to a computer, use the tones to send data and thus obtain stock market quotations, credit rating cheques and so on with tape recorded voice replies or voice synthesisation.

Other forms of push button terminals exist. One form uses an electronic circuit within the instrument to remember the sequence of buttons pushed and to generate the loop-disconnect signals for the digits. This permits the use of push button instruments without the need to change a switching system based on loop-disconnect signaling from other terminals.

Another form uses different forms of d.c. conditions between earth and each member of the pair of wires. This form is used in private automatic branch exchanges because it is cheaper Than the tone method. It is unsuitable for public systems because it is affected by interference from power cables and it is also difficult to provide a good earth connection at every user's premises.

Other forms of local signaling consist of audible tones. The accept signal is the start of dial tone Terminal free is the start of the ring tone. Answer is the cessation of the ring tone.

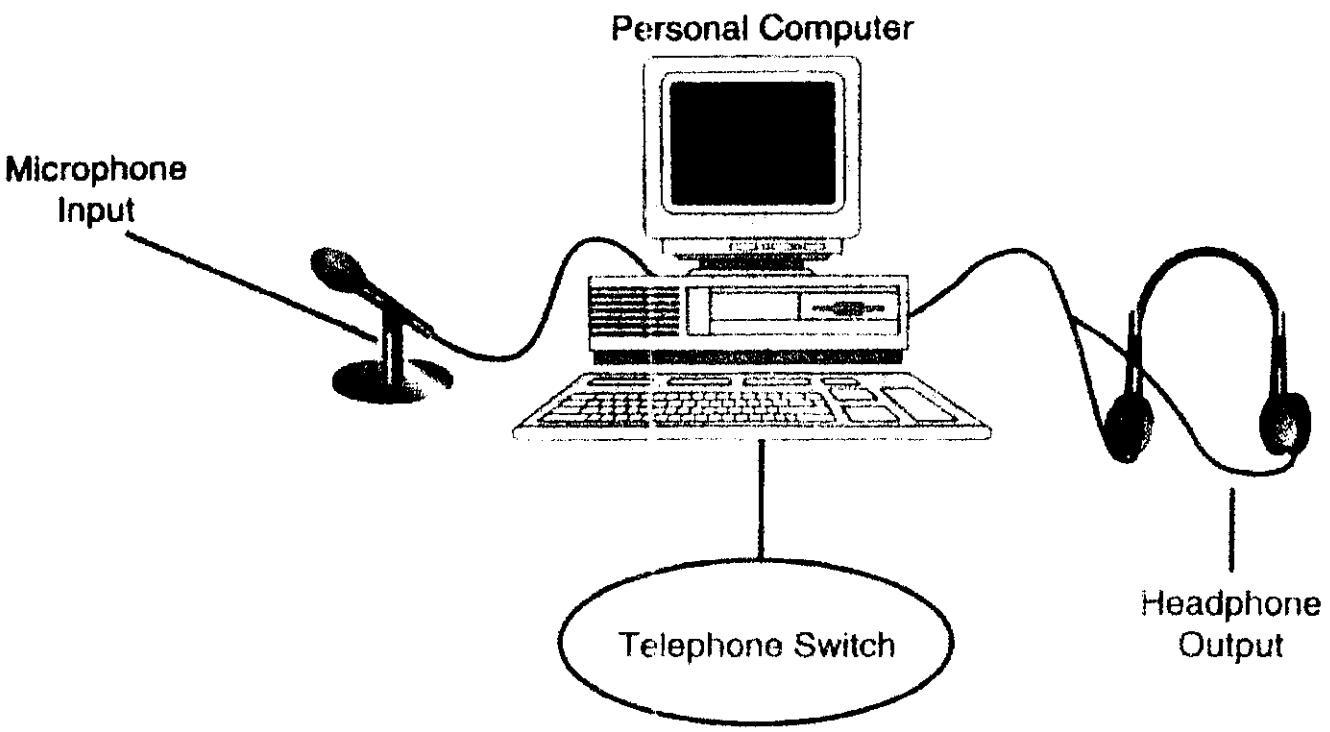
1.2 PROPOSED SYSTEM

The proposed system completely eliminates the need for a telephone handset. In this system the PC and Multimedia are linked to telephone lines. The phone line is connected to the PC through a modem. The multimedia speakers and microphone can be used instead of the conventional telephone handset. Thus, the user can place calls and receive calls from his workstation itself.

This system also provides a database of names and phone numbers. The user can look up a name and the corresponding number before dialing. This reduces errors and increases the efficiency of the system.

Call monitoring and charging are two important features in any communication system installed in an office environment. This system includes a logbook containing details of the date and time of the calls made, duration of each call and the called number. The logbook can be viewed by a system administrator who can monitor and charge calls.

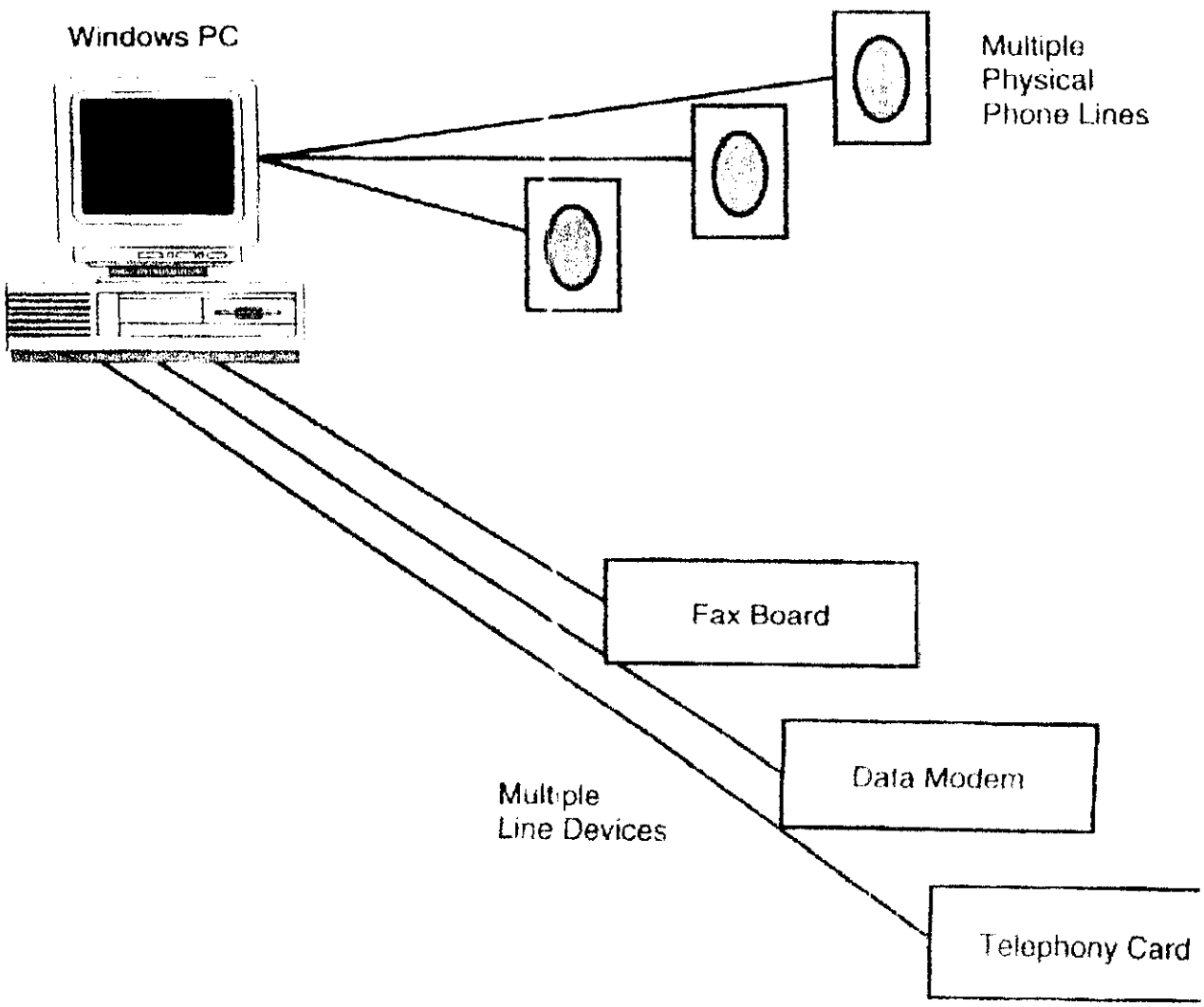
The last important feature of this system is its user friendliness. The user can use a mouse to select the telebook, logbook or the phone dialer. A combo box displays a list of names. If a name is selected , it's corresponding number is automatically dialed.


Personal Computer
Microphone
Input
Telephone Switch
Headphone
Output

1.3 WHAT IS TAPI

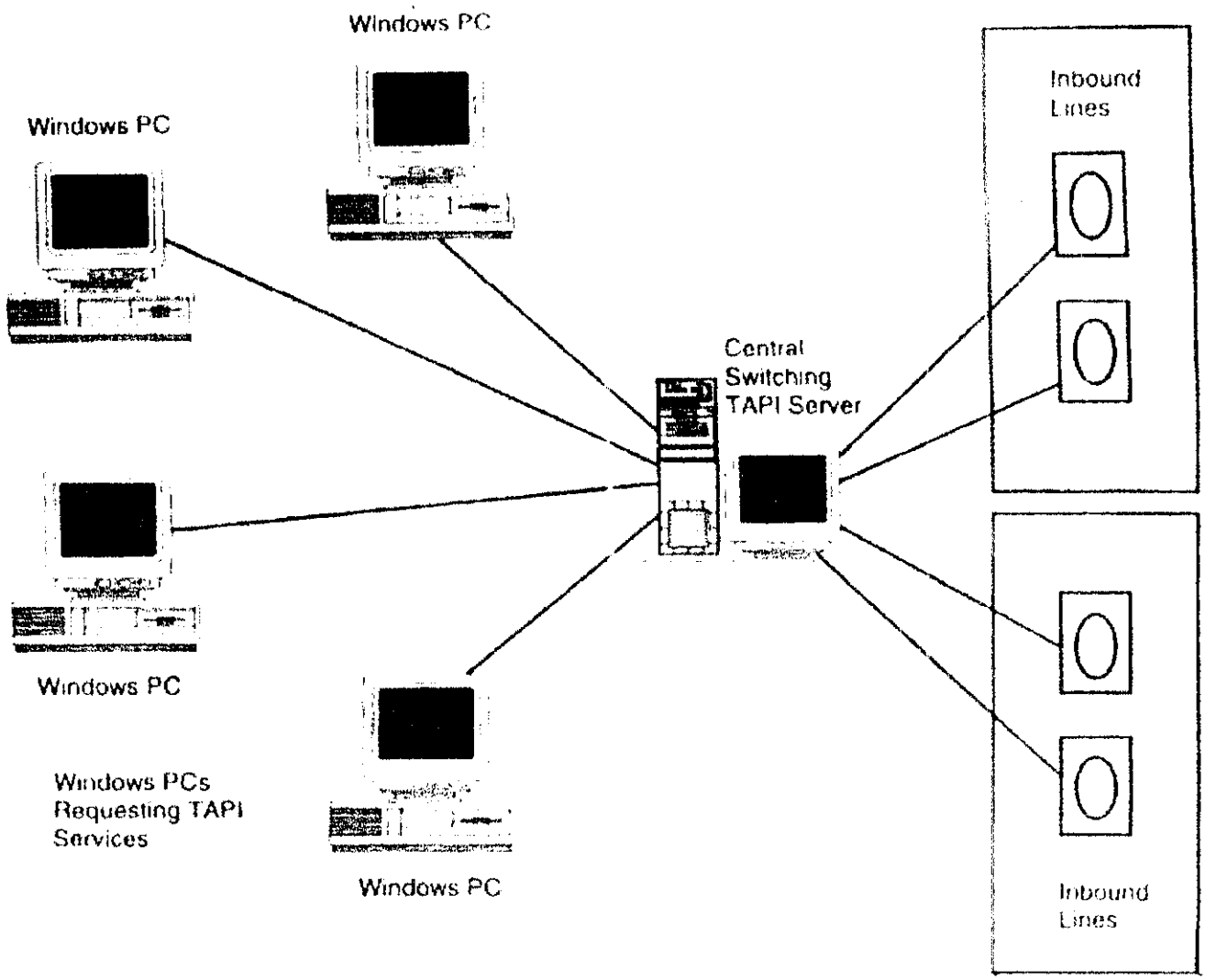
TAPI is an application programming interface that is used to communicate by means of telephones. Some of the abilities that TAPI facilitates are:

- Connecting directly to a telephone network.
- Automatic phone dialing.
- Transmission of data. (files, faxes, electronic mail)
- Access to data.(news, information services)
- Conference calling.
- Voice mail.
- Caller identification.
- Control of a remote computer.
- Collaborative computing over telephone lines.


Windows PC
Multiple
Physical
Phone Lines
Fax Board
Data Modem
Multiple
Line Devices
Telephony Card

THE TAPI ARCHITECTURE

TAPI operates independently of the underlying telephone network and equipment. An application that uses TAPI doesn't have to worry about what it is hooked up to . It just needs to interact with the Application Programmer's Interface, API itself. Fig 1(on the next page) shows the relationship among the application, TAPI and the telephony service provider. The application interacts with the telephony dynamic link library (DLL) by means of TAPI. The telephony DLL is part of the Windows operating system and implements TAPI support. The telephony DLL interacts with service providers which are written by telephony hardware vendors to support operating within windows.


Windows PC
Windows PC
Windows PC
Windows PC
Windows PCs
Requesting TAPI
Services
Windows PC
Central
Switching
TAPI Server
Inbound
Lines
Inbound
Lines

2. REQUIREMENTS

The application needs a wide range of hardware and software tools. The user's workstation must be equipped with all the tools necessary to access all the Windows services used by this application.

4.1 The following list summarizes the **software tools** which are vital for this application.

- Windows 95 Operating system.

The Application Programming Interface (API) set used in this application is a part of the Windows Open Services Architecture (WOSA) . The concept of WOSA is to design a way to access extended services from the Windows operating system that require only a minimum amount of information about the services. This application will run without problems in both Windows 95 as well as Windows NT.

- Visual C++ (Version 5.0)

Visual C++ is a powerful Windows object - oriented visual programming and development system. Complex 32 - bit applications can be developed using VC++. The Microsoft Foundation Class (MFC) version 4.0 is used . By using MFC , large applications can be developed using the various MFC classes , objects and functions.

- Telephony API tools.

This application was developed using TAPI v1.4 for Windows 95 operating system and Visual C++ (v4.0 and above).The version 1.4 run time files are shipped as part of the Windows 95 operating system. TAPI services are accessed via

custom OCX files . This is a 32-bit application developed for Windows operating system. A 16-bit version of TAPI (v1.3) and a new version of TAPI (v2.0) for Win NT have also been released.

4.2 The **hardware tools** required for this application are listed below.

- Sound cards, speakers and microphones.

This application makes use of sound cards, speakers and microphones to simulate a telephone device. Any WAV-compatible audio card can be used. The user can speak into the microphone as though speaking into the microphone in a telephone handset. The caller's voice can be heard on the Multimedia speakers.

- Data modems.

A basic data/fax modem that supports the Unimodem/V communications drivers required for this application. Not all modems support Unimodem/Veven though they offer voice-mail or telephony features. Thus an appropriate modem must be selected.

- Telephony cards.

Most of the TAPI features can be accessed from a workstation without a telephony card. Telephony cards provide a much more advanced set of telephony features than do TAPI-compliant data modems The user must use a TAPI

compliant telephony card that has TAPI drivers (service providers) .

- Access to live phone service.

Access to a live phone line is needed for this application. A simple analog phone line can be used. The nature of the TAPI services makes it easy to use digital lines such as T1 and ISDN to run this application. If digital lines are used it must be made sure that the hardware is compatible with the type of phone service that is being used.

****

2.3 WORKSTATION CHARACTERISTICS

Operating system	:	Windows 95
Computer name	:	IBM Personal Computer.
Microprocessor	:	Intel P54c Pentium - 130 Mhz
Memory	:	16MB.
Level 2 cache	:	256KB asynchronous.
Input/Output features	:	One 25 pin ECP/EPP parallel port, Two RS-232C, 9 pin serial (UART), 15 pin monitor port, Keyboard and mouse port.
Video	:	Cirrua logic 5436 SVGA Controller.
Drives	:	Diskette 3.5 inch, 1.44 MB Hard Disk 1.2 GB, IDE CD-ROM 8x Speed, IDE.
Multimedia	:	Sound blaster card Compact disc - Digital video Microphone Speakers - FX20 MIDI Synthesizer

3.1 FOUR PHASES OF THE PROJECT

This project has been broadly divided into four phases . The first phase involves connecting phone lines with the PC. This is achieved using TAPI functions. The second phase consists of recording and playing voice and transmitting them through phone lines in the appropriate format. The third phase comprises of Database management , which is creation and maintenance of a telebook and a logbook. The fourth and final phase of the project is User Interface. The application is made as friendly to the user as possible.

Each phase is handled in individual chapters in this report . A detailed explanation of the requirements , design and implementation of each phase is given in the following chapters.

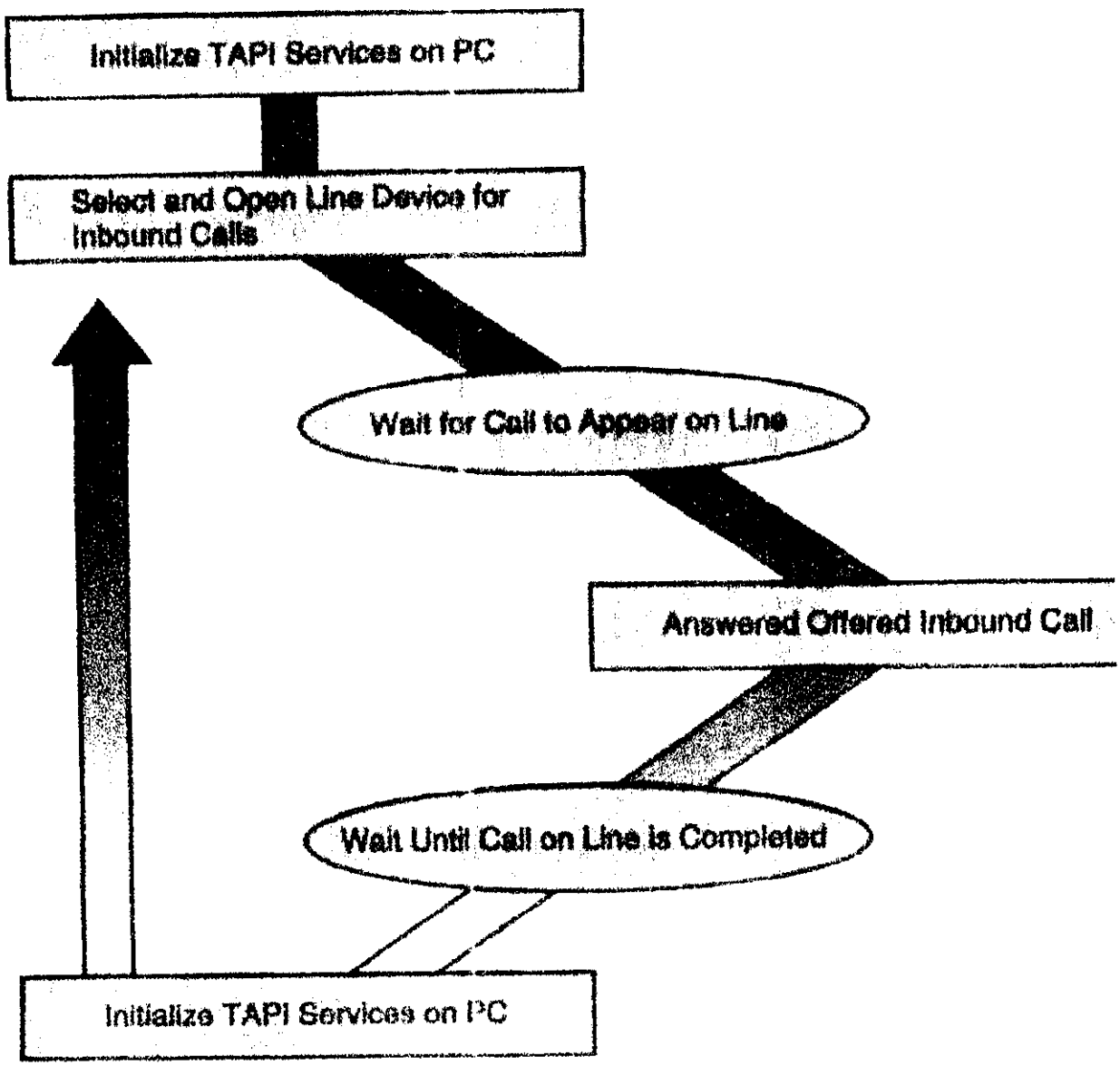
3.2 CONNECTING PC WITH PHONE LINES.

The simplest form of TAPI service is assisted telephony. Under the assisted telephony interface the programmers can place outbound calls and check the current dialing location of the workstation . In fact the Assisted telephony model only provides access for programs to request the placement of an outbound call. The actual dialing of the call is handled by another Windows/TAPI application. This is a C++ program that allows users to enter a phone number and use TAPI to place an outbound call . The basic telephone line can be connected to a PC by the following steps :

1. Using TAPI to place outbound calls.
2. Using TAPI to receive inbound calls.

The minimal steps required to place an outbound call using TAPI services are

- Initialize the TAPI session
- Confirm the TAPI version you are using
- Get a list of device capabilities
- Open a line for outbound calling
- Place an outbound call
- Drop an active call
- De-allocate the idle call
- Close an open line device


Initialize TAPI Services on PC
Select and Open Line Device for Inbound Calls
Wait for Call to Appear on Line
Answered Offered Inbound Call
Wait Until Call on Line is Completed
Initialize TAPI Services on PC

Calling line initialize to start the TAPI session

A TAPI session is started by calling the lineInitialize routine to initialize the link between your application and the TAPI service provider. The lineInitialize routine includes a pointer to the callback function in your code that will handle all messages. After initialization , the routine returns a value in the lineHandle parameter. This value is used throughout the TAPI session. The return value gives a count of the total number of TAPI lines defined for a workstation. This information is used to check the API version and line parameters before you attempt to place a call.

If an error occurs a non-zero value is returned.

Calling lineNegotiateAPIVersion to check TAPI services

After successfully opening the TAPI session the lineNegotiateAPIVersion is called for each line. The version of TAPI which is being used (in this case -version 1.4) needs to be checked because it is possible that the user is requesting a version of TAPI that is not available in the machine. The API version request is passed to the function and the version of TAPI that the workstation can provide is got back. A pointer pointing to a structure that holds information about vendor-specific extension services is also got. This is a method for allowing non-TAPI services to be recognized using TAPI interface.

Using lineOpen to locate an appropriate TAPI line device

After successfully negotiating the API version the lineOpen function is to be passed through each line and request an appropriate level of request . The user can choose the

type of service - voice, fax, data and so on. If no such device is available the user gets an error message. However ,if an appropriate line is available, a zero is returned. A value indicating the handle of the open line is also returned. This is used in subsequent TAPI calls.

Setting call parameters with LINECALLPARAMS structure

After locating an appropriate line the calling parameters are set using the LINECALLPARAMS structure. This structure is used to tell TAPI the speed and media type of the call and other values.

Setting the LINECALLPARAMS structure is optional. Default values will work just fine.

Using lineMakeCall to place the call

The lineMakeCall function is used to actually place the call. This function passes the string that contains the phone number to call , a handle for the open line (from lineOpen)and , optionally ,a pointer to the LINECALLPARAMS structure.
If the call is placed successfully , a call handle is returned. Otherwise the returned code is non-zero and can be checked for appropriate correction.

At this point the call has been placed but not completed. TAPI will continue to receive status information and route that to the application through the callback function which

was registered when lineInitialize was called. The quality of the status information (dialing , ringing, busy, idle , and so on) is determined by the TAPI service provider.

Resources and code review

A message queue is declared , the current instance of this program is got and then the main dialog box is called. The main dialog box contains only a few controls. An input box for the phone numbers , a list box to show the status messages supplied by TAPI , and three command buttons - Placecall, disconnect , exit are created. The code for the main dialog box can be broken into three main sections

- responding to user commands (button presses)
- displaying posted messages from the callback routine.
- performing code operations based on posted messages from the callback function

The first part of main dialog code responds to initial loading of the dialog box and to user actions of the command buttons. First, when the dialog box first starts , three properties are created. These will hold values used throughout the dialog. The next event is the pressing of the ID_CALL button . This tells the dialog box to attempt to place a call. The first step is to check to see if a call is already in progress. If so, a message is displayed to the user. If no call is currently in progress , the phone number is gathered from the input box and then passed to the PlaceCall function for final dispatch. If the user presses the ID_DISCONNECT button , the program checks the COMM and CALL handles and ,if they are set , clears them using the clear handle and lineDrop functions. Finally, when the user presses the exit button (ID_OK) , the same type of routines executed in

ID_DISCONNECT must also occur here. In addition to the clear handle and lineDrop functions , the line lineClose and lineShutdown routines are called. This performs final closure on all TAPI services for this session .

The user can use the number pad for entering the phone number into the input box . If the user selects redial option , the number that was previously dialed is automatically dialed again. The user can also choose between the mode of dialing (Pulse / Tone).

The second session of main dialog is used to respond to TAPI messages received via the lineCallback function and passed on to the main dialog . The only message that needs attention is when a call goes idle. If a call goes idle , the main dialog needs to close down the line resources just as if the user had pressed the disconnect button.

The PlaceCall function

The real heart of the program is the PlaceCall function. This routine is the one that actually places the requested call. The step of initialize, check API , look for a open line , set call parameters and place call are all here in this one routine .

The key functions used are

- Use lineInitailize to register the callback function
- Use lineNegotiateAPIVersion to check for valid TAPI services
- Use lineOpen to ask for a line that suits the need
- Use LINECALLPARAMS to optionally set calling parameters
- Use lineMakeCall to place the call

Once TAPI performs the lineMakeCall successfully you it is still not connected to the called party. The callback messages should check the status of the call.

The ShowProgress and SetVarProps procedures

There are two helper routines in this application. The first , ShowProgress, is used to post the messages to the list box on the main dialog form.

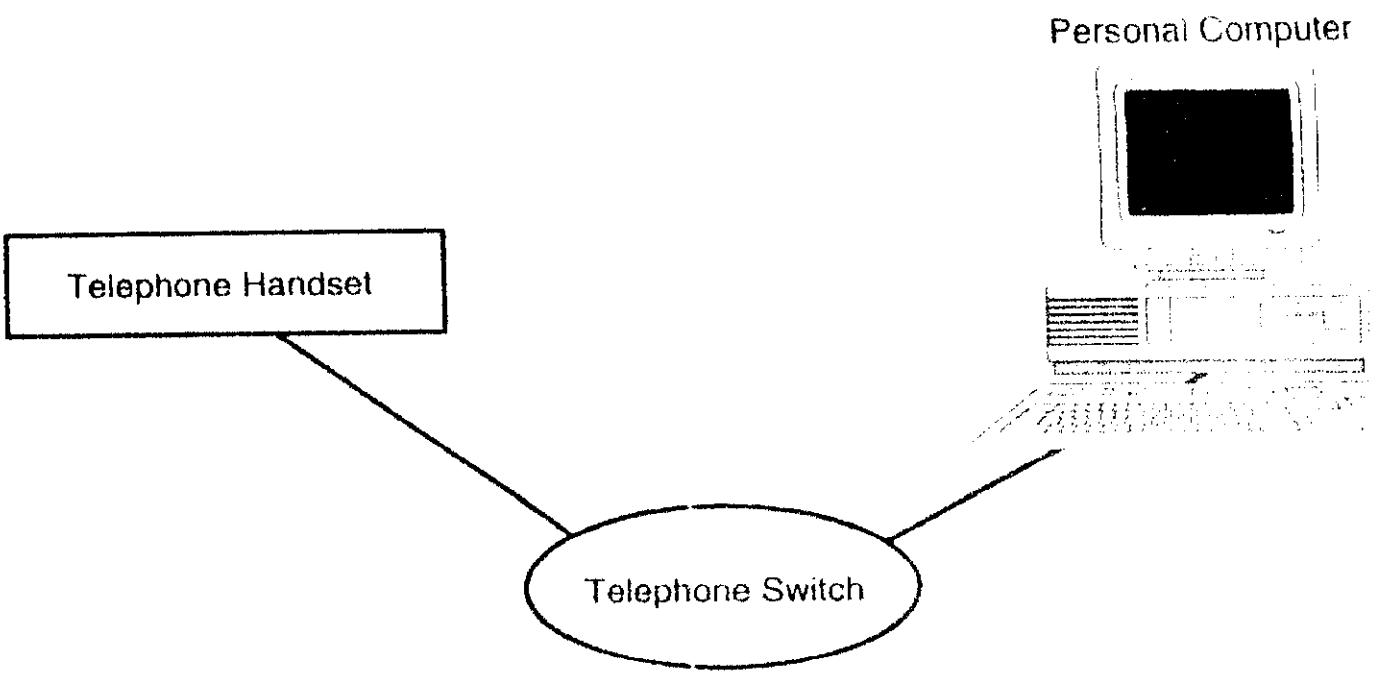
The lineCallBackProc procedure

The lineCallBackProc is the routine registered (using lineInitialize) to receive all TAPI messages for this application. This code is broken into three segments :

- Handling LINE_CALLSTATE messages
- Passing status information to the main dialog box
- Other messages currently ignored by the application

The first section of the lineCallBackProc contains code to respond to changes in the LINE_CALLSTATE message. The LINE_CALLSTATE_CONNECTED routine contains code that retrieves the call handle and gets the lineID . The LINE_CALLSTATE_IDLE routine de-allocates the device handle to free up resources for the next call.

The second segment of the callback routine passes messages to the dialog box for display. These are added here to show how the messages work and how the call progress can be reported.


Personal Computer
Telephone Handset
Telephone Switch

PHASE II

3.3 LINKING MULTIMEDIA TO TELEPHONE LINE VIA MODEM

After establishment of connection using TAPI the user should be able to converse using the multimedia kit at his desk. The user will talk into the multimedia microphone. The voice is stored in a file in wave format.

As it is not possible to transmit wave files on telephone lines, the wave files are converted into telephone compatible VOX format . The VOX file is transmitted over the line and the user at the other end who will be using the telephone handset will be able to listen to the voice. Also the voice from the other end is played in the multimedia speakers

The VOX file is transmitted to the communication port of the modem . The file is stored in the transmit buffer of the comm port and will be transmitted over the line. Similarly, the voice coming in will be stored in the receive buffer of the modem comm port and can be retrieved and played in the speakers.

For communicating with the ports , Communications control are used. The Communications control can be used to open a serial port, change its settings, send and receive data through the port, and monitor and set many of the different data lines. Its dual-method access allows for both polling and event-driven communications.
The communications control provides the following two ways for handling communications:

Event-driven communications is a very powerful method for handling serial port interactions. In many situations we want to be notified the moment an event takes place, as when a character arrives or a change occurs in the Carrier Detect (CD) or Request To Send (RTS) lines. In such cases, we would use the communications control's OnComm event to trap and handle these communications events. The OnComm event also detects and handles communications errors.

We can also poll for events and errors by checking the value of the CommEvent property after each critical function of your program. This may be preferable if your application is small and self-contained.

Important properties of communications control

Properties	Description
CommPort	Sets and returns the communications port number.
Settings	Sets and returns the baud rate, parity, data bits, and stop bits as a string.
PortOpen	Sets and returns the state of a communications port. Also opens and closes a port.
Input	Returns and removes characters from the receive buffer.
Output	Writes a string of characters to the transmit buffer.

CDTimeout	Sets and returns the maximum amount of time (in milliseconds) that the control waits for the Carrier Detect (CD) signal before timing out.
Handshaking	Handshaking refers to the internal communications protocol by which data is transferred from the hardware port to the receive buffer. A handshaking protocol ensures that data is not lost due to a buffer over-run, in which case data arrives at the port too quickly for the communications device to move the data into the receive buffer.
RTSEnable	Determines whether to enable the Request To Send (RTS) line. Typically, the Request To Send signal that requests permission to transmit data is sent from a computer to its attached modem.
InBufferSize	InBufferSize refers to the total size of the receive buffer.
SThreshold	Sets and returns the minimum number of characters allowable in the transmit buffer .
OutBufferSize	OutBufferSize refers to the total size of the transmit buffer.
ParityReplace	Sets and returns the character that replaces an invalid character in the data stream when a parity error occurs.

The following table lists the valid baud rates.

110,300,600,1200,2400,9600 (Default),14400,19200,38400

56000 (reserved),128000 (reserved),256000 (reserved).

The following table describes the valid parity values.

Setting	Description	
E	Even	
M	Mark	
N	(Default)	None
O	Odd	
S	Space	

The following table lists the valid data bit values.

Setting	
4	
5	
6	
7	
8	(Default)

The following table lists the valid stop bit values.

Setting	
1	(Default)
1.5	
2	

RTSEnable Property :

Setting	Description
True	Enables the Request To Send line.
False	(Default) Disables the Request To Send line.

When RTSEnable is set to True, the Request To Send line is set to high (on) when the port is opened, and low (off) when the port is closed.

The Request To Send line is used in RTS/CTS hardware handshaking.

MODEM HANDSHAKING

Most of the currently available modems contain a dedicated microprocessor. The built-in intelligence allows these units to automatically dial a specified number with either tones or pulses, and redial the number if it is found busy or doesn't answer. Some modems allow the user to establish voice contact and then switch over to digital data communication.

After a modem dials up another modem, a series of handshake signals take place. The modem which makes the call is called the calling modem while the modem which receives the call is called the called modem.

Assuming that the $\overline{\text{DTR}}$ of the called modem is asserted, the ringing signal on the line will cause the data access arrangement, DAA circuitry to assert the ringing input, RI. If the $\overline{\text{DTS}}$ and the $\overline{\text{RTS}}$ of the calling modem are asserted indicating that data is ready to be sent , the calling modem puts a mark (a tone of 2225 Hz) on the line for 8 ms to let the called modem know that contact is complete. In response to this mark the called modem asserts its carrier detect output, $\overline{\text{CD}}$ to enable the receiving UART. The calling modem then starts sending data until its $\overline{\text{RTS}}$ input is released by the computer or terminal sending the data.

For a full duplex system the handshake is similar , but the data rates are equal in both directions.

Two organizations are responsible for most of the current standards regarding modems. They are the Bell Telephone standards of the United States and the Comite Consultatif Internationale Telephonique et Telegraphique (CCITT) , which is part of the International Telecommunications Union.

3.4 DATABASE MANAGEMENT

Database is linked to TAPI Applications as additional feature to it. The Database contains three tables phone_det, userdet and usrmst.

The table phone_det provides the user with phone numbers and addresses of customers.

Usrmst table has the details of the user with each user having an unique ID.

Userdet has the details of each user whether the user has sent or received the call , personal or official call.

This saves the valuable time of user who has to search the phone_details of the customer in the phonebook instead. Also it calculates the duration of each call by the user.

In each table the data can be inserted, updated or deleted.

TABLE 1-Useraddress

This table gives the user name , address and the ID for each user .The primary key is the User_ID which is unique to each person in the company.

TABLE 2-Userdetails

This table gives the type of call ,mode of call ,the duration of the call for each call of the user . Here the User_ID is the foreign key which acts as the primary key in the Useraddress table .It also gives the called number .

TABLE 3- Phone details

This table is the directory to every person using the system .

Here phone number is the primary key as it is unique . On entering a name in the edit box ,the phone numbers of all the persons with that name are listed and the user can select the number he wants to dial to On selecting the number the rest of the details are automatically filled.

OPENING AND CLOSING OF THE DATABASE

An ODBC data source is a specific set of data, the information required to access that data, and the location of the data source, which can be described using a data-source name. The data source includes the data, the DBMS, the network (if any), and ODBC.To access data provided by a data source, your program must first establish a connection to the data source. All data access is managed through that connection.

Data-source connections are encapsulated by class CDatabase Once a CDatabase object is connected to a data source, you can:

- Construct recordsets, which select records from tables or queries.
- Manage transactions, batching updates so all are "committed" to the data source at once (or the
- whole transaction is "rolled back" so the data source is unchanged)—if the data source supports
- the required level of transactions.
- Directly execute Structured Query Language (SQL) statements.

When you finish working with a data-source connection, you close the CDatabase object and either destroy it or reuse it for a new connection.

A CDatabase object represents a connection to a data source, through which one can operate on the data source. A data source is a specific instance of data hosted by some database management system (DBMS).

All the three tables has been created in a database called pubsdb.The tables uses m_db , Cdatabase class to establish a link. The member function **CDatabase::Open** is called to initialize a newly constructed CDatabase object. The database object must be initialized before using it to construct a recordset object.

The member function CDatabase::Open is initialized as follows:

```
m_db.Open("pubsdb",FALSE,FALSE,"ODBC;UID=sa;PWD=kovai");
```

Similarly the member function CDatabase::Close is called to disconnect from a datasource.

Fetching data from the database

Class CRecordset provides support for bulk row fetching, which means that multiple records can be retrieved at once during a single fetch, rather than retrieving one record at a time from the data source. You can implement bulk row fetching only in a derived CRecordset class. The process of transferring data from the data source to the recordset object is called bulk record field exchange (Bulk RFX). Note that if you are not using bulk row fetching in a CRecordset-derived class, then data is transferred via record field exchange (RFX).

There are two principal kinds of recordsets: **snapshots and dynasets**. Both are supported by class CRecordset. Each shares the common characteristics of all recordsets, but each also extends the common functionality in its own specialized way. Snapshots provide a static view of the data and are useful for reports and other situations in which you want a view of the data as it existed at a particular time. Dynasets are useful when you want updates made by other users to be visible in the recordset without having to requery or refresh the recordset. Snapshots and dynasets can be updatable or read-only. In order to reflect records added or deleted by other users, call **Crecordset::Requery.**

ClassWizard uses a default connection string to establish a connection to a data source. You use this connection to view tables and columns while you are developing your application. However, this default connection string may not be appropriate for your users' connections to the data source through your application. For example, their data source and the path to its location may be different from the one used in developing your application. In that case, you should re-implement the **Crecordset::GetDefaultConnect** member function in a more generic fashion and discard ClassWizard's implementation

Recordset: How Recordsets Select Records (ODBC)

Recordsets select records from a data source through an ODBC driver by sending SQL statements to the driver. The SQL sent depends on how you design and open your recordset class.

To open a recordset

1. Construct an object of your CRecordset-derived class.

   You can construct the object on the heap or on the stack frame of a function.

2. Optionally modify the default recordset behavior. For the available options.

3. Call the object's **Open** member function.

   In the constructor, pass a pointer to a CDatabase object, or pass NULL to use a temporary database object that the framework will construct and open based on the connection string returned by the **GetDefaultConnect** member function. The CDatabase object may or may not already be connected to a data source.

   The call to Open uses SQL to select records from the data source. The first record selected (if any) is the "current record". The values of this record's fields are stored in the recordset object's field data members. If any records were selected, both the IsBOF and IsEOF member functions return 0.

In your **Open** call, you can:

Specify whether the recordset is a dynaset or snapshot. Recordsets open as snapshots by default. Or you can specify a forward-only recordset, which allows only forward scrolling, one record at a time.

By default, a recordset uses the default type stored in the CRecordset data member m_nDefaultType. Wizards write code to initialize m_nDefaultType to the recordset type you choose in the wizard. Rather than accepting this default, you can substitute another recordset type.

Specify a string to replace the default SQL SELECT statement that the recordset constructs. Specify whether the recordset is read-only or append-only. Recordsets allow full updating by default, but you can limit that to adding new records only or you can disallow all updates.

After you call Open, use the member functions and data members of the object to work with the records. In some cases, you may want to requery or refresh the recordset to include changes that have occurred on the data source.

Crecordset used in phone details table is C_Prec , C_Urec used in user address table and in User details is Drec .

These Recordsets are created using class wizard while creating the table.

IMPLEMENT_DYNAMIC(C_Prec, CRecordset)

3. 5 USER INTERFACE

User interface is an important aspect of any application. It is desirable to make the interface as user friendly as possible. Today graphical interfaces have replaced textual interfaces as the standard means for user-computer interaction. The goal of user interface design is :

- Speed of use.
- Reduction of errors.

User interfaces rely on point-and-click methods to allow the user to select menu items, icons and objects on the screen. In a well designed user interface, typing is necessary only to input text to be stored and manipulated. An interface is made user friendly by adding user interface objects to it. Giving visual feedback to the user is another way of making the interface user friendly. Menus, toolbars, statusbars and dialog boxes are some of the user interface objects used in this application.

User interface objects have been derived from the MFC library.

ANSWERING INCOMING CALLS

The screen used for handling incoming calls contains only three push buttons. The first push button is similar to the cradle switch in a conventional telephone handset. Any incoming call can be answered by simply clicking the push button with the caption, " ANSWER ". The called phone device stops ringing. The user can then click the " OK " push button to close the screen and return to the main screen after finishing his conversation. The " CANCEL " push button is used to return to the main screen even without answering any calls.

4. CONCLUSION

The software has been developed and has fulfilled the necessary requirements. This software acts as an efficient substitute for the conventional communication through speakers and microphones in a phone device. Multimedia support offered by the software opens several areas for further enhancement. The user can communicate with a remote terminal with telephone lines by using a multimedia kit.

ODBC - Open Database Connectivity has been successfully employed to create and maintain a telebook and a logbook. The telebook is similar to telephone directory holds details concerning the date / time of each call, duration of the call , called number etc.

This software is also extremely user - friendly . It has a wide range of icons, dialog boxes and other attractive user interface objects.

Finally, this system has been proved to be quite useful as a dialer , call manager , directory of phone numbers and also a call monitor . In other words this system can be seen as a computerized substitute for a separate telephone infrastructure.

5. FIELDS OF APPLICATION

This application simply eliminates the need for a telephone handset. This unique feature makes this application very useful in an office environment.

In this age of computers almost every desk in an office has a computer and the average employee spends most of his/her time in front of a terminal. By using this application he/she would be able to place calls and also receive calls through the terminal itself. The user doesn't have to leave his/her desk to pick up a telephone handset. The need for separate telephone devices at each and every desk is also automatically eliminated. Installing this application in a LAN environment might be more economical than providing individual telephone devices at each desk.

This application acts as an efficient call manager. It can place a call to any acceptable number specified by the user. It can receive any external call and intimate the user. The application maintains a logbook which holds details such as the date and time of each call, called number, call duration, etc. A telebook or a directory of phone numbers is also included in this application.

This application can also be used in PCs at home. The user can conveniently place calls and receive calls from his terminal. There is no need to interrupt work and pick up a telephone handset.

Another important use for this application is as a tool, for the disabled. Software and hardware have been extensively developed to make the disabled more comfortable with a PC. This facility can be utilized as an interface between the disabled and other

devices which can be linked with the PC. For instance if a person is using a PC with a mouse through limited movement of his fingers, then he/she can also place calls and receive calls with the help of this application . Thus, the disabled can use a telephone even with limited use of fingers. Without this application, using a telephone would have been impossible to such disabled people.

6. SCOPE FOR FURTHER DEVELOPMENT

This is an innovative application by which the operand uses the multimedia speakers and microphone as a substitute for the telephone handset.

This application which is currently a single user application can be developed into a network application. Consider an office environment. Any incoming call can be received and routed to a particular user's desk.

The software can be developed for LAN network by which the network users can interact with each other even without handsets at their at desks. In other words , this eliminates the need for a separate intercom system.

A conferencing feature can be incorporated. The user can drag names from a database , drop them into a "Conference Box" and can converse with them simultaneously.

The dialer application has been developed using TAPI. The advantage of using TAPI to establish the call is that the line can be monitored and calling parameters can be set. The TAPI callback functions can be used to monitor the present line conditions and incorporate additional advanced telephony features like call transfer, call redirecting , call hold and call parking .

The log book can be used to monitor the duration of each user's call. This can be used for call monitoring and charging.

The user friendliness of the application can be improved by making it more interactive.

Call identification feature can also be incorporated whereby the user can identify the person calling even be he picks up the call.

The efficiency of voice transmission across workstations can be improved by incorporating other advanced features in this system.

BIBLIOGRAPHY

BOOKS

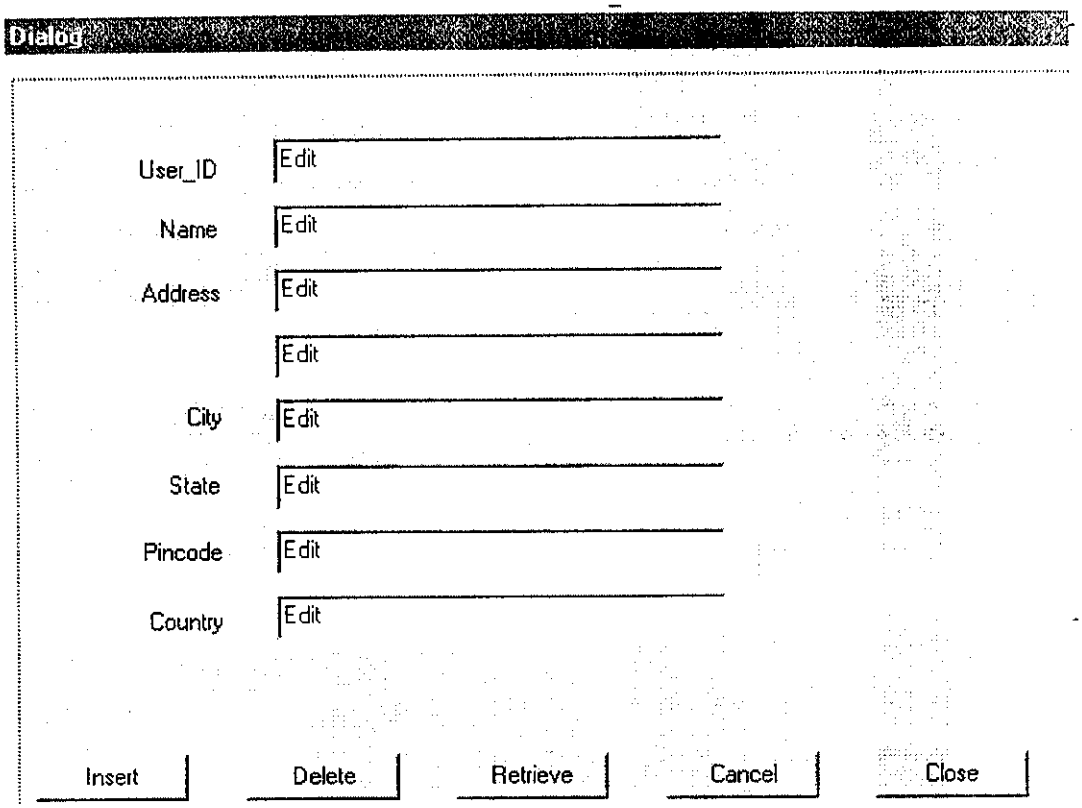
1. Charles Petzold, Progamming Windows 95, Microsoft Press.
2. Hills.M.T., Telecommunications Switching Principles, George Allen & Unwin, London.
3. Michael Amundsen, MAPI,SAPI & TAPI - A Developer's Guide.
4. Andrew S. Tanenbaum, Computer Networks, Prentice-Hall International.
5. John F. Koegel Buford, Multimedia Systems, ACM Press SIGGRAPH Series.
6. Brian Myers and Eric Hamr, Mastering Windows NT Programming, Sybex/Tech Asian Edition.
7. Mickey Williams, Essential Visual C++, Prentice-Hall International
8. Paul Yao and Joseph Yao, Foundations of Visual C++ programming, Comdex IDG Books.
9. Bjarne Stroustrup, The C++ Programming Language, Addison Wesley.

ARTICLES

1. Nigel Thompson, Recording and Playing Waveform Audio, Microsoft Developer Network Technology. Oct-95.
2. Computer Telephony, Aug-96.

WEB SITES

1. www. super-hwy.com
2. www.swi.com
3. www.afisys.com
4. www.glp.net
5. www.stylus.com
6. www.vive.com


Dialog
User_ID
Edit
Name
Edit
Address
Edit
Edit
City
Edit
State
Edit
Pincode
Edit
Country
Edit
Insert
Delete
Retrieve
Cancel
Close

ACRONYM

CCITT	Comite Consultatif Internationale Telephonique et Telegraphique
CD	Carrier Detect
DAA	Data Access Arrangement
DLL	Dynamically Linked Library
DTR	Data Terminal Ready
ISDN	Integrated Service Digital Network
MFC	Microsoft Foundation Class
ODBC	Open Database Connectivity
RFX	Record Field Exchange
RTS	Ready To Send
TAPI	Telephony Application Programmer`s Interface
UART	Universal Asynchronous Receiver and Transmitter
WOSA	Windows Open Services Architecture