Automation of Inquiry
Of
PBX Exchange.

Undergraduate Degree Project TE-35

By
Capt Shahid Ahmed(Project Leader)
Capt Kifayat-ullah
Capt Haseen-ullah
Capt Iftikhar

Project Supervised by
Lt Col Nadeem.

Dissertation to be presented as partial requirement for the award of
B.E Degree in Telecommunication Engineering.

Military College Signals
National University of Sciences and Technology,Rawalpindi.

ACKNOWLEDGMENTS

We are grateful to Allah Almighty for giving us the strength to visualize undertake and complete this project. Our humble gratitude to our teacher and project supervisor Lt Col Nadeem, For guiding and encouraging us through out the course of this project. We are thankful to him for providing us with innovative ideas, project management techniques and constant moral support.

We are very grateful to DR. Majid Alvi and Mr. Ali Imran for providing guidance, support and in depth knowledge to understand contemporary systems and their problems.

We are thankful to all our friends for lively discussions, exchange of ideas and source of encouragement during this project.

Finally, we would like to thank our parents and family members for their perseverance and continued moral support, which helped us to get through hard times.

Abstract

This document has been prepared as a dissertation of final year degree project, to be presented to MES/NUST in partial requirement for the award of B.E. degree in the discipline of Telecommunication Engineering.

This dissertation discusses automatic inquiry system in PBX exchanges: An on- demand phone in service. It encompasses the vision behind the project, the way its feasibility and requirements were analyzed, the design was drawn and implementation took place. The aim of the project was to implement an automated inquiry system in PBX exchanges. The research and subsequent development of this project was carried out by Capt Shahid Ahmed,Capt Kifayat-ullah,Capt Haseen-ullah and Capt Iftikhar under the Guidance and supervision of Lt Col Nadeem.

Project specifications

Statement

Automated Inquiry System: Implementation of automated Inquiry system in PBX exchanges.

Development Environment/Tools

1. JDK 1.3.
2. Sun's JTAPI.
3. Sun's JCOMM.
4. KAWA.

Development languages

1. JAVA 2.
2. SQL.
3. PL/SQL.

Platforms supported

1. MS Windows 95, 98.
2. MS Windows NT 4.0 with service pack 3.
3. MS Windows 2000.
4. MS Windows Me
5. MS Windows XP

Table of Contents

1.2 Background
1.3 Minimum Requirement for Automated Inquiry.
1.4 Areas of Application
2-Speech Interface.
2.1 Speech Recognition.
2.2 Speech Synthesis.
2.3 Java Telephony API.
2.3.1 Voice Mail
2.3.2 Accessing Data
2.3.3 Call Routing.
2.3.4 Multiline Applications.
3-Database Development.
3.1 Dr Codd's 13 Points.
3.1.1 File Based System.
3.1.1.1 drawbacks.
3.1.2 Database Approach.
3.1.2.1 Database.
3.1.2.2 DBMS
3.1.2.2.1Characteristics of DBMS
3.1.2.2.2 Components of DBMS
3.1.2.3 Roles of Database Environment.
3.1.2.4 Advantages of Database System.
3.1.2.5 Disadvantages.
3.1.2.6 Functions of DBMS.
3.1.3 Foundational Rules.
3.1.4 Structural Rules.
3.1.5 Integrity Rules
3.1.6 Data Manipulation Rules.
3.1.7 Data Independence Rules.
3.2 Database Languages.
3.2.1 Data Definition Language.
3.2.2 Data Manipulation Language.
3.2.3 4^{th} Generation.
3.3 Data Models.
3.3.1 Object Oriented.
3.3.2 Physical Data models.
3.3.3 Record Based.
3.3.3.1 Hierchical data model.
3.3.3.2 Network Data model.
3.3.3.3 Relational data model.

3.4 Data Dictionary.
3.5 ER Model.
3.5.1 ER Diagram.
3.5.2 Entity.
3.5.3 Attributes.
3.5.4 Relationship.
3.5.5 Normalisation.
3.5.5.1 1NF.
3.5.5.2 2NF.
3.5.5.3 3NF.
4-Modems.
4.1 Modem Internal Composition
4.1.1 UART
4.1.1.1 UART Types.
4.1.1.2 Regiters
4.1.1.2.1 Receive Buffer Register.
4.1.1.2.2 Transmitter Holding Register.
4.1.1.2.3 Interrupt Enable Register.
4.1.1.2.4 Interrupt Identification Register.
4.1.1.2.5 FIFO Control Register.
4.1.1.2.6 Line Control Register.
4.1.1.2.7 Modem Control Register.
4.1.1.2.8 Line Status Register.
4.1.1.2.9 Modem Status Register.
4.1.1.2.10 Scratch Register.
4.1.1.2.11/12 Division Latch Register.
4.2 Why Modulate
4.2.1 Advantages of Passband Modulation
4.2.2 Different Modulation Schemes.
4.2.2.1 Binary Modulation Schemes.
4.2.2.1.1 Amplitude Shift Keying.
4.2.2.1.2 Frequency Shift Keying.
4.2.2.1.3 Phase Shift Keying.
4.2.2.2 M-ary Modulation Schemes
4.2.2.2.1 Quadrature Phase Shift Keying.
4.2.2.2.2 M-ary quadrature Amplitude Modulation
4.3 The Intelligent Network.
5-Telephone Line.
5.1 Line States.
5.2 Telephone line Characteristics.

5.2.1 Attenuation Distortion.
5.2.2 Propagation Delay.
5.2.3 Envelope Delay.
5.3 Line Impairments
5.3.1 Crosstalk.
5.3.2 Signal Return.
5.3.3 Frequency Shift.
5.3.4 Nonlinear Distortion.
5.3.5 Jitter
5.3.5.1 Amplitude.
5.3.5.2 Phase.
5.4 Transients.
5.4.1 Impulse Noise.
5.4.2 Gain Hits.
5.4.3 Dropouts.
5.4.4 Phase Hits.
6-RS-232 Characteristics.
6.1 Rs-232 Signals.
6.1.1 Protective Ground
6.1.2 Tranamitted Data.
6.1.3 Received Data.
6.1.4 Request To Send.
6.1.5 Clear To Send.
6.1.6 Data Set Ready.
6.1.7 Signal Ground.
6.1.8 Received Line Signal Detect.
6.1.9 +P(for Testing)
6.1.10 -P(for Testing).
6.1.11 (unassigned)
6.1.12 Secondary Received Line Signal Detect.
6.1.13 Secondary Clear to Send.
6.1.14 Secondary Transmitted Data.
6.1.15 Transmission Signal Element Timing.
6.1.16 Secondary Received Data.
6.1.17 Received Signal Element Timing.
6.1.18 (Unassigned).
6.1.19 Secondary Request To send.
6.1.20 Data terminal ready.
6.1.21 Signal Quality Detector.
6.1.22 Ring Indicator.

Chapter-1

1.Introduction.

The basic idea behind an automated Inquiry system is to make easy query of subscriber's numbers and a number of useful services available to the user through the normal telephone set, without the involvement of the operater. This PC can work as PBX exchange in small military installations like military headquarters, hospitals and units. This PC can easily provide facilities like E-mail by phone, News update, Weather forecast, Sports news, Personalize messaging and many more.

The service that will be most useful in this context is the automated inquiry by phone, which will enable the users to check their requisite subscriber's numbers and other available facilities without the involvement of operator. This facility can also include the option of personalized messaging in which a user can record his message for another subscriber at the exchange with the help of telephone. This message can later be retrieved by the person, for whom it is intended, by calling at his identification number.

1.1 What is Automated inquiry via phone, and what can it do?

Automated Inquiry via phone enables a subscriber to access provided exchange facilities independently without the involvement of operater using any touch-tone phone.This phone can be used to read e-mail messages using text-to-speech technology and serve as a voice mail system, recording callers messages and making them available by phone.

This Pc can be used to locate individual subscriber's numbers and can also be used to connect subscriber's calls as well.This thing is shown in the following figure, where Provider (means PC exchange here) is acting as exchange.

1.2 BACKGROUND

In order to chech the feasibility of this project we carried out research work and came to know that this idea has never been implemented in Pakistan. Although there are some sites available in Canada and USA,which gives full real time communication concept, but majorities of them were based on either answering machine or voice messaging. But the problem with real time communication providers is that they want others to forward their calls to their number, which is highly infeasible for the local market of Pakistan,in terms of cost of long distance call charges.

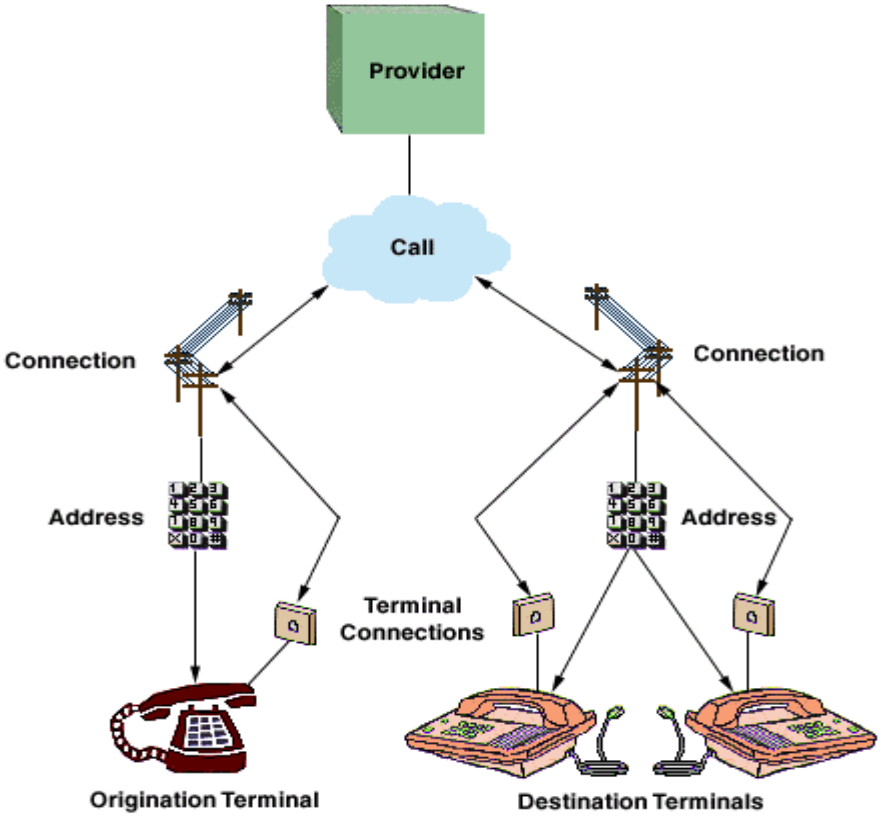
During the research work we went to different companies, Army and civil exchanges and came to know that we are importing exchanges like Mega tech/ABM from other countries,which are costing about 5 million ruppees and performing same kind of job as we can get from this simple PC exchange.So we decided to set the initial mile stone in the development of our own PBX exchange by automating Inquiry system through simple PC.

To implement this Idea, we started visiting software companies and ISP's like Enabling technologies,Netsole,Alexer ,A and OR and Brain Net, but we got different vague ideas of this project implementation through software.

We initially started our work in Turbo c++ along with interrupt functions,assembly language,AT commands relating to modems,but after three weeks struggle it became clear to us that it was very difficult way to accomplish the task,so we adopted JAVA programming for the same purpose.Atlast we started with a rough model and implemented the idea.Accomplishment of this task gave us clear idea of making PBX exchange,which can provide all the facilities of ABM/MEGA Tech exchanges, from simple 486 or P1 PCs.This exchange will hardly cost about 6-10 thousand rupees as compared with ABM/MEGA Tech exchanges.

This figure represents the implementation of this idea, where provider is showing simple PBX exchange.

FIGURE



1.3 Minimum Requirement for Automated Inquiry.

1. Intel Pentium processor (100 MHZ min).
2. Voice Modem.
3. Touch Tone telephone.
4. Microsoft Windows 9x, NT. Operation system.
5. Sound Card(8 bit).
6. Hard Disk(540 MB).
7. RAM(32 MB).
8. JDK 1.2.

1.4 Areas of Application

The areas of application of such automatic inquiry system are as vast as the combined domain of computer telephony and Text-to-Speech. An existing PSTN, Internet Telephony service provide its customers with an advanced mode of communication.

Many dot.com Companies in USA are offering similar services. Notable among them are JFAX, Shout mail, Mail Call, eVoice, eFax, CallWave, Message Click, oneBox, uReach, CoolSpeak etc.

With the advent of PTCL semi-automatic inquiry and Personal Mail Box service, and its subsequent success,it can easily be said that providing this service on a commercial basis in Pakistan is momentarily feasible plan.

Chapter-2

2 Speech Interface

Today, just about every new PC and even many older PCs can handle speech. . Speech recognition enables users to translate speech into commands/data and text, and simplifies the interface between the user and the computer. Speech synthesis enables the computer to provide output to the user via the spoken word. Although these technologies have been available for a few years, they have not yet been integrated into mainstream software applications. The Java Speech API , which is being developed by sun and several other companies, will bridge this gap and make speech capabilities standard features in Java applications.
One of the most common devices that we use to speak and listen is the telephone. Mobile devices are being developed ,which can integrate computer and telephone capabilities. The java Telephony API is designed to incorporate telephony features into java applications. This API let you place and answer calls from within a java applications, provide touch-tone navigation, and manage multiple telephone connections. A number of advanced telephony capabilities are also being planned.

The Java Speech API provides the capability to incorporate speech technology (both input and output) into Java applets and applications. When it becomes available, it supports speech-based program navigation, speech-to-text translation, and speech synthesis. The Java Speech API is being developed by Sun in collaboration with IBM,AT&T, Texas Instruments, Philips , Apple and other companies.

2.1 Speech Recognition

Speech recognition allows computers to listen to a user's speech and determine what the user has said. It can range from simple, discrete command recognition to continuous speech translation .Although speech recognition has made much progress over the last few years, most recognition systems still make frequent errors. These errors can be reduced

by using better microphones , reducing background noise, and constraining the speech recognition task .

Speech recognition constraints are implemented in term of grammars that limit the variety in user input. The JSGF provides the capability to specify grammar's rule, Which are used for speech recognition system that are command and control-oriented. These systems only recognize speech as it pertains to it's program operation and do not support general dictation capabilities.

Event with the constraints posed by grammars, errors still occur and must be corrected. Almost all applications that employ speech recognition must provide error correction facilities.

2.2 *Speech Synthesis*

Speech synthesis is the opposite of speech recognition. It allows computer to generate spoken output to users. It can take the form of bulk text-to-speech translation, or of intricate speech-based responses that are integrated into an application's interface.

Speech synthesis systems must satisfy the two main requirements of understandability and naturalness.

Understandability is improved by providing adequate pronunciation information to speech generators. This eliminates guesses on the port of the speech synthesizer. JSML is used to provide pronunciation information, as required. Naturalness is improved by using a non-mechanical voice and manning emphasis, intonation, phrasing, and pausing. JSML also provides markup capability that control these speech attributes.

When you're synthesizing speech, it is often desirable to select attributes of the voice that is generated. For example, you might want to choose between male and female voice or old and young voices. The speech API provides control over these features. In addition, text that is to be synthesized can be marked up with event markers that cause events to be generated as they are processed.

2.3 The Java Telephony API

The Java Telephony API (JTAPI) is a set of APIs that provide telephony capabilities for Java applications. It supports basic telephony capabilities. Such as call placement and call answering and advanced capabilities ,such as call centers and media stream. JTAPI provides both direct control over telephony resources and indirect access through networked resources .This mean that you can create server applications through these resources.

Speech Recognition is a vast research field in itself and it requires lot of hardwork, dedication and expertise to reach to its performance climax. This is ,because, every person in this world has its own different accent, tone and pitch of voice. Furthermore, there are very few who are perfect in pronounciation and grammar. So in speech recognition, error correction is another big problem .

Although, we searched material on this field and checked performance of speech Interfacing softwares in the market like Dragon Naturally speaking, Windows ME 2000 built in capabilities, but it was not satisfactory. We also tried JAVA packages to handle this problem, but it was not user friendly. So we studied dial pulses and DTMF tones to accomplish our project.

2.3 Telephony Applications

some typical telephony applications include:

2.3.1 Voice Mail or Answering Machine Software.

Most users are familiar with "Voice mail" or computerized answering machine software.

The pieces of software allow users to call into a computer and access audio messages that have been left for them. Voice mail and answering machine software programs are often extended to E-mail, address books, and other type of data. Automated exchange can be programmed in this kind of telephony applications.

2.3.2 Accessing databases.

Large number of telephony applications allow users to access databases such as movie listing, stock quotes, or news.

2.3.3 Call Routing.

Many of the same telephony applications that provide voice mail or database access also allows incoming calls to be routed to other phone line. Because most contemporary call routing systems rely on DTMF(touch-line) to rout the call they ask for an extension number, but with speech recognition this could be very recommendable.

2.3.4 MultiLine Applications

Most telephony applications are designed to handle several phone lines comming into the same PC.Multi line telephony applications need to be designed to handle the multiple input channels in such a way that one channel doesn,t slow down or harm another channel.

The easiest multi-line application has one process running at least one thread per phone line. Because each line has its own thread, the lines are independent and (generally)one line will not cause another line to slow down. Multi-threaded lines also allow for improved performance on multi-processor CPUs.

The most stable multi telephony design is to have one process per phone line.This insures that one phone line cannot crash and pull down the other lines.It also parallelizes well. It is more difficult to code. Presently this exchange is designed for one subscriber at a time . Special devices are available in the market that allow up to 1**6** incoming lines.

Chapter-3

3. Database Development.

3.1 DR Codd's 13 points

3.1.1 file based system.

1. A collection of application program that perform services for the end user.
2. Each program defines and manages its own data.

1. **Drawbacks of file based approach**

1. Separation and isolation of data.
2. Duplication of data.
3. Data dependence.
4. Incompatible file formats.
5. Fixed queries.

3.1.2 Database approach.

1. **Database.**

A shared collection of logically related data designed to meet the information needs of an organization.

2. **DBMS**.

A software that enables users to define,creat and maintain database ,which provides controlled access to this database.

1. **Characteristics of DBMS**.
   1. Data definition language(DLL)
   2. Data manipulation language(DML)
   3. A security system.
   4. An integrity system.
   5. A Concurrency controlled system.
   6. A recovery control system.
   7. A user accessible catalog.
   8. Complex relationship b/w data.
   9. Tight control of data redundancy.

10.Centralised data dictionary.

11.Ensures that data can be shared across applications.

2. Components of DBMS.

1.Hardware .
2.Software .
3.Data.
4.Procedures.
5.People.

3. Roles of database environment.

1.Data and Database administrators.
2. Database designers.
3. Application programmers.
4. End users(native and sophisticated).

4. Advantages of Database systems.

1.Control of Data redundancy.
2.Data consistency.
3.More information from same data.
4.Sharing of Data.
5.Data Integrity.
6.Improved security.
7.Enforcement of standards.
8.Economy of scale.
9.Balanced conflicting requirements.
10.Data accessibility and responsiveness.
11.Increased productivity.
12.Improved maintenance.
13.Increased concurrency.
14.Back up and recovery services.

5. Disadvantages.

1.Complexity.
2.Size.
3.Cost of DBMS.
4.Hardware cost.
5.Cost of conversion.
6.Performance.

7.Higher impact of failure.

6. Functions of DBMS.

In 1982,Codd represented fol 10 functions of DBMS:

1.Data retrieval,storage and update.
2.A user accessible catalogue.
3.Transaction support.
4.Concurrency control services.
5.Recovery services.
6.Authorisation services.
7.Support for data communication.
8.Integrity services.
9.Services to promote data independence.
10.Utility services.

DR E.F CODD'S 13 Points.

Reorganised into five functional areas.

3. Foundational rules(0,12).
4. Structural rules(1,6).
5. Integrity rules(3,10).
6. Data Manipulation Rules(2,4,5,7).
7. Data Independence Rules(8,9,11).

3.1.3 Foundation Rules.

1. Rule 0(Foundational rule).
2. Rule12(Non-subversion rule).

3.1.4 Structural rules.

1. Rule1(Information representation).
2. Rule6(view updating).

3.1.5 Integrity rule.

1. Rule3(Systematic treatment of null values).
2. Rule10(Integrity Independence).

3**.1.6 Data Manipulation Rules.**

1. Rule2(guaranteed access).
2. Rule4(Catalog Based relational model).
3. Rule5(comprehensive data sub-language).

4. Rule7(High level insert,delete,update).

3.1.7 Data Independence rules.

1. Rule8(Physical data independence).
2. Rule9(Logical data independence).
3. Rule11(Distribution independenc).

3.2 Database languages.

1.Data definition languages(DDL).
2.Data manipulation languages(DML).
3.4th Generation(4GL).

3.2.1 DDL.

A descriptive language that allows the DBA or user to describe and name the entities,required for the application and the relationship that may exist b/w the different entities.

3.2.2 DML.

A language that provides a set of operations that supports the basic data manipulation operations on the data held in the basic database.

3.2.3 4GL.

Types.

1. Form generation .
2. Report generation.
3. Graphics generation.
4. Application generation.

3.3 Data Models.

An integrated collection of concepts for describing data, relationships b/w data and constraints on the data in an organization.

Types.

1. Object Based.
2. Physical Data Models.

3. Record based.

3.3.1 Object based data models.

Types.

1.Entity relationship.
2.Semantic relationship.
3.Functional relationship.
4.Object oriented.

3.3.2 Physical Data Models.

Types.

1.Unifying model.
2.Frame memory.

3.3.3 Record based data models.

Types.

1.Hierarchical data model.
2.Network data model.
3.Relational data model.

1. Hierarchical Data Model.

A model comprising record stored in a general tree structure.There is no one root record type,which has zero or more dependent records types.

2. Network Data Model.

A model comprising records,data items and (1:m)association b/w records.

3. Relational Data Model.

It based on the concept of mathematical relations.In this model,data and relationships are represented as tables,each of which has a number of columns with a unique name.

3.4 Data dictionary.

A repository of information describing the data in the data base i.e the MATA DATA or “Data about Data”.

3.5 ER-MODEL.

A detailed, logical representation of the entities, associations and data elements for an organization or business area.

3.5.1 ER-DIAGRAM.

A graphical representation of an ER-Model.

3.5.2 Entity.

It corresponds to a table in relational environment.

3.5.3 Attribute.

A named property or characteristics of an entity that is of interest to the organization.

3.5.4 Relationship.

An association b/w the instances of one or more entity types.

3.5.5 Normalization.

The process of converting complex data structures into simple stable data structures.

3.5.5.1 1NF.

A relation that contains no repeating groups.

3.5.5.2 2NF.

If it is already in 1NF.
Every non-key attribute is fully functionally dependent on Pk.

3.5.5.3 3NF.

If it is already in 2NF
There is no transitive dependency.
Additional Normal forms are:
-boyce codel NF
-4NF
-5NF

Chapter-4

4.Modems

The modem is a peripheral device for computers which allows two computers to communicate over standard phone lines. We use our modems to connect to an Internet Service Provider (ISP),through telephone lines and the ISP connects us into the Internet. The Internet lets us connect to any machine in the world.

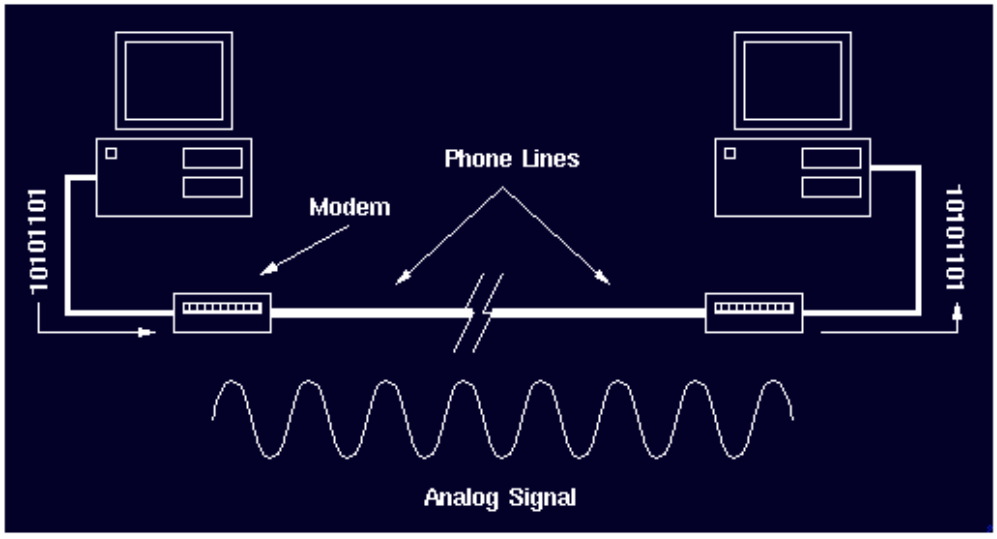
TECHNICAL DEFINITION

The word modem is a contraction of the words modulator –demodulator .A modem is typically used to send digital data over a phone line . The sending modem modulates the data into a signal that is compatible with the phone line, and the receiving modem demodulates the signal back into digital data. Wireless modems are also frequently seen converting data into radio signals and back. Modems came into existence in the 1960s as a way to allow terminals to connect to computers over the phone lines .

4.1 Modem Internal Composition

4.1.1 UART

Modem is basically comprised of an UART (universal asynchronous receiver / transmitter) chip which is responsible for performing the main task in serial communications with computers. The device changes incomming parallel information to serial data which can be



sent on a communication line. A second UART can be used to receive the information. The UART performs all the tasks, timing, parity checking, etc. needed for the communication. The only extra devices attached are line driver chips capable of transforming the TTL level signals to line voltages and vice versa.

To use the device in different environments, registers are accessible to set or review the communication parameters. Setable parameters are for example the communication speed, the type of parity check, and the way incomming information is signalled to the running software.

4.1.1.1 **UART TYPES**

Serial communication on PC compatibles started with the 8250 UART in the XT. In the years after, new family members were introduced like the 8250A and 8250B revisions and the 16450. The last one was first implemented in the AT. The higher bus speed in this computer could not be reached by the 8250 series. The differences between these first UART series were rather minor. The most important property changed with each new release was the maximum allowed speed at the processor bus side.

The 16450 was capable of handling a communication speed of 38.4 kbs without problems. The demand for higher speeds led to the development of newer series which would be able to release the main processor from some of its tasks. The main problem with the original series was the need to perform a software action for each single byte to transmit or receive. To overcome this problem, the 16550 was released which contained two on-board FIFO buffers, each capable of storing 16 bytes. One buffer for incomming, and one buffer for outgoing bytes.

A marvellous idea, but it didn't work out that way. The 16550 chip contained a firmware bug which made it impossible to use the buffers.

The 16550A which appeared soon after, was the first UART which was able to use its FIFO buffers. This made it possible to increase maximum reliable communication speeds to 115.2 kbs. This speed was necessary to use effectively modems with on-board compression. A further enhancment introduced with the 16550 was the ablity to use DMA, direct memory access for the data transfer. Two pins were redefined for this purpose. DMA transfer is not used with most applications. Only special serial I/O boards with a high number of ports contain sometimes the necessary extra circuitry to make this feature work.

The 16550A is the most common UART at this moment. Newer versions are under development, including the 16650 which contains two 32 byte FIFO's and on board support for software flow control. Texas Instruments is developing the 16750 which contains 64 byte FIFO's.

4.1.1.2 REGISTERS

The communication between the processor and the UART is completely controlled by twelve registers. These registers can be read or written to check and change the behaviour of the communication device. Each register is eight bits wide. On a PC compatible, the registers are accessible in the I/O PORT map.The function of each register will be discussed here in detail.

1. RBR:Receiver Buffer Register(RO)

The receiver buffer register contains the byte received if no FIFO is used, or the oldest unread byte with FIFO's. If FIFO buffering is used, each new read action of the register will yield the next byte, until no more bytes are present. Bit 0 in the line status register can be used to check if all received bytes have been read. This bit wil change to zero if no more bytes are present.

2. THR:Transmitter Holding Register(WO)

The transmitter holding register is used to buffer outgoing characters.

If no FIFO buffering is used, only one character can be stored. Otherwise the amount of characters depends on the type of UART. Bit 5 in the line status register can be used to check if new information must be written to the transmitter holding register. The value 1 indicates that the register is empty. If FIFO buffering is used, more than one character can be written to the transmitter holding register when the bit signals an empty state. There is no indication of the amount of bytes currently present in the transmitter FIFO.

The transmitter holding register is not used to transfer the data directly. The byte is first transferred to a shift register where the information is broken in single bits which are sent one by one.

3. IER:Interrupt enable register(R/W)

The smartest way to perform serial communications on a PC is using interrupt driven routines. In that configuration, it is not necessary to poll the registers of the UART periodically for state changes. The UART will signal each change by generating a processor interrupt. A software routine must be present to handle the interrupt and to check what state change was responsible for it.
Interrupts are not generated, unless the UART is told to do so. This is done by setting bits in the interrupt enable register.

4. IIR:Interrupt identification Register(RO)

An UART is capable of generating a processor interrupt when a state change on the communication device occurs. One interrupt signal is used to call attention. This means, that additional information is needed for the software before the necessary actions can be performed. The interrupt identification register is helpful in this situation. Its bits show the current state of the UART and which state change caused the interrupt to occur.

5. FCR:FIFO Control Register(W/O)

The FIFO control register is present starting with the 16550 series.

This register controls the behaviour of the FIFO's in the UART. If a logical value 1 is written to bits 1 or 2, the function attached is triggered. The other bits are used to select a specific FIFO mode.

6. LCR: Line control Register(R/W)

The line control register is used at initialisation to set the communication parameters. Parity and number of data bits can be changed for example. The register also controls the accessibility of the DLL and DLM registers.
The line control register is used at initialisation to set the communication parameters. Parity and number of data bits can be changed for example. The register also controls the accessibility of the DLL and DLM registers. These registers are mapped to the same I/O port as the RBR, THR and IER registers. Because they are only accessed at initialisation when no communication occurs. This register swapping has no influence on performance.

Some remarks about parity:

The UART is capable of generating a trailing bit at the end of each dataword which can be used to check some data distortion. Because only one bit is used, the parity system is capable of detecting only an odd number of false bits. If an even number of bits has been flipped, the error will not be seen.

When even parity is selected, the UART assures that the number of high bit values in the sent or received data is always even. Odd parity setting does the opposite. Using stick parity has very little use. It sets the parity bit to always 1, or always 0.

Common settings are:

8 data bits, one stop bit, no parity
7 data bits, one stop bit, even parity

7. MCR : Modem Control Register(R/W)

The modem control register is used to perform handshaking actions with the attached device. In the original UART series including the 16550, setting and resetting of the control signals must be done by software. The new 16750 is capable of handling flow control automatically, thereby reducing the load on the processor.

The two auxiliary outputs are user definable. Output 2 is sometimes used in circuitry which controls the interrupt process on a PC. Output 1 is normally not used, however on some I/O cards, it controls the selection of a second oscillator working at 4 MHz. This is mainly for MIDI purposes.

8. LSR : Line status register(RO)

The line status register shows the current state of communication. Errors are reflected in this register. The state of the receive and transmit buffers is also available.

Bit 5 and 6 both show the state of the transmitting cycle. The difference is, that bit 5 turns high as soon as the transmitter holding register is empty whereas bit 6 indicates that also the shift register which outputs the bits on the line is empty.

9. MSR:Modem status register(RO)

The modem status register contains information about the four incomming modem control lines on the device. The information is split in two nibbles. The four most siginificant bits contain information about the current state of the inputs where the least significant bits are used to indicate state changes. The four LSB's are reset, each time the register is read.

10. SCR: Scratch register(R/W)

The scratch register was not present on the 8250 and 8250B UART's. It can be used to store one byte of information. In practice, it has only limited use. The only real use is checking if the UART is a 8250/8250B, or a 8250A/16450 series. Because the 8250 series are only found in XT's even this use of the register is not commonly seen anymore.

11/12. DLL and DLM: Division latch register(R/W)

For generating its timing information, each UART uses an oscillator generating a frequency of about 1.8432 MHz. This frequency is divided by 16 to generate the time base for communication. Because of this division, the maximum allowed communication speed is 115200 bps. Modern UARTS like the 16550 are capable of handling higher input frequencies upto 24 MHz which makes it possible to communicate with a maximum speed of 1.5 Mbps. On PC's higher frequencies than the 1.8432 MHz are rarely seen because this would be software incompatible with the original XT configuration.

This 115200 bps communication speed is not suitable for all applications. To change the communication speed, the frequency can be further decreased by dividing it by a programmable value. For very slow communications, this value can go beyond 255. Therefore, the divisor is stored in two seperate bytes, the DLL and DLM which contain the least, and most significant byte.

For error free communication, it is necessary that both the transmitting and receiving UART use the same time base. Default values have been defined which are commonly used.

4.2-WHY MODULATE
CONVERTING DIGITAL SIGNALS TO ANAOLOG SIGNALS

You can't simply transmit bits directly across telephone lines. It first needs to be converted into sound waves. If you've ever picked up your telephone while you were using your modem, you've no doubt heard a constant loud, screeching noise. This noise is the sound of digital information after it's been converted to analog sound waves.

4.2.1 ADVANTAGES OF PASSBAND MODULATION

In passband modulation the desired information signal modulates a sinosoid called a carrier wave. A carrier signal is a single frequency that is used to carry the intelligence (data). For digital, the intelligence is either a 1 or 0. When we modulate the carrier, we are changing its characteristics to correspond to either a 1 or 0.

1-Digital symbols are transformed into waveforms that are compatible with the characteristics of the channel. The voice frequency channels of the general switched telephone network are used for the transmission of digital data.To use these channels,the data must be put in a form that can be sent over a limited bandwidth line.In voice grade telephone networks, transformers ,carrier systems and loaded lines attenuate all signals below 300 Hz and above 3400 Hz.

While the bandwidth from 300 Hz to 3400 Hz is fine for voice transmission,it is not possible for transmission of digital data because the data has many frequency components above this range. To transmit data through phone lines,it is necessary to convert digital data into the signal that is totally within the frequency range.

In case of modems which are connected by telephone lines whose characteristic bandwidth is 0-3000Hz we choose carriers whose frequencies lie in this range.

Modems constantly transmit a **carrier signal**. The carrier signal's function is, basically, to "carry" information from one place to another.

The analytical expression for the carrier is

$$\text{Carrier}(t)=A\cos[\varpi\, t+\text{phase}(t)]$$

Where

A=amplitude of carrier

$\varpi=(2\pi f)$=frequency of carrier in radians per second

1. phase(t)=phase of carrier

Modem transmit information by varying, or modulating, this carrier signal.

The amplitude, phase and Frequency of carrier is modulated by the message signal.

4.2.2-DIFFERENT MODULATION SCHEMES

As modulation is the basic theme in a modem it is necessary to study different modulation schemes to better understand the modem functions.

We have two types of modulation:

Coherent modulation.

Non-coherent modulation.

In coherent modulation phase information is available whereas phase reference is not required in non coherent modulation. Further in coherent and non coherent modulation we have;

4.2.2.1-Binary Modulation Schemes

4.2.2.2-M-ary Modulation Schemes

4.2.2.1-BINARY MODULATION SCHEMES:

1.AMPLITUDE SHIFT KEYING:

For AM the general analytical expression is

$$Si(t)=\sqrt{2Ei(t)/T}\cos(\varpi_o t+\text{phase}) \qquad 0<=t<=T, \quad i=1,\ldots\ldots,M$$

Where the amplitude term, $\sqrt{2Ei(t)/T}$ will have M discrete values

Consider the following figure for binary amplitude modulation

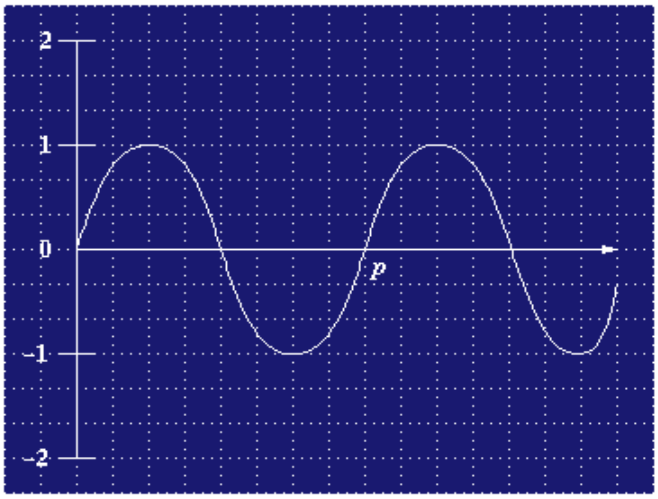


Figure 1.2 amplitude modulation (a)

The graph above is of some random signal, with an **amplitude**, or strength, of 1. This signal has a **period** of *p* (a **period** is the amount of time before a wave repeats itself). We might let a signal with amplitude 1 represent a binary 0.

Figure 1.3 amplitude modulation(b)

This is the same sound wave as above , because it has the same period *p*. However, this signal has a higher amplitude of 2. We might let a signal with amplitude 2 represent a binary 1.

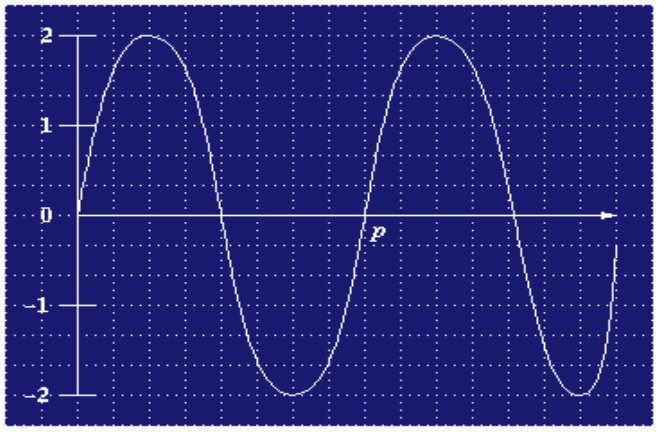
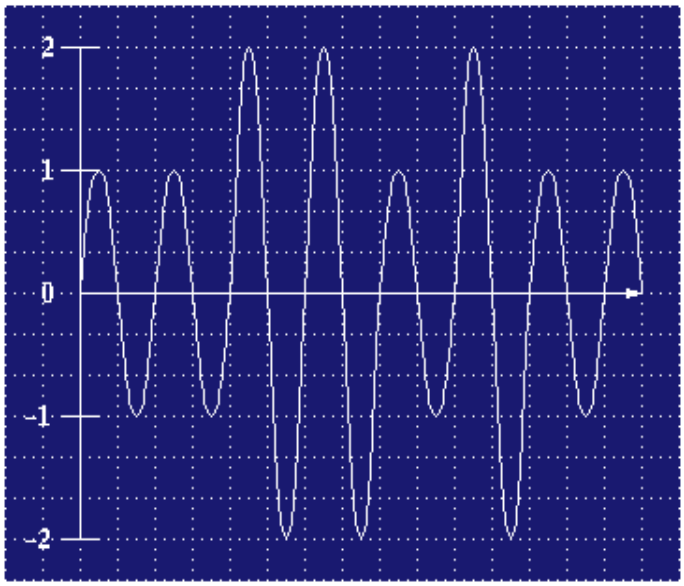


Figure 1.4 amplitude modulation (c)



This is what **amplitude modulation** looks like. Notice that the period is the same for the entire signal, and that only the amplitude varies.
Assuming amplitude 1 = binary 0, and amplitude 2 = binary 1, this signal would represent 0011010.

Advantages:

- Simple to design.

Disadvantages:

- Noise spikes on transmission medium interfere with the carrier signal.
- Loss of connection is read as 0s.

2-FREQUENCY SHIFT KEYING:

In frequency modulation information is contained in the frequency of the carrier
For FM the general analytical expression is

$$Si(t)=\sqrt{2E(t)/T}\cos(\varpi_i t+\text{phase}) \qquad 0<=t<=T$$

–

i=1,…….,M

Where the frequency term, **Wi** will have M discrete values
For binary FSK, consider the following figure

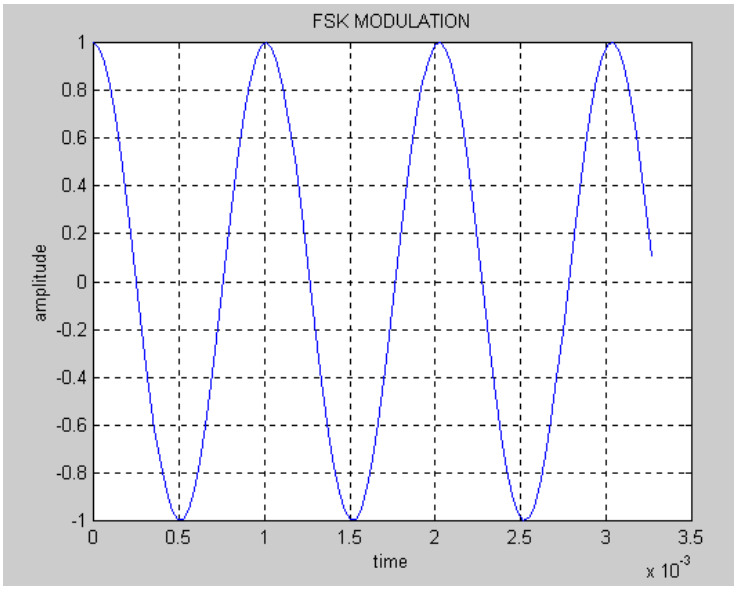
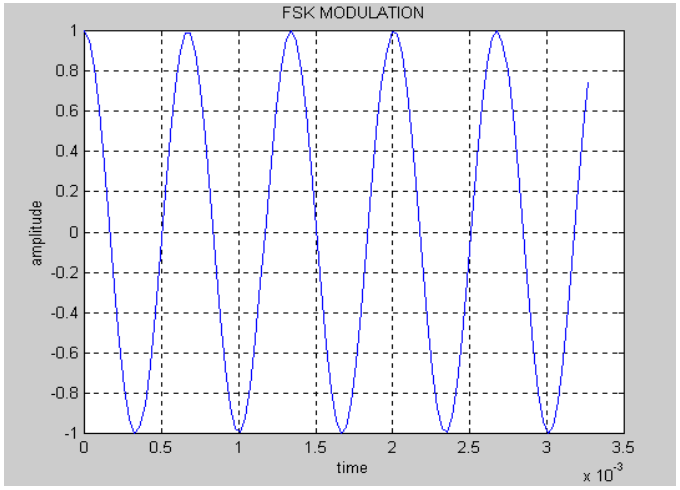


Figure 1.5 frequency modulation(a)

We let the signal above with a certain frequency represent binary 1

Figure 1.6 frequency modulation(b)



Above figure shows a signal representing binary 0

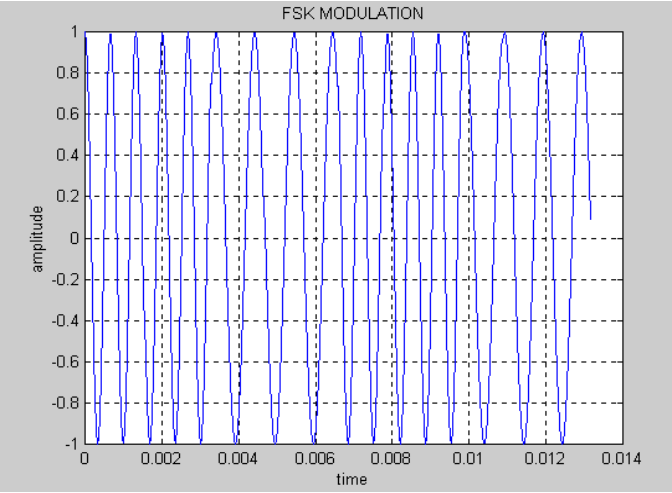


Figure 1.7 frequency modulation(c)

This is what frequency modulation looks like.Digital information contained in above analog signal is 0 1 0 1

Advantages:

- Immunity to noise on transmission medium.
- Always a signal present. Loss of signal easily detected.

Disadvantages:

- Requires 2 frequencies .
- Detection circuit needs to recognize both frequencies when signal is lost.

3-PHASE SHIFT KEYING:

In phase modulation information is contained in the phase of the carrier
For PSK the general analytical expression is

$$Si(t)=\sqrt{2E(t)/T}\cos(\varpi t+(phase)_i t)) \qquad 0<=t<=T$$

$$i=1,\ldots\ldots,M$$

Where the phase term, $(phase)_i(t)$ will have M discrete values
for Bianry PSK M will have 2 values.

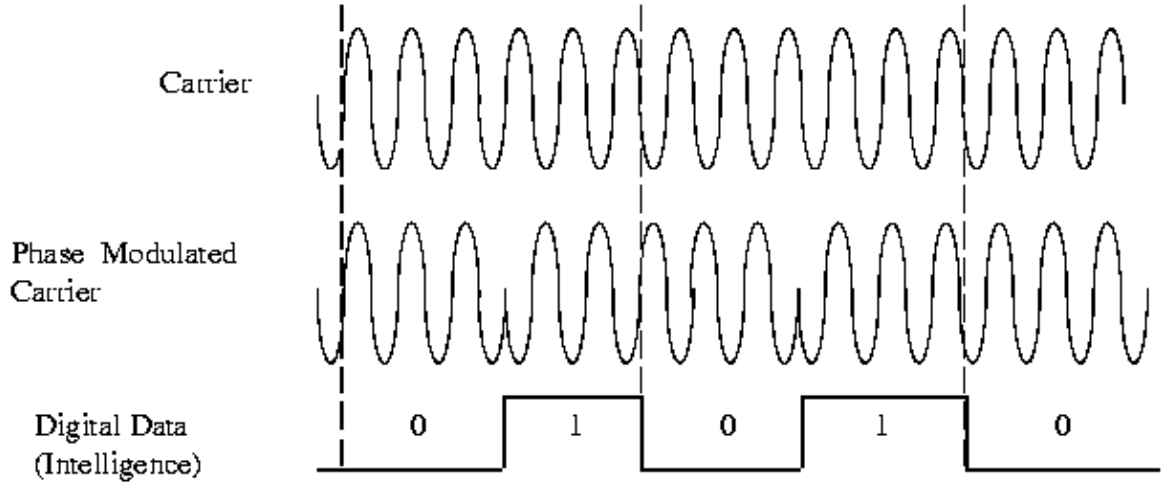


Figure 1.8 phase modulation

The carrier phase is switched at every occurrence of a 1 bit, but remains unaffected for a 0 bit. The phase of the signal is measured relative to the phase of the preceding bit. The bits are timed to coincide with a specific number of carrier cycles (3 in this example = 1 bit).
Advantages:

- Only 1 frequency used
- Easy to detect loss of carrier

Disadvantages:

- Complex circuitry that is required to generate and detect phase changes.

4.2.2.2-M-ARY MODULATION SCHEMES:

1-QUADRATURE PHASE SHIFT KEYING(QPSK):

Quadrature Phase Shift Keying employs shifting the phase of the carrier at a 600 baud rate puls an encoding technique. The originate modem transmits at 1200 Hz, and receives on 2400 Hz. The answer modem receives on 1200 Hz, and transmits on 2400 Hz.
The digital information is encoded using 4 (Quad) level differential PSK at 600 baud.

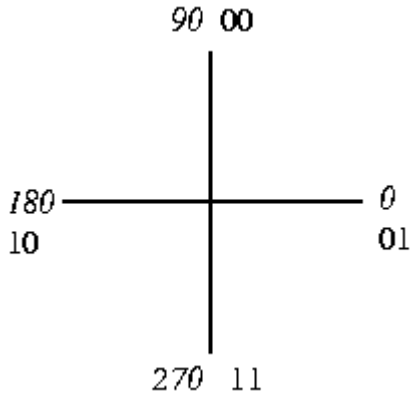
Here we need to explain the relationship between **baud** and **bits** .These two terms are used interchangeably. However, the two are not the same at all. The carrier signal is characterized by the number of signal intervals, or pulses, that are transmitted per second. Each pulse is called a **baud**.
Bps stands for **b**its **p**er **s**econd. Bps is a measure of how many bits can be transmitted during one pulse (one baud). So,

```
bps = baud * number of bits per baud.
```

The two are often confused because early modems used to transmit only 1 bit per baud, so a 1200 baud modem would also be transmitting 1200 bps. These days, we need higher speeds. But for two-way communications, the baud limit is 1200 baud. So the technique is to try and "pack" as many bits as you can into 1 baud.
A modem operating at 9600 bps is still only transmitting at 1200 baud. But it is "packing" 8 bits into each baud:

```
9600 bps = 1200 baud * 8 bits per baud
```

The four phases are



The data is encoded as follows:

DIBIT	Phase Shift
00	+90
01	0
10	180
11	270

For every change in the baud rate (phase shift), we can decode 2 bits. This leads to the following:
2 bits x 600 baud = 1200 bps
Example of Carrier Phase Modulation:

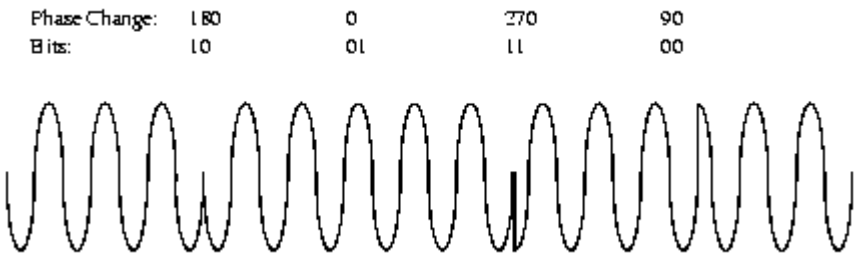


Figure 1.9 QPSK

2-M-ARY QUADRATURE AMPLITUDE MODULATION:

Quadrature Amplitude Modulation refers to QPSK with Amplitude Modulation. Basically, it is a mix of phase modulation and amplitude modulation. QAM phase modulates the carrier and also modulates the amplitude of the carrier.
Phase Modulated and Amplitude Modulated Carrier:

Figure 1.10 QAM

There are two types, 8-QAM and 16-QAM. 8-QAM encodes 3 bits of data (2 for every baud and 16-QAM encodes 4 bits of data for every baud. 8-QAM transfers 4800 bps and 16-QAM transfer9600 bps.

16-QAM has 12 phase angles, 4 of which have 2 amplitude values! 16-QAM changes phase with every baud change.

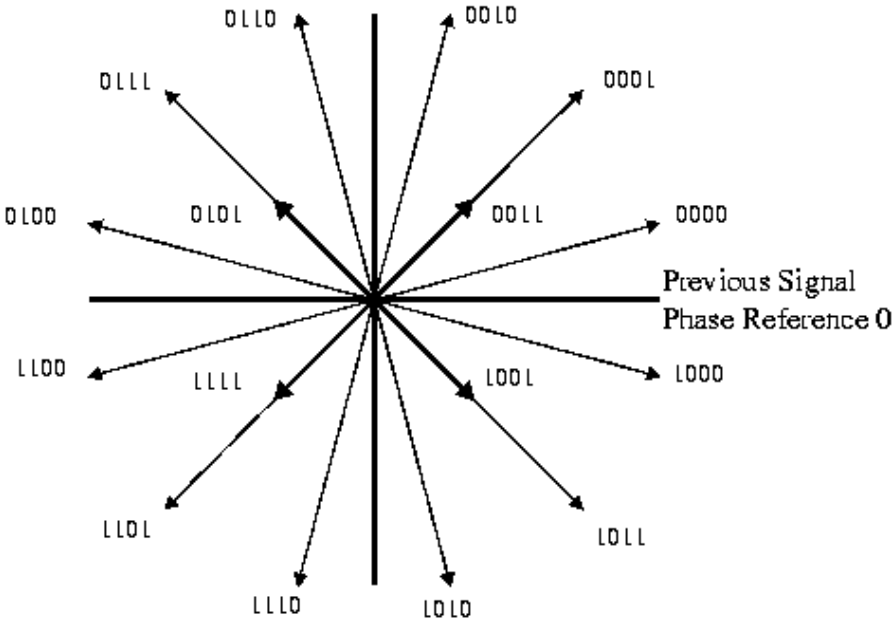


Figure 1.11- 16-Qam phasor diagram

Higher transfer rates use much more complex QAM methods.
Similarly there are other modulation techniques like M-ary ASK, Trellis coded modulation etc that are used according to the design requirements to modulate the carrier signal.

(5) (9) (1)

—

For automated Inquiry, we need a voice modem with a high speed (56 kbps is recommended) at the PC used as exchange.This modem is the service point between exchange's telephony module and the user through a normal touch-tone phone.It is assumed that the whole system is functioning in an ideal intelligent Network.

4.3 The Intellegent Network

The PSTN is an intelligent network throughout much of the world. In practical terms, this means that the network has the capacity to utilize real-time database interactions to control the routing of telephone calls.Many of the services that modem telephone users expect, rely upon this capability. As mentioned earlier the environment in which automated exchanges will be working is supposed to be an intelligent Network.This supposition is not far from the fact as well because almost 90% of data lines in Pakistan are on fibre optics and controlled by digital switches of PTCL.

Chapter-5

5.Telephone line

The parameters like voltages, currents, resistances and frequencies may vary from one country to another, or from one exchange to another.

The telephone line at telephone jack point connects to a telephone exchange. This exchange is either a public exchange, or an electrically equivalent private business exchange. These lines are often called POTS (Plain Old Telephone System) lines. Some private exchanges use special or intelligent telephones - these often use non-standard proprietary interfaces which are not equivalent to a POTS line.

The exchange provides a line with certain characteristics, and a telephone, fax machine, modem, etc (called an *appliance* in this document) has electrical characteristics which allow it to perform the normal operations of going off-hook (busy), dialling, monitoring call progress, sending sound to the other party (the callee), receiving sound from the callee, going on-hook (idle), ringing, and receiving calling line identification.

The line to the exchange has two wires, known for historical reasons as "tip" and "ring" ("ring" does not relate to the ringing of the telephone).

5.1 Line states

If nothing is connected to a telephone line, the exchange will regard the line as on-hook (idle). The exchange provides a DC "idle line" voltage, norminally 48V (typically between 45V and 50V), which appears across the line. You can measure this by connecting a voltmeter across the line. The polarity of this voltage is not defined in relation to the telephone line connector, and might change if lines are re-wired or changes are made at the exchange. For this reason, any device connected to the line should be designed to operate correctly regardless of the polarity of the line voltage. An appliance, when on-hook (idle), must appear as an open circuit (or nearly an open circuit) to the telephone line. In other words, it must draw no current, or very little current.

As current is drawn from the line, the line voltage drops, until at a certain current, the exchange regards the line as off-hook (in use) and transmits a dial tone down the line. In this state the appliance (telephone, fax etc) has a resistance of a few hundred ohms, and draws about 25 to 40 milliamps (up to 80 milliamps in unusual cases) from the line, dragging the line voltage (measured at the appliance) down to typically 5 to 10 volts. The appliance can then place a call, using pulse dialling (also called "loop disconnect" or "LD" dialling) or DTMF (dual tone multiple frequency) dialling (also called "touch-tone" dialling) to inform the exchange of the number to be dialled. When the appliance goes back on-hook, the exchange regards the line as being in the idle state again, and terminates any call that was in progress.

A ring signal is generated by the exchange to indicate an incoming call. It consists of bursts of low frequency AC voltage across the line, typically 90 volts RMS at 20Hz. The normal idle line voltage (48V DC) may change, disappear, or reverse polarity during ringing. While AC ring voltage is present, the audible ringer (bell or warbler) on the appliance will emit its noise. The timing and pattern of the bursts of AC is known as the "ring cadence" and varies from country to country and also if the "distinctive ring" feature is used. Regardless of the ring cadence, the presence of a large AC voltage on the line can be interpreted as a ring signal of some kind. The exchange will only send a ring signal if the appliance is currently on-hook (idle).

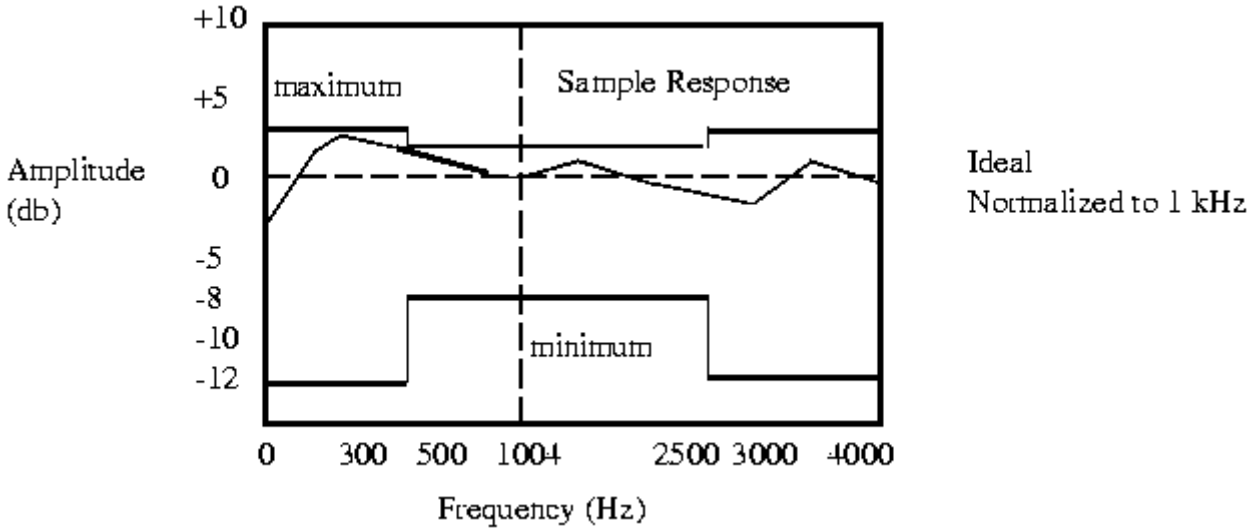
5.2 Telephone Line Characteristics

Telephone lines are not perfect devices due to their analog nature. The quality of the telephone line determines the rate that modulated data can be transferred. Good noise-free lines allow faster transfer rates (such as 14.4 kbps) while poor quality lines require the data transfer rate to be stepped down to 9600 bps or less. Phone lines have several measurable characteristics that determine the quality of the line.

- Attenuation Distortion
- Propagation Delay
- Envelope Delay Distortion

5.2.1 Attenuation Distortion

Attenuation Distortion is the change in amplitude of the transmitted signal over the Voice Band: it is the frequency response curve of the Voice Band.



Attenuation versus Frequency

To measure Attenuation Distortion, the phone line has a test frequency. This frequency is transmitted from 0 - 4 kHz into the line at a standard amplitude of 0 db. The loss of signal--or attenuation--is measured at the receiving end, and compared to a standard reference frequency of 1004 Hz.

Decibel (db) is a relative unit of measure. It is a log unit and a +3 db gain will indicate an amplitude of 2x the reference. It is a logarithmic ratio between input voltage and output voltage, calculated by the following formula:

db =10 x log (Vout/Vin)

The resulting information is graphed on an Attenuation vs. Frequency chart. Attenuation is a loss of signal amplitude (the receive signal is a smaller amplitude than the transmitted signal). It is indicated by a positive db. It's

also possible to have a signal appear at the receiving end, with a larger amplitude than when it started (this is indicated by negative db).

Attenuation occurs because the signal has to pass through many pieces of electronic equipment and transmission media. Some can amplify the signal (make it a larger amplitude) and some may attenuate the signal (make it smaller).

There are maximum and minimum acceptable limits for the Attenuation Distortion that is on phone lines. The Basic channel conditioning is as follows:

Frequency Range	Loss (db)
500 - 2500	-2 to +8
300 - 3000	-3 to +12

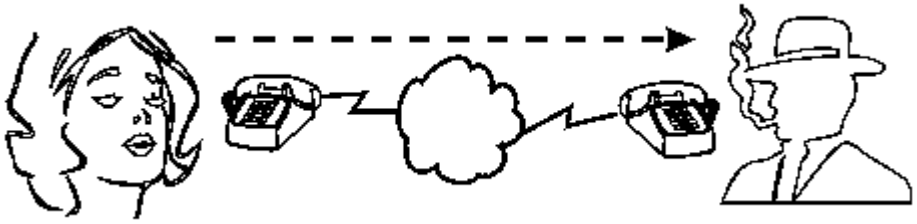
The above Loss is a range of acceptable values for the frequency range. In the Basic Channelling Conditioning, it is acceptable to have a loss in signal--in the frequency range of 500-2500 Hz--of "8 db loss to -2 db loss" (referenced to the amplitude at 1 kHz). Note that this is shown as -8db and +2 db (see the graph on the previous page) .

A +3 db attenuation is equal to -3 db in signal amplitude and a +8 db attenuation equates to -8 db in signal amplitude.

5.2.2 Propagation Delay

Signals transmitted down a phone line will take a finite time to reach the end of the line. The delay from the time the signal was transmitted to the time it was received is called Propagation Delay. If the propagation delay was the exact same across the frequency range, then there would be no problem. This would imply that all frequencies from 300 to 3000 Hz have the same amount of delay in reaching their destination over the phone line. They would arrive at the destination at the same time, but delayed by a small amount of propagation delay.

For example, this delay is heard when talking on long distance telephones. In this instance, we have to wait a little longer before we speak (to ensure that the other person hasn't already started to talk). Actually, all phone lines have propagation delay.

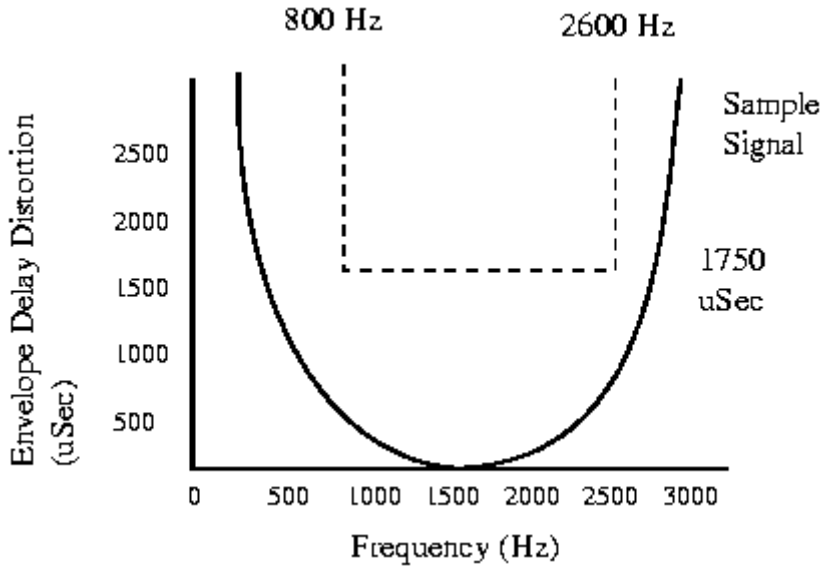


If the Propagation Delay is long enough, the modem or communications package may time-out and close the connection. In other words, it may think that the receive end has shut off!

5.2.3 Envelope Delay Distortion

If the Propagation Delay changes with frequency, we would then have the condition where the lower frequencies--such as 300 Hz-- may arrive earlier or later than the higher frequencies--such as 3000 Hz. For voice communication, this would probably not be noticeable. However, for data communication using modems, this could affect the phase of the carrier or the modulation technique that's used to encode the data.

When the Propagation Delay varies across the frequency range, we call this Envelope Delay Distortion. We measure propagation delay in microseconds (us), and the reference is from the worst case to the best case.

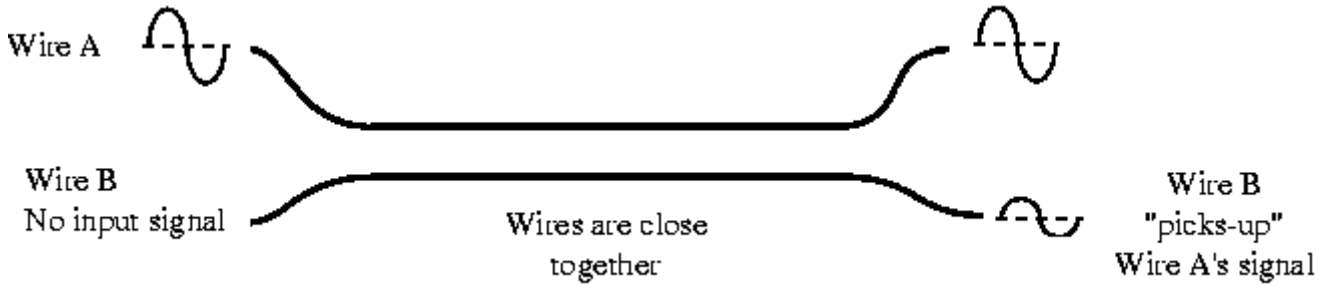


5.3 Line Impairments

Line Impairments are faults in the line that occur due to either improper line terminations or equipment out of specifications. These cannot be conditioned out, but can be measured to determine the amount of the impairment.

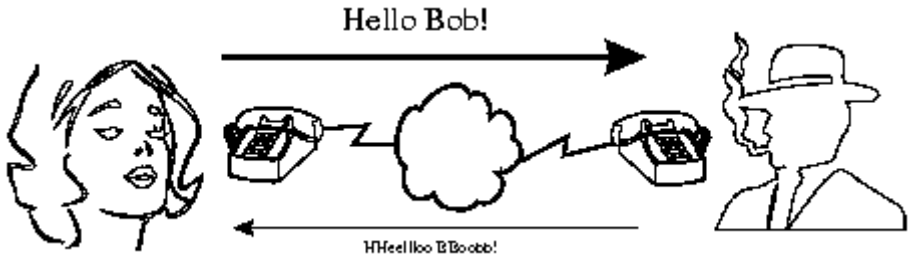
5.3.1 Crosstalk

Crosstalk is when one line induces a signal into another line. In voice communications, we often hear this as another conversation going on in the background. In digital communication, this can cause severe disruption of the data transfer. Cross talk can be caused by the overlapping of bands in a multiplexed system, or by poor shielding of cables running close to one another. There are no specific communication standards that are applied to the measurement of crosstalk.



5.3.2 Echo or Signal Return

All media have a preferred termination condition for perfect transfer of signal power. The signal arriving at the end of a transmission line should be fully absorbed, otherwise it will be reflected back down the line to the sender (and appear as an Echo). Echo Suppressors are often fitted to transmission lines to reduce this effect.

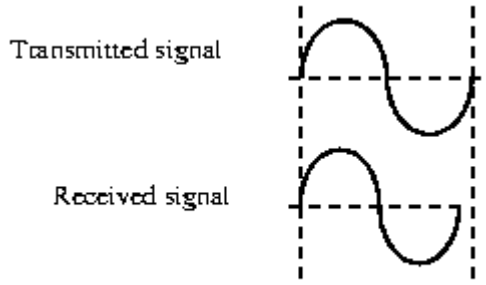


Usually during data transmission, these suppressors must be disabled or they will prevent return communication in full duplex mode. Echo suppressors are disabled on the phone line if they hear carrier for 400ms or more. If the carrier is absent for 100 mSec, the echo suppressor is re-enabled.

Echo Cancellers are currently used in Modems to replicate the echo path response. These cancellers then combine the results to eliminate the echo (thus, no signal interruption is necessary).

5.3.3 Frequency Shift

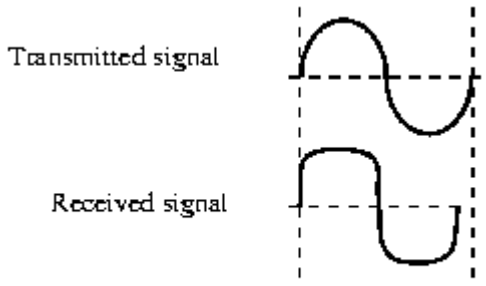
Frequency shift is the difference between the transmitted frequency and the received frequency. This is caused by the lack of synchronization of the carrier oscillators.



Frequency Shift

5.3.4 Nonlinear Distortion

Nonlinear distortion changes the wave shape of the signal. If the signal was transmitted as a sine wave (and arrived as a square wave), it would be an example of severe nonlinear distortion. Amplitude modulated carriers would suffer drastically if the original wave shape was distorted.



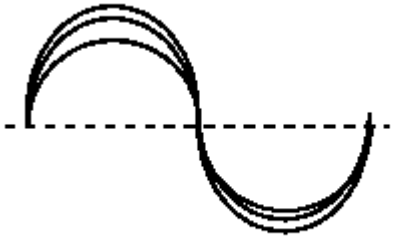
Non-linear Distortion

5.3.5 Jitter: Amplitude and Phase

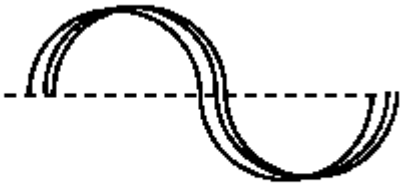
Here are the 2 types of Jitter:

1. Amplitude Jitter
2. Phase Jitter

Amplitude Jitter is the small constantly changing swing in the amplitude of a signal. It is principally caused by power supply noise (60 Hz) and ringing tone (20 Hz) on the signal.



Amplitude Jitter



Phase Jitter

Phase Jitter is the small constantly changing swing in the phase of a signal. It may result in the pulses moving into time slots that are allocated to other data pulses (when used with Time Domain Multiplexing).

Telephone company standards call for no more than 10 degrees between 20 and 300 Hz and no more than 15 degrees between 4 and 20 Hz.

5.4 Transients: Impulse Noise, Gain Hits, Dropouts & Phase Hits

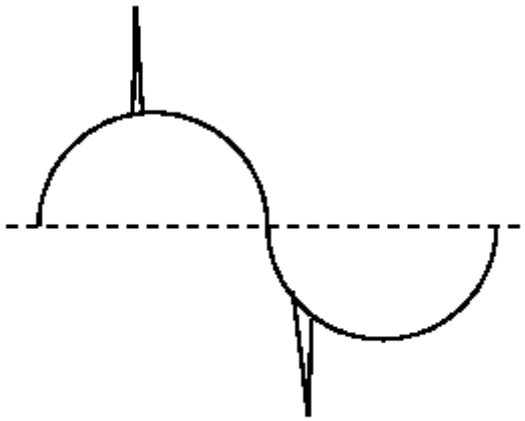
Transients are irregular-timed impairments. They appear randomly, and are very difficult to troubleshoot. There are 4 basic types of Transients.

1. Impulse Noise
2. Gain Hits
3. Dropouts
4. Phase Hits

5.4.1 Impulse Noise

Impulse noise is a sharp and quick spike on the signal that can come from many sources: electromagnetic interference, lightning, sudden power switching, electromechanical switching, etc..

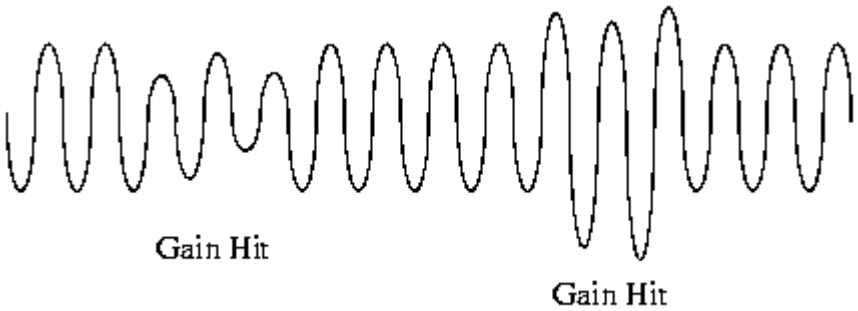
These appear on the telephone line as clicks and pops: they're not a problem for voice communication, but can appear as a loss of data (or even as wrong data bits) during data transfers. Impulse noise has a duration of less than 1 mSec and their effect is dissipated within 4 mSec.



Impulse Noise

5.4.2. Gain Hits

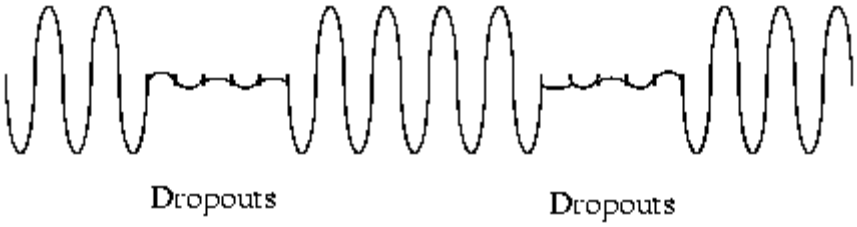
Gain Hits are sudden increases in amplitude that last more than 4 mSec. Telephone company standards allow for no more than 8 gain hits in any 15 minute interval. A gain hit would be heard on a voice conversation as if the volume were turned up for just an instance. Amplitude modulated carriers are particularly sensitive to Gain Hits.



5.4.3. Dropouts

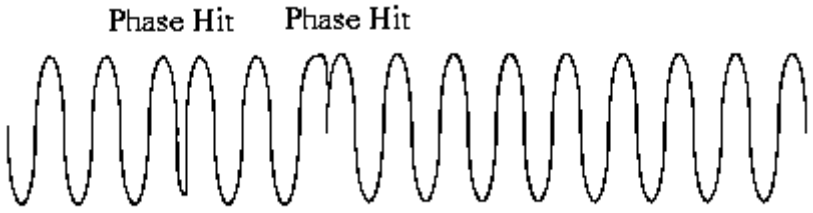
Dropouts are sudden losses of signal amplitude that are greater than 12 db, and last longer than 4 mSec.

They cause more errors than any other type of transients. Telephone company standards allow no more than 1 dropout for every 30 minute interval. Dropouts can be heard on a voice conversation (similar to call waiting), where the line goes dead for a 1/2 second. This is a sufficient loss of signal for some digital transfer protocols (such as SLIP), where the connection is lost and would then have to be re-established.



5.4.4 Phase Hits

Phase Hits are either a sudden--and large--change in the received signal phase (20 degrees), or a frequency that lasts longer than 4 mSec. Phase Hits generally occur when switching between Telcos, common carriers, or transmitters. FSK and PSK are particularly sensitive to Phase Hits. The data may be incorrect until the out-of-phase condition is rectified. The telephone company standard allows no more than 8 phase hits in any 15 minute period.



Chapter-6

6-RS-232 CHARACTERISTICS

6.1 RS-232 Signals

The number preceding each signal name correspond to the pin number defined in the standard

1. Protective Ground
2. Transmitted Data
3. Received Data
4. Request to Send
5. Clear to Send
6. Data Set Ready
7. Signal Ground
8. Received Line Signal Detect(Carrier Detect)
9. +P(for testing only)

10 -P(for testing only)

11. (unassigned)

12. Secondary Received Line Signal Detect
13. Secondary Clear To Send
14. Secondary transmitted Data
15. Transmission signal element timing

16. Secondary Received Data
17. Receiver Signal Element Timing
18. (unassigned)
19. Secondary Request To Send
20. Data Terminal Ready
21. Signal Quality Detector
22. Ring Indicator
23. Data Signal Rate Selector
24. Transmitter Signal Element Timing
25. (unassigned)

RS-232 Signals Functional Description

General: The first letter of the EIA signal name categorizes the signal into one of five groups, each representing a different "circuit":

- **A - Ground**
- **B - Data**
- **C - Control**
- **D - Timing**
- **S - Secondary channel**

- 1 Protective Ground

  .Name: AA

  .Direction:-

  .CCITT:101

  This pin is usually connected to the frame of one of the devices, either the DCE or the DTE, which is properly grounded.

The sole purpose of this connection is to protect against accidental electric shock and usually this pin should not be tied to signal ground.

This pin should connect the chassis (shields) of the two devices, but this connection is made only when connection of chassis grounds is safe (see ground loops below) and it is considered optional.

Ground loops are low impedance closed electric loops composed from ground conductors. When two grounded devices are connected together, say by a RS-232 cable, the alternating current on the lines in the cable induces an electric potential across the ends of the grounding line (either Protective Ground or Signal Ground), and an electric current will flow across this line and through the ground.

Since the loops impedance is low, this current can be quite high and easily burn out electric components. Electrical storms could also cause a burst of destructive current across such a loop. Therefore, connection of the Protective Ground pin is potentially hazardous. Furthermore, not all signal grounds are necessarily isolated from the chassis ground, and using a RS-232 interface, especially across a long distance, is unreliable and could be hazardous. 30 meters is considered the maximum distance at which the grounding signals can be connected safely.

- 2 Transmit Data

  . Name :BA
  . Direction:DTE->DCE
  . CCITT:103

  Serial data (primary) is sent on this line from the DTE to the DCE. The DTE holds this line at logic 1 when no data are being transmitted. A "On" (logic 0) condition must be present on the following signals,

where implemented, before data can be transmitted on this line : CA, CB,CC and CD (request to send,clear to send,data set ready,data terminal ready).

- 3 Receive Data

  . Name :BB
  . Direction:DTE <-DCE
  . CCITT:104

  Serial data (primary) is sent on this line from the DCE to the DTE. This pin is held at logic 1 (Mark) when no data are being transmitted, and is held "Off" for a brief interval after an "On" to "Off" transition on the Request To Send line, in order to allow the transmission to complete.

- 4 Request To Send

  .Name:CA
  .Direction:DTE->DCE
  .CCITT: 105

  Enables transmission circuits. The DTE uses this signal when it wants to transmit to the DCE. This signal, in combination with the Clear To Send signal, coordinates data transmission between the DTE and the DCE.

  A logic 0 on this line keeps the DCE in transmit mode. The DCE will receive data from the DTE and transmit it on to the communication link.

The Request To Send and clear to send signals relate to a half- duplex telephone line. A half duplex line is capable of carrying signals on both directions but only one at a time. When the DTE has data to send, it raises Request To Send, and then waits until the DCE changes from receive to transmit mode. This "On" to "Off" transition instructs the DCE to move to "transmit" mode, and when a transmission is possible, the DCE sets Clear To Send and transmission can begin.

On a full duplex line, like a hard-wired connection, where transmission and reception can occur simultaneously, the clear to send and Request To Send signals are held to a constant "On" level.

A "On" to "Off" transition on this line instructs the DCE to complete the transmission of data that is in progress, and to move to a "receive" (or "no transmission") mode.

- 5 Clear To Send

.Name :CB.
.Direction:DTE<-DCE.
.CCITT:106

An answer signal to the DTE. When this signal is active, it tells the DTE that it can now start transmitting (on Transmit data line).

When this signal is "On" and the request to send,data set ready and data terminal ready are all "On", the DTE is assured that its data will be sent to the communications link. When "Off", it is an indication to the DTE that the DCE is not ready, and therefor data should not be sent.

When the data set ready and data terminal ready signals are not implemented, in a local connection which does not involve the telephone network, the Clear To Send and request to send signals are sufficient to control data transmission.

- 6 Data Set Ready

  .Name:CC
  .Direction:DTE<-DCE
  .CCITT:107

  On this line the DCE tells the DTE that the communication channel is available (i.e., in an automatic calling system, the DCE (modem) is not in the dial,test or talk modes and therefor is available for transmission and reception). It reflects the status of the local data set ,and does not indicate that an actual link has been established with any remote data equipment.

- 7 Signal Ground

  .Name :AB
  .Direction:-
  .CCITT:102

  This pin is the reference ground for all the other signals, data and control.

- 8 Receive Line Signal Detect or Data Carrier Detect

  .Name :CF
  .Direction:DTE<-DCE
  .CCITT:109

  The DCE uses this line to signal the DTE that a good signal is being received (a "good signal" means a good analog carrier, that can ensure demodulation of received data).

- 9 +P

  This pin is held at +12 volts DC for test purposes.

- 10 -P

  This pin is held at -12 volts DC for test purposes.

- 12 Secondary Receive Line Signal Detect

  .Name:SCF
  .Direction:DTE<-DCE
  .CCITT:122

  This signal is active when the secondary communication channel is receiving a good analog carrier (same function as the receive line signal detect).

- 13 Secondary Clear To Send

  .Name :SCB
  .Direction:DTE<-DCE
  .CCITT:121

  An answer signal to the DTE. When this signal is active, it tells the DTE that it can now start transmitting on the secondary channel (on the secondary transmitted data line).

- 14 Secondary Transmitted Data

  .Name :SBA

.Direction:DTE->DCE
.CCITT:118

Serial data (secondary channel) is sent on this line from the DTE to the DCE. This signal is equivalent to the transmitted data line except that it is used to transmit data on the secondary channel.

- 15 Transmission Signal Element Timing

  .Name:DB
  Direction:DTE<-DCE
  .CCITT:114

  The DCE sends the DTE a clock signal on this line. This enables the DTE to clock its output circuitry which transmits serial data on the transmitted data line.

  The clock signal frequency is the same as the bit rate of the transmitted data line. A "On" to "Off" transition should mark the center of each signal element (bit) on the transmitted data line.

- 16 Secondary Receive Data

  .Name :SBB
  .Direction:DTE<-DCE
  .CCITT:119

  Serial data (secondary channel) is received on this line from the DCE to the DTE. When the secondary channel is being used only for diagnostic purposes or to interrupt the flow of data in the primary channel, this signal is normally not provided.

- 17 Receiver Signal Element Timing

.Name:DD
.Direction:DTE<-DCE
.CCITT:115

The DCE sends the DTE a clock signal on this line. This clocks the reception circuitry of the DTE which receives serial data on the received data line.

The clock signal frequency is the same as the bit rate of the received data line (BB). The "On" to "Off" transition should indicate the center of each signal element (bit) on the received data line.

- 19 Secondary Request To Send

Name:SCA
Direction:DTE->DCE
CCITT:120

The DTE uses this signal to request transmission from the DCE on the secondary channel. It is equivalent to the request to send signal.

When the secondary channel is only used for diagnostic purposes or to interrupt the flow of data in the primary channel, this signal should turn "On" the secondary channel un-modulated carrier.

- 20 Data Terminal Ready

.Name:CD
.Direction:DTE->DCE
.CCITT:108.2

When on, tells the DCE that the DTE is available for receiving. This signal must be "On" before the DCE can turn data set ready "On", thereby indicating that it is connected to the communications link.

The Data Terminal Ready and data set ready signals deal with the readiness of the equipment, as opposed to the clear to send and request to send signals that deal with the readiness of the communication channel.

When "Off", it causes the DCE to finish any transmission in progress and to be removed from the communication channel.

- 21 Signal Quality Detector

  Name:CG
  Direction:DTE<-DCE
  CCITT:110

  This line is used by the DCE to indicate whether or not there is a high probability of an error in the received data. When there is a high probability of an error, it is set to "Off", and is "On" at all other times.

- 22 Ring Indicator

  .Name:CE
  .Direction:DTE<-DCE
  .CCITT:125

  On this line the DCE signals the DTE that there is an incoming call. This signal is maintained "Off" at all times except when the DCE receives a ringing signal.

- 23 Data Signal Rate Selector

  Name:CH/CI
  Direction:DTE->DCE
  CCITT:111/112

The DTE uses this line to select the transmission bit rate of the DCE. The selection is between two rates in the case of a dual rate synchronous connection, or between two ranges of data rates in the case of an asynchronous connection.

Typically, when this signal is "On", it tells the DCE (modem) that the receive speed is greater than 600 baud.

- 24 Transmitter Signal Element Timing

  Name:DA
  Direction:DTE->DCE
  CCITT:113

  The DTE sends the DCE a transmit clock on this line. This is only when the master clock is in the DTE.

  A "On" to "Off" transition should indicate the center of each signal element (bit) on the transmitted data line.

- A note on signal travel direction

  The pin names are the same for the DCE and DTE. The Transmit Data (pin number 2) is a transmit line on the DTE and a receive line on the DCE, Data Set Ready (pin number 6) is a receive line on the DTE and a transmit line on the DCE, and so forth.

6.2 Electrical Signal Characteristics

- Voltage levels defined in the standard
- **Data signals "0","Space" "1","Mark"**
-

- **Driver (Required) 5 - 15 -5 - -15 Volts**
- **Terminator (expected) 3 - 25 -3 - -25 Volts**
-
- **Control signals "Off" "On"**
-
- **Driver (Required) -5 - -15 5 - 15 Volts**
- **Terminator (expected) -3 - -25 3 - 25 Volts**

- **6.3 The Noise Margin Issue**

  Note that terminator (receiving end) voltages are not the same as driver required voltages. This voltage level definition compensates for voltage losses across the cable.

  Signals traveling along the cable are attenuated and distorted as they pass. Attenuation increases as the length of the cable increases. This effect is largely due to the electrical capacitance of the cable.

  The maximum load capacitance is specified as 2500pf (picofarad) by the standard. The capacitance of one meter of cable is typically around 130pf, thus the maximum cable length is limited to around 17 meters. However, This is a nominal length defined by the standard, and it is possible to use longer cables up to 30 meters, with low-capacitance cables, or with slow data rates and a proper error correction mechanism.

- **6.4 Interface Mechanical Characteristics**

  The connection of the DCE and the DTE is done with a pluggable connector. The female connector should be associated with the DCE. The following table lists the pin assignments defined by the standard. The type of connector to be used is not mentioned in the standard, but

the DB-25 (or on IBM-AT's, a minimal DB-9) connectors are almost always used.

- ## 6.5 Pin designation for the 25-pin and 9-pin DB connector

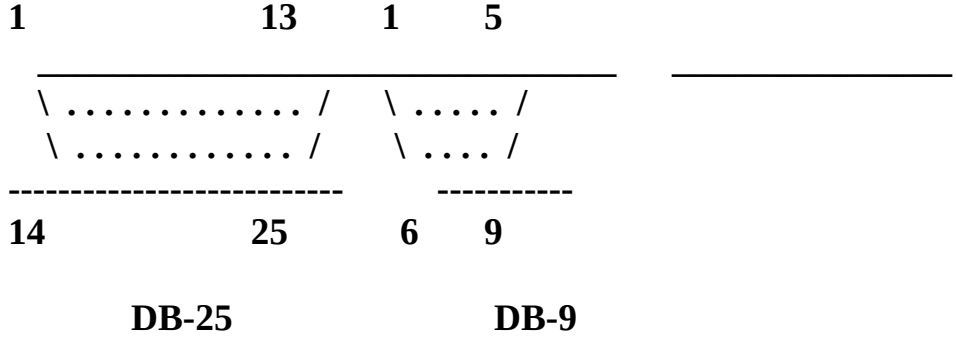
includes equivalent CCITT V.24 identification, and signal direction

DB-25 Pin #	DB-9 Pin #	Common Name	EIA Name	CCITT	DTE-DCE	Formal Name
1		FG	AA	101	-	Frame Ground
2	3	TD	BA	103	--->	Transmitted Data, TxD
3	2	RD	BB	104	<---	Received Data, RxD
4	7	RTS	CA	105	--->	Request To Send
5	8	CTS	CB	106	<---	Clear To Send
6	6	DSR	CC	107	<---	Data Set Ready
7	5	SG	AB	102	----	Signal Ground, GND
8	1	DCD	CF	109	<---	Data Carrier Detect
9		--	--	-	-	+P
10		--	--	-	-	-P
11		--	--	-	-	unassigned
12		SDCD	SCF	122	<---	Secondary Data Carrier Detect
13		SCTS	SCB	121	<---	Secondary Clear To Send
14		STD	SBA	118	--->	Secondary Transmitted Data
15		TC	DB	114	<---	Transmission Signal Element Timing
16		SRD	SBB	119	<---	Secondary Received Data

17 RC DD 115 ---> Receiver Signal Element Timing
18 -- -- - - unassigned
19 SRTS SCA 120 ---> Secondary Request To Send
20 4 DTR CD 108.2 ---> Data Terminal Ready
21 SQ CG 110 <--- Signal Quality Detector
22 9 RI CE 125 <--- Ring Indicator
23 -- CH/CI 111/112 ---> Data Signal Rate Selector
24 -- DA 113 <--- Transmitter Signal Element Timing
25 -- -- - - unassigned

- Diagram of the DB-25 and DB-9 connectors

male connectors , front view



Chapter-7

7.DTMF Specifications

A total computer system includes both hardware and software.Hardware consists of the physical components and all associated equipment.Software refers to the programs that are written for the specific microengine.Although the possibility of not being aware of the hardware and writing the software or vice versa exits, but when the process of validation of the hardware and of generating the machine language is concerned,knowledge of both the aspects is overiding .
Dual tone Multiple Frequency signaling is used in telephone dialing, digital answering machine, computers and even in interactive banking system.DTMF signaling represents each symbol on a telephone touch tone keypad (0-9,A,B,C,D,*,#) using two sinusoidal tones. When a key is pressed, a DTMF signal consisting of a row frequency tone plus a column frequency is generated and transmitted.

The process of decoding this signal is performed at the exchange. The Micro processors installed at the exchange performs number of operation to detect this signal.The usually employed micro engines work on extensive floating point operation and thereby require heavy computing power. The new design would work on an algorithm which operates on fixed point operation written by Amey A .Deosthali, shawn R. Mccaslin and Brial

L.Evans, which was presented to IEEE in October 1999 for an optimized LOW-COMPLEXITY ITU COMPLIANT DUAL TONE MULTIPLE FREQUENCY DETECTOR.

7.1 What is DTMF

The word DTMF is the acronym for “Dual Tone Multiple Frequency” DTMF tones are the sounds emitted when one dials a number on a touch-tone phone .

A DTMF codec incorporates an encoder that translates key strokes or digit information into dual-tone signals ,as well as a decoder that detects the presence and the information content of incoming DTMF tone signals . each key on the keypad is identified uniquely by its row frequency and its column frequency.

7.2 AT&T Specfifications for Tone Generation:

Tone duration specifications by AT&T state the following:10 digits/sec is the maximumdata rate for touch-tone signals.For a 100-msec time slot,the duration for the actual tone is at least 45 msec and not longer than 55 msec.The tone generator must be quiet during the remainder of the 100-msec time slot.

7.2.1 DTMF Tone Generation

Though the domain of this project is limited to automated inquiry system,but the syndicate has to thoroughly study the mechanism of DTMF tone

generation. These techniques,in theory,directly affect the algorithm used to detect DTMF tones.

Modems have traditionally been the device used to generate these tones from a computer.But the more sophisticated modems on the market today are nothing more than a DSP (digital signal processor) with accompanying built –in software to generate and interpret analog sounds into digital data. The computers sitting on desk have more CPU power,a more complex OS, and very often a just as sophisticated DSP.

7.2.1.1 Methods of Tone Generation

There are three methods for generating DTMF tones by summing two sine waves. These are:

Table look up
Taylor Series
Harmonic Resonator

7.2.1.2 Table Look –up Method

The table look-up method retrieves previously computed sine wave values from memory.The sine function is periodic and only one period must be computed. Since this is sampled data,an accuratre sine wave generator must confirm that the samples starting and ending point are the same.The easiest way to determine this is to find the smallest value of I(an integer) that when multiplied by the ratio below will result in an integer.

$$(Fs/Fo) * I = \text{integer\# of samples}$$

Fs = sampling frequency

Fo = frequency of tone to be generated

The period of the frequency to be generated must be evenly divisible by a multiple of the sampling rate. This method can require large amounts of memory if the frequency is not an easy divisor of the sampling rate . If there are numerous frequencies to generate ,or the frequency is unknown beforehand, then the table look-up method may not be the best solution.

7.2.1.3 Taylor Series Expansion

The Taylor series expansion method reduces the memory required to compute an approximation of the sine value.The accuracy can be selected.The Taylor series expansion method expresses a function by polynomial approximation. The expansion for a sine function order 5 is :
$\text{Sin}(x) = 3.140625*x + 0.02026367 * x^2 - 5.325196* x3 + 0.544678\ x4 + 1.800293\ x5$
Where $0<x<p/2$

Note that x is in radians and that the other three quadrants must be accounted for by manipulating the sign and the input value,x.The Taylor series expansion method requires more computations but less memory than the table.

7.2.1.4 Harmonic Resonator

This method is based upon two programmable,second-order digital sinusoidal oscillators,one for the row tone and one for the column tone .

Two oscillators, instead of eight, facilitate the code and reduce the code size .Of course, for each digit that is to be encoded, each of the two oscillators needs to be loaded with the appropriate coefficient and initial conditions before oscillation can be initiated .

Since typical DTMF frequency ranges from approximately 700 Hz to 1700 Hz, a sampling rate of 8 kHz for this implementation is within a safe area of the Nyquist criteria. The following figure displays the block diagram of the digital oscillator pair.Note that 1/z corresponds to a delay of one sampling period.

$$P1,2 = e+jw$$

The discrete time impulse response

$$H(n) = Asin(n+1)wo)*u(n)$$

Corresponding to the above second-order system clearly indicates a clean sinusoidal output due to a given impulse input .Therefore, this system can be termed as digital sinusoidal oscillator or digital sinusoidal generator. For the actual implementation of a digital sinusoidal oscillator, the corresponding difference equation is the essential system descriptor, given by

$$Y(n)= -a1*y(n-1)-a2*y(n-2) +bo*\&(n)$$

Where initial conditions y(-1) and y(-2) are zero .Note that the impulse applied at the system input serves the purpose of beginning the sinusoidal oscillation. Thereafter, the oscillation is self-sustaining, as the system has no damping and is exactly marginally stable. Instead of applying a delta impulse at the input, let the initial condition Y(-2) be the systems oscillation initiator and remove the input. With this in mind , the final difference equation is given by:

$$Y(n) = 2*\cos (wo) *y(n-1) –y(n-2)$$

Where

$$Y(-1) =0$$
$$Y(-2)=A\sin (wo)$$
$$Wo =2* \ * fo/fs$$

With fs being the sampling frequency, fo being the frequency of tone and A being the amplitude of the sinusoid to be generated. Note that the initial condition y(-2) solely determines the actual amplitude of the sine wave.

7.3 DTMF TONE DETECTION

DTMF tone detection was one of the major tasks involved in this project. Different functions are performed by the system with different keystrokes of the user. Theory behind this whole process is explained in the following.

The task to detect DTMF tones in an incoming signal and to convert them into actual digits is certainly more complex than the encoding process. The decoding process is by its nature a continuous process, meaning it needs to continually search an incoming data stream for the presence of DTMF tones.

7.3.1 Collecting Spectral Information Using Goertzel's Algorithm

The Goertzel algorithm is the basis of the DTMF detector. This method is a very effective and fast way to extract spectral information from an input signal. This algorithm essentially utilizes two –Pole type filters to compute DTMF values effectively .It is , thereby, a recursive structure (always operating on one incoming sample at time), as compared to the DFT (or FFT) that needs a block of data before being able to start processing.

Another major advantage of Goertzel's algorithm is that it gives only the magnitude of the frequency in demand.

For the actual tone detection, the magnitude (here, squared magnitude) information of the DFT is sufficient. After a certain number of samples N (equivalent to a DFT block size). The Goertzel filter output converges towards a pseudo DFT value k(n) , which can then be used to determine the squared magnitude.
The Goertzel algorithm is much faster than a true FFT, as only few of the set of spectral line values are needed and only for those values are filters provided.

Squared magnitudes are needed for eight row/column frequencies and for their eight-second harmonics. The second harmonics information later enables discrimination of DTMF tones from speech or music.

The choice of N is mainly driven by the frequency resolution needed, which sets a lower boundary. N also is chosen so that (k/N) fs most accurately coincides with the actual DTMF frequencies (see Table 1) assuming ks are integer values and fs is a sampling frequency of 8 kbps.

As the first stage in the tone-detection process, the goertzel algorithm is one of the standard schemes used to extract the necessary spectral information from an input signal.Essentially, the goertzel algorithm is a very fast way to compute DFT values under certain conditions. It takes advantage of two facts:

The periodicity of phase factors kn allows the expression of the computation of the DFT as a linear filter operation utilizing recursive difference equations.

Only a few of the spectral values of an actual DFT are needed (in this application, there are eight row/column tones plus an additional eight tones or corresponding 2^{nd} harmonics).

Keeping in mind that a DFT o size N is defined as

$$X(k) = \quad x(m)e\text{-}jkm2\ /N$$

It is possible to find the sequence of a one- pole resonator

$$Yk(n) = \quad x(m)e\ \ j2\ \ k(n\text{-}m)/N$$

Which has a sample value at n = N coinciding exactly with the actual DFT value.In other words, each DFT value X(k) can be expressed in terms of the sample value at n = N resulting from a linear filter process (one-pole filter).

It can be verified that

$$X(k) = Yk(n) = \quad X(m)e\text{-}j2\ km/N$$

The difference equation corresponding to the above one-pole resonator which is essential for the actual implementation, is given by

$$Yk(n) = e \qquad yk(n\text{-}1) + x(n)$$

With y(-1) =0 and pole location

Being a one-pole filter. This recursive filter description yet remains complex two pole filter with complex conjugate poles and only real multiplications in its difference equation,

$$V\ k\ (n) = 2\cos\ (2\ \ k\ /N)^*\ vk\ (n-2) + x(n)$$

Where vk (-1) and vk (-2) are zero.

In the Nth iteration, only a complex multiplication is needed to compute the DFT value, which is

X k = yk (n) = vk (n) –e vk (N-1)

However, the DTMF tone-detection process does not need the phase information of the DFT squared magnitudes of the computed DFT values, in general , suffices

$$X\ (k)\ 2\ \ = yk\ (N)\ yk^*\ (N)$$

After some arithmetical manipulation, it is found that

$$X(k)\ 2 = vk2\ (N) + vk2\ (N-1) -2\ \cos\ (2\ \ k/N)vk(N)\ vk(N-1)$$

Which gives the energy of the tone .

7.3.2 Validity Checks

Once the spectral information (in the form of squared magnitude at each of the row and column frequencies and their second harmonics) is collected,a series of tests need to be executed to determine the validity of tone and digit results.

7.3.2.1 Signal Strength Check

A first check makes sure the signal strength of the possible DTMF tone pair is sufficient. The sum of the squared magnitudes of the peak spectral row component and the peak spectral column component needs to be above a

certain threshold . Since already small twists (row and column tone strength are not equal) result in significant row and column peak differences, the sum of row and column peak provides a better parameter for signal strength than separate row and column checks.

7.3.2.2 Twist Check

Tone twists (the ratio of column to row or row to column signal strength) are investigated in a separate check to make sure the twist ratio specifications are met. The spectral information can reflect the types of twists.The more likely one, called “reverse twist” assumes the row peak to be larger than the column peak .Row frequencies (lower frequency band) , assuming a low-pass filter type telephone line . The decoder, therefore, computes a reverse twist ratio and sets a threshold of 8 dB acceptable reverse twist.

The other twist, called “standard twist” occurs when the row peak is smaller than the column peak. Similarly, a “standard twist ratio” is computed and its thresholod is set to 4 dB acceptable standard twist.

7.3.2.3 Relative Peak Check

The program makes a comparison of spectral components within the row group as well as within the column group . The strongest component must stand out (in terms of squared amplitude) from its proximity tones within its group by more than a certain threshold ratio.

7.3.2.4 Second Harmonic Strength Check

Finally , the program checks on the strength of the second harmonics in order to be able to discriminate DTMF tones from possible speech or music . It is assumed that the DTMF generates tones only on the fundamental frequency; however, speech will always have significant even-order harmonics added to its fundamental frequency component. This second harmonics check, therefore, makes sure that the ratio of the second harmonics component and the fundamental frequency component is below a certain threshold. If the DTMF signal pair passes all these checks, we say a valid DTMF tone pair, which corresponds to a digit, is present.

7.3.2.5 Check For Validity of Tone

.We now need to determine if the valid DTMF tone information contains stable digit information. This is done by mapping the tone-pair to its corresponding digit and comparing it with the previously detected digit. We call the digit information stable if it has been detected twice successively.

7.3.2.6 Check Whether New Digit Pressed

Finally, we compare the detected digit with the previous-to-last digit. Only if the last digit was preceded by a pause do we accept the current digit as a valid digit. The detector is then forced into a state where it waits for a pause before being able to accept a new digit . This last step is necessary to ensure the discrimination of identical strokes succeeding one another.

7.4 Modification in Goertzel Algorithm

Since we only require the magnitude information associated with Goertzel we modify it further to output only the energies .
This chapter outlines the implementation of E-Mail via phone system including all three packages related to Telephony (TFX0, Voice Messaging (VoiceM) and Integration. This implementation is based on the design explained in the last chapter. The packages explained in this chapter include the classes for POP3 client, Text-to-Speech conversion,message recording and playback.

Chapter-8

8. AT COMMANDS

The modem may be configured in response to AT voice commands to provide enhanced Adaptive Differential Pulse CodeModulation ((ADPCM) coding and decoding for the compression and decompression os digitized voice. ADPCM compression supports the eddicient storage of voice messages, while koptional coder silence deletion and decode r silence interpolation significantly increase compression rates. The ADPCM voice Mode Supports three sub modes once a voice connection is established (see#CLS command):On line VoiceCommand Mode, Voice Receive Mode, and Voice transmit Mode .

8.1 VOICE SUBMODES

8.1.1 ONLINE VOICE COMMAND MODE

Online VoiceCommand Mode is the default Voice sub mode entered when the #CLS=8 command is issued, and may also be entered from voice

receive mode or voice transmit mode. Entry into online Voice Command Modeis indicated to the DTE via the VCON message , after which AT commands can be entered without aborting the telephone line connection. If the modem is the answerer, it en enters online Voice Command Mode immediately after going off-hook, and can report instances of DTMF tonesand calling tones to the DTE. If the modem is the originator, it enters Online voice Command Mode based on detection of the ring back cadence going away, upon expiration at the ring bach never came timer, or upon detection of answer tone, and the ,odem can report DTMF tones, answer tones, busy tone, and dial tone to the DTE.
(Nofte that DTMF tone reporting is supported in this mode if DTMF reporting is enabled
via the #VTD command.)

When this mode is entered as a result of going off-hook with the D or A vommand, VCON is always sent to the DTE, after which the modem accepts commands. If this mode is entered from voice Transmit Mode, the DTE has issued the <DLE><ETX>, and the modem responds with VCON. If this mode is entered from the voice revceive Mode because of a key abortr, the modem issues the <DLE><ETXfollowed by VCON.
If the #VLS command has switched in a handset or other device in place of the telephone line, online voice CommandMode is immediately entered,whereas if the telephone line is selected, a physical connection with another station must occur before entering this mode.

8.1.2 VOICE RECEIVE MODE

Voice receiveMode is jintered when the DTE issues the #VRX command because it wants to receive voice data. This typically occurs when either recording a greeting messae, or when recording voice messages from a remote station.
In Receive Mode, voice samples from the modem analog-to-digital converter(ADC) are sent to the ADPCM codec for compression, and can then be read by the host. AT commands control the codec bits-bits-per-samplerate and select (optional) silence deletion including adjustment at the silence detecionperiod.
In this mode, the modem detects and reports DTMF , dial tone, busy tone cadence, and inactivity (periods of silence) as enabled by the # VTD and

#VSS commands, respectively. The modem can exit the Voice Receive Mode only via a DTE Key Abort, or via Dead man timer Expiration(S30).

8.1.3 VOICE TRANSMIT MODE

Voice Transmit Mode is entered whin the DTE issues the #VTX command because it wants to transmit voice data. In this mode , the modem continues to detect and report DTMF and calling tones if enabled by the #VTD command. This mode is typically used whin playing bach greeting messages or previously received/recorded messages. In this mode , voice decompression is provided by the codec, and decompressed data is reconsitituted into analog voice by the DAC at the original voice compression quantization sample-per-bits rate. Optional silence interpolation is enabled if silence deletion was selected for voice compression.

8.2 VOICE CAPABILITIES

8.2.1 CALL STABLISHMENT- ANSWER

For most call originations, it is known ahead of time what type of call is being attempted, and it is acceptable to disconnect if the remote side of the connection does not cooperate . In this case, the modem can be configured ahead of time with the existing +FCLASS (+FAA) or the #CLS command to be a data , fax or voice modem. For data and Fax Modes, the modem subsequently either succeeds with the desired type of connection, or eventually hangs up. For the Voice Mode, the DTE has the option of hanging up if there are indications that the remote station has not answered in voice , thus implementing a directed originate for voice. The following are the three connection type choices

1.VOICE

The modem dials and reports call progress to the DTE, which reduces to reporting NO DIAL TONE, or BUSY. The modem allows the DTE to program a time period, which if elapsed after any ring back is detected, forces the modem to assume the remote has gone off-hook. A secondary time period (safety valve) can define a maximum elapsed time after dialing

for receiving no ring back before the modem assumes that the remote has gone off-hook. This safety valve is devised in case the remote picks up the telephone before any ring bach is generated, and no other tones are detected .In this mode, the modem isattempting to make a voice connection only and therefore, while waiting for ring back to disappear, it is also feasible to disconnect upon detection something which is definitely not voice from the remote, such as any answer tone. The modem provides detection of "ringback" went away or never came

2.FAX CAPABILITIES

The modem dials and reports call progress to the DTEas in all modes . Afax Class 1 or class 2 handshake is pursued according to the current configuration.

3.DATA

The modem ddials and reports call progress to the DTE as in all modes. A data handshake is pursued according to the current configuration. Adaptive Originate (Dial with Voice/Data/fax Discrimination). The DTE may wish to originate a call, which adapts to the remote answer, For instance, the user may wish to send a voice message jif a human pichs up the telephone , but a facsimile if a fax machine answers. The modem can facilitate this type of adaptive originate by extending what it does for the picked up the line, the modem goes back to Online VoiceCommandMode, thus terminating the " connecting state" Once in this mode , the modem reports what it receives from the answerer via specific result codes to the DTE The DTE can then have the option of pursuing a data , fax, or voice connection.

8.2.2 CALL ESTABLISHMENT – ANSWER

If the DTE wants to be only one kind of answerer (i.e,voice, fax, or data) it can configure the modem to answer exclusively in the chosen mode.

1.VOICE

The modem is configured to answer in Voice Mode only and assumes the caller will cooperate. After going off-hook , the voice VCON is issued , no

answer tone is generated , and the modem is immediately placed in Online Voice Command Mode. The DTE typically responds by sending a greeting message of some type , and DTMF tone recognition/reporting can be enabled. Eventually, an Incoming voice message can be recorded by the host. (Unpredictble results occur if the caller is not prepared for a voice call.)

2.FAX CAPABILITIES

The modem is cofigured to answer in Class 1 or Class 2 Fax Mode only, and it assumes the caller is going to cooperate. This configuration has the effect of disabling Voice Mode, forcing +FCLASS to either 1 or 2, and forcing both+ FAA and +FAE to 0.

3.DATA

The modem is configured to answer in Data Mode only and assumes the caller is going to cooperate. This configuration has the effect of disabling Voice Mode, forcing +FCLASS=0, and forcing both +FAA and +FAE to 0.

8.2.3 **ADAPTIVE ANSWER (ANSWER WITH VOICE/DATA/FAX DISCRIMINATION)**

In normal operation, it is desirable for a modem supporting fax and voice to provide the ability to discriminate between the two when answering unsolicited or unattended calls. (It is most often the case that a fax is received or a Voice message recorded when nobody is present.)

1. DATA/FAX DISCRIMINATION

If the DTE wishes to allow for a data or fax call, the + FCLASS and +FAA or+FAE commands can be configured for adaptive answer between data and Class 1 or Class 2 fax

2. VOICE/FAX DISCRIMINATION

This is the most important ddiscrimination capability needed from the users standlpoint. The modem must be configured for Voice (#CLS=8),

causing the modem to enter Online Voice Command Mode immediately upon going off-hook. In Voice Mode, the DTE automatically receives indications of DTMF tones and Calling Tones. The DTE can now switch to Voice Transmit Mode in order to play a greeting message, perhaps one which instructs the caller how to enter specific DTMF sequences to switch modes. The dte can then react to the response, or the lack thereof , to such a message . The modem supports switching to a Class 1 or Class 2 answer mode by virtue of the #CLS=1 or 2 commands, and if such a switch is made and fails, the modem reports the failure but does hang up, allowing the DTE further experimentationtime. If the user wishes to switch to Class 1 or 2, but also wants the DTE to indeed hang hup the line if the fax fails, the +FCLASS command shoul;d be used instead of the #CLS command. The only difference between these commands is that issuing +FCLASS cancels the modems memory of voice, where as #CLS causes the modem to remain off-hook, even if a fax or data handshake fails, until it receives an H command .

3. VOICE/DATA/FAX DISCRIMINATION

The DTE can try data modem operation after an answer by changing the #CLS setting to 0. Adata handshake attempt can be added based upon DTMF responses or lack thereof.

8.3 VOICE DATA TRANSFER

A significant area of concern when handling the transfer of voice data is the data transfer rate on the modem/DTE interface. Data transfer rates can be expressed as the number of interruptswhich must be serviced per time period to keep up. This is a function of the samlpling rate and compression method (if any) used by the modem, and the DTE interface speed required to handle the data flow on the telephone line side.

The modem can detect specific tones and other status information, and report these to the DTE while in any of the three voice sub modes. The modem simultaneously looks for 1300 and 1100 Hz calling tones when answering, and for CCITT and Bell qnswer tones when originating. The modem can also detect dial or busy tones in any of the three voice sub modes. All detected tones, as well as certain other statuses addressed such as silence and “ teleset off-hook” (i.e, handset off-hook) are reported as

shielded codes. When ih Online Voice command Mode or voice transmit Mode, the codes are sent to the DTEimmediately upon verification by the modem kof the associated tone , status, or cadence. In this mode, the 2-character code is not buffered, nor does the DTE have the ability to stop the code with l
Flow control. If the DTE has started (but not completed) sending any AT command, the Tone Monitoring function is disabled until the command has been received and processed. The modem can discriminate between single and multiple DTMF tones received . If calling todne, dial tone, busy tone, or answer tone is detected , this detection is reported repeatedly (at reasonable intervals) if the DTE takes no action, and the tone continues to be detected.

8.4 TABLE SHIELDED CODES SENT TO THE DTE

CODE SENT TO DTE	MEANING
<DLE>0-<DLE>9	DTMF. Digits 0 through 9, *, or A through D detected
<DLE>*,<DLE>#	by the hmodem, ie, user has pressed a key on a local or <DLE>A-<DLE>D remote telephone, Thde modem sends only one <DLE> code per DTMH button pushed.

<DLE>a	Answer Tone (CCITT) . Send to the DTE when the V.25/T.3 0 2100 Hz Answer Tone (data or Fax) is detected. If the DTE fails to react to the code, and the modem continuesto detect as often as once every half second.
<DLE>b	Busy. Sent in Voice Receive Mode when the busy cadence s detected , after any remaining data in the voice in receive buffer. The modem sendsthe busy <DLE>b code every 4

seconds if busy continues to be detected and the DTE does not react. This allows the DTE the flexibility of ignoring what could be a false busy detection.

8.5 VOICE PLAYBACK

To deeect playback of a message recorded via a handset or microphone, or of a message recorded during a voice call , the DTE must configure the modem for voice Mode (#CLS=8) and select the proper relay setup (#VLS) to instruct the modem whether to use the handset or speaker. The modem responds to the #VLS

command by issuing a relay activate command to select the input device. The hardware must provide a means of selecting a handset and/or microphone instead of the telephone line, as this input device. When a device other that the telephone line is selected, the modem immediately enters Online voice command Mode (indicated by VCON). DTMF detection is thus enabled as soon as the DTE selects the device, such as a handset, although the user still needs to physically pich up the telephone before he can issue DTMF tones . Once selected, however, the user can indeed pich up the telephone and " press buttons." Even if the DTE has not entered voice Receiver or transmit Modes (#VTX or #VRX), these DTMF tones are delivered via shielded codes, identically to when a physical telephone connection exists but the DTE has not yet commanded receive nor transmit.

When the DTE decides to play the message , it issues the #VTX command , and the modem immediately switched in, the modem immediately issues the CONNECT message indicating that the modem is in Voice Transmit Mode and is expecting Voice datafrom the DTE . A subsequent <DLE><ETX> has to be issued to switch back to Online voice Command Mode.

8.6 VOICE CALL TERMINATION

8.6.1 LOCAL DISCONNECT

The DTE can disconnect from a telephone call by commanding a mode change to Online Voice Command Mode (if not already in it), and by issuing the H command.

8.6.2 REMOTE DISCONNECT DETECTION

When in Voice Receive Mode, the modem sends the proper shielded <DLE> code when loop break, dial tone, or busy tone is detected. The nodem stays in Voice Receive Mode, however, until the DTE issues a key abort to force Online Voice Command Mode,. The DTE must issue the H command if it wisher to hang up.

8.7 MODE SWITCHING

8.7.1 VOICE TO FAX

If the modem is in Online Voice Command Mode (i.e. it has gone off-hook with #CLS=8 in effect). The DTE can attempt a fax handshake by setting #CLS+1 or #CLS=2 followed by the A or D command corresponding to fax receive or send. This has the dffect of beginning a fax Class 1 or Class 2 handshake (see#CLS command).

8.7.1.1 UNSUCCESSFUL FAX CONNECTION ATTEMPT TO VOICE

A Fax handshake which does not succeed, attempted as the result of the DTE modifying the #CLS setting from voice (8) to fax (1 or 2) does not result in the modem hanging up, allowing the DTE the flexibility of commanding a switch back to Voice Mode with #CLS=8.

8.7.2 VOICE TO DATA

If the modem is in the Online Voice command Mode the DTE can attempt a data.
Handshake by setting #CLS=0 followed by the A or D command. This has the effect of beginning a Data Mode handshake according to the current Data Mode S-register and command settings.

8.7.2.1 UNSUCCESSFUL DATA CONNECTION ATTEMPT TO VOICE

A data handshake which does not succeed attempted as the result of the DTE modifying the #CLS setting from voice (8) to data (0), does not result in the modem hanging up , allowing the DTE the flexibility of commanding a switch back to voice Mode with #CLS=8.

8.8 CALLER ID

The modem supports caller ID by passing the information received in Bell 202 FSK format to the DTE after the first RING detect . The modem supports both formatted and unformatted reporting of caller ID information received in ICLID (Incoming call Line ID) format as supported in certain areas of the U.S and Canada. The DTE enables this feature via th #CID command.

8.9 AT VOICE COMMAND SUMMARY

Table provides a complete summary of the AT voice command described in detail in following sections.

8.9.1 GLOBAL AT COMMAND SET EXTENSIONS

The AT commands in the following section are global mea ningthat they can be issued in any appropriate mode (i.e. any #CLS setting). For consistency , the command set is divided jinto action commands and parameters (non-action commands). Those commands which are action commands i.e , those which cause some change in the current operating

hehavior of the modem) are identified as such, and the remaining commands are parameters.

8.9.2 ATA – ANSWERING IN VOICE

The answer action command works analogously to the way it works in Data and Fax Modes except for the following:

1. When configured fo Voice Mode (#CLS=8), the modem enters Online Voice Command Mode immediately after going off-hook. When the #CLS=8 command is issued, the modem can be programmed to look for 1100 and 1300 Hz callimg tones (see #VTD). Thus eliminating the need to do so as part of A command processing. After the VCON message is issued the modem re-enters Online voice Command Mode while sending any incoming DTMF or calling tone indications to the DTE.
2. After answering in Voice Mode (#CLS=8) the DTE, as part of its call discriminmation processing can decide to change the #CLS setting to ttempt receiving a fax in Class 1 or to make a data connection. In such a case the DTE commands the modem to proceed with the data or fax handshake via the a command even though the modem is already off- hook.

8.9.3 VCON

Issued in Voice Mode (#CLS=8) immediately after going off-hook

Command Function

A Answering in Voice Mode
D Dial command in Voice Mode
H Hang up in Voice Mode.

Z Reset from Voice Mode
#BDR Select baud rate (turn off auto baud)
#CID Enable callerID detection and select reporting format
#CLS Select data fax or voice
#MDL Identify model.

#MFR Identify manufacturer.
#REV .Identify revision level
#VBQ Query buffer size.
#VBS Bits per sample (ADPCM).
#VBT Beep tone timer.
#VCI? Identify compression method (ADPCM0).
#VLS Voice line select (ADPCM).
#VRA Ringback never came timer (originate)
#VRN Ringbacknever came timer (originate).
#VRX Voice Receive Mode (ADPCM).
#VSD Silence deletion tuner (voice receive ADPCM).
#VSK Buffer skid setting.
#VSP Silence detection period (voice receive ADPCM).
#VSR Sampling rate selection (ADPCM).
#VSS Silence deletion tuner (voice revceive)
#VTD DTMF/tone reporting capability.
#VTX Voice Transmit Mode (ADPCM).

8.9.4 ATD

Dial command In Voice

The dial action command works analogously to the way it works in Data or Fax modes. When In Mode (#CLS=8):

1. The modem attempts to determined when the remote has picked upto the telephone line and once this determination has been made, the VCON message is sent to the DTE . Thius determination is initially made based upon ringback detection and disappearance . (See #VRA and #VRN commands.)

2. Once connected in Voice Mode the modem immediately enters the command state and switches to Online Voice Command Mode which enables unsolicited reporting of DTMH and answer tones to the DTE.
3. Parameters: Same as Data and Fax modes.

8.9.5 VCON

Issued in Voice Mode (#CLS=8) when the modem determines that th remote modem or handset has gone off-hook, or when returning to the Online voice Command Mode. (See #VRA and #VRN.)

8.9.6 NO ANSWER

Issued in Voice Mode (#CLS=8) when the modm determines that the remote has not picked up the line before the S7 timer expires.

8.9.7 ATH

Hang Up In Voice

This command works the same as in Data and Fax modes by hanging up (Disconnecting) the telephone line. There are, however, some specific considerations when in Voice Mode:

1. The H command forces #CLS=0 but does not destroy any of the voice parameter settings such as #VBS, #VSP, etc. Therefore if the DTE wishes to issue an H command and then pursue another voice call it must issue a subsequent#CLS=8 command, but it needn't reestablish the voice parameter settings again unless a change in the settings is desired.

2. The #BDR setting is forced back to 0, re-enabling auto baud.

3. If the #VLS setting is set to select a device which is not, or does not include the telephone line (such as a local handset or microphone), the H command deselects this device and reselects the normal default setting (#VLS=0). Normally, the DTE should not issue the H command while connected to a local device each as a handset, because nerely selecting this device results in VCON.The normal sequence of terminating a session withsuch a device is to use the #VLS command to select the telephone line, which by definition makes sure it is on-hook.

8.9.8 ATZ

Reset from Voice Mode

This command works the same s jin Data and faxmodes . In addition, the Z command resets all voice related parameters to default states , forces

the #BDR=0 condition (autobaud enabled), and forces the telephone line to be selected with the handset on-hook. No voice parameters are stored in NVRAM so the profile loaded does not affect the voice aspects of this command.

8.9.9 #BDR

Select Baud Rate (Turn offautobaud)

This command forces the modem to select a specific DTE/modem baud rate without further speed sensing on the interface. When a valid #BDR =n command is entered, the OK result code is sent at the current assumed sped. After the OK has been sent, the modem switches to the speed indicated by the #BDR=n command it has hjust received .

When In Online Voice Command Mode and the #BDR setting is nonzero (no autobaud selected), the modem supports a full duplex DTE interface. This means that the DTE can enter commands at any time, even if the modem is on the process of sending a shielded code indicating DTMF detection to the DTE . When in Online Voice Command Mode and the #BDR setting is zero (autobaud selected), shielded code reporting to the DTE is ddisabled . [Note that when #BDR has been set nonzero, the modem emplpoys the S30 Deadman Timer, and this timer starts at the point where #BDR is set the DTE interface, the modem reverts to #BDR =0 and #CLS=0.

8.9.10 #BDR?

Returns the current setting jof the #BDR command as an ASCII decimal value in result code format.

8.10 AT#V COMMANDS ENABLED ONLY IN VOICE MODE (#CLS=8)

The commands described in the following subsection are extensions to the command set which the modem recognizes only when configured for Voice Mode with the #CLS=8 command.

#VBQ? Bite Per Sample (Compression Factor)

#VBS? Returns the current setting of the #VBS command as an ASCII decimal
value in result code format.
#VBS=? Returns “2,3,4” which are the ADPCM compression bits/sample rates available. These bits/sample rates are correlated with the #VCI? Query command response which provides the single compression method available.
#VBS=2 Selected 2 bits per sample
#VBS=3 Selected 3 bits per sample
#VBS=4 Selected 4 bits per sample.
#VBT Beep Tone Timer

8.11 DEVICE TYPES SUPPORTED BY #VLS

8.11.1 ASCII DIGIT DEVICE TYPE AND CONSIDERATIONS

0 Telephone Line with Telephone handset. This is the default device selected . In this configuration, the user can pick up a handset which is connected to the same telephone line as the modem, and * record both sides of a vonversation with a remote station.
The modem currently supports one telephone line/handset, which is in the first position of the #VLS-? Response . (Note that the modem can interface to multiple telephone lines by having “0”’s in multiple positions in the #VLS? Response.) I telephone line is selected , the modem must be on-hook or it hangs up . The OK message is generated.

1 Transmit/Receive Device(other than telephone line). This is a handset, headset, or speaker-phone powered directly by the modem. When such a device is selected ,the modem immediately enters Online voice Command Mode, DTMF monitoring is enableif applicable, and the VCON response is sent . The modem

supports one such device as the second device listed in the #VLS=? Response.

2 Transmit Only Device. Normally, this is a microphone . When such a device is selected, the modem jimmediately enters Online voice Command Mode, DTMF monitoring is enabled if applicable, and the VCON response is sent. The modem supports selection of the internal speaker as the third device listed in #VLS=? Response.

3 Receive Only Device . Normally , this is a microphone. When such a device Is selected , the modem immediately enters Online voice Command Mode . DTMF monitoring is enabled if applicable, and the VCON response is sent . The modem supports one microphone as the fourth element returned in the #VLS=? Response.

4 Telephone line with Speaker On and handset. This device type can be used to allow the DTE to select the telephoneLine/headset(if picked up) with the modem speaker also turned ON. This can be used by the DTE to allow the user to monitor an incoming message as it is recorded.

8.12 S-REGISTES

The following S- register is global, meaning that it can be set in any appropriate mode(i.e, any#CLS setting). S30 – Deadman (inactivity) timer
Range: n = 0 – 225
Default: 0(OFF, which mians DTE should usually set jit to some value for Voice)
Command options:
S30=0 Dead man timer off. No matter how long it might continue , the modem
Never spontaneously hangs upthe telephone line or switches to audio baud
Hmode as a result of inactivity.

S30=1 to 255 This is period of time (in seconds) , which if expired caused the modem to hang up the telephone line if it is off-hook and hno data has passed during the period . The timer is also active whenever the #BDR setting is non-zero. In order to avoid a state where speed sence is disabled (even though the PC can crash, come back up , and try to issue commands at what should be a supported speed), the inactivity time-out occurs if there is no data passed on the DTE interface within the S30 period, even if the modem is on-hook . DTE software must not select a nonzero setting for #BDR until it is ready to establish a telephone call or virtual connection to a speaker or microphone . When there is an inactivity time out with #CLS+8, the modem always forces #CLS+0 and #BDR=0.

8.13 RESULT CODES FOR VOICE OPERATION

VCON is sent when the modem is configured for voice (#CLS=8), or when after answering or originating a call, the modem enters the Online voice command mode for the first time.

Typically, this is immediately after an off-hook in answer mode, and after ringback ceased in originate mode. VCON is also sent when the DTE requests a switch from Voice Transmit Mode to Online voice command Mode by issuing a <DLE><ETX> to the modem, or when the DTE requests a switch from voice receive mode to online voice command mode via the key abort . CONNECT CONNECT is sent when switching from the Online voice Command mode to either voice Receiver mode via the #VRX command , or to Voice transmit Mode via the #VTX command . This message is sent to the DTE to inform it that it may begin receiving or sending ADPCM data.

Chapter-9

9-OVERVIEW OF JCOMM

9.1 JAVAX.COMMEXTENSION PACKAGE

There are three levels classes in the java communications API:

. High –level classes like comm.portdentifier and comport manage access and ownership of communication ports.
. Low-level classes like SerialPort and ParallelPort provide an interface to physical connunications ports. The corrent release of the java communications API enables access to serial (RS-232) and parallel (IEEE 1284) ports.
. Driver –level classes provide an interface between the low-level classes and the underlying operating system . Driver-level classes are part of the implementation but not the java communications APL They should not be used by application programmers.

The java.comm pachage provides the following basic services:

. Enumerate the available ports on the system. Yhe static method commPortidentifier get portidentifiers returns an enumeration object that contains a CommPortimdentifier object for each available port.

This commporidentifier object is the central mechanism for controlling access to a communications port.

. Open and claim ownership of communications ports by using the high level methods in their CommPortIdentifier objects.
. Resolve port ownership contention between nultiple java applications. Events are propagated to notify interested applications of ownership contention and allow the port's owner to relinquish ownership. PortInUseException is thrown when an application fails to open the port.
. Perform asynchronous and synchronous I/O on connunications ports . Low level classes likeSerialPort and ParallelPort have methods for managing I/O on communications ports.
. Receive events describing communication port state changes . For example , when a serial port has a state change for Carrier Detect, Ring Indicator , DTR, etc. the SerialPort object propagates a SerialPortEvent that describes the state change.

A Simple reading Example

. Simple Read.java opens a serial port and creates a thread for asynchronously reading data through an event callback technique .

A Simple Writing Example

. Simple write java kopens a serial port for writing data.

9.2 SERIAL SUPPORT WITH JAVAX.COMM PACKAGE

Sun's Java soft division provide support for RS-232 and Parallel devices with standard extensions.

SUMMARY

One of the most popular interfaces on a PC is the serial port. This interface allows computers to perform input and output with peripheral devices. Serial interfaces exist for devices such as modems , printers, bar code scanners, smart card readers, PDA interfaces , and so on. Sun's javaSoft division recently has made available the javax. Comm. Packageto add serial support to java. This pckage provides support for serial and parallel devices using traditional java semantics such as streams and events.

In order to communicate with a serial device using a serial port on a host computer from a java application or applet, an devices connected to your serial port. In addition, the API provides a complete set of options for setting all of the parameters associated with serial and parallel devices. This artcle focuses on how to use javax comm. To communicate with a serial device based on RS-232; discusses what the javax.comm API does and does not provide; and offers a small example program that shows you how to communicateto the serial port using this API. We will end with a brief discussion of how this API will work with other device drivers, and also go over the requirements for performing a native port of this API to a specific OS. (2,700 words)

The java communications (a.k.a. javax.comm) API is a proposed standard extension that enables authors of communications applications to write java software that accesses communications ports in a platform-independent way. This API may be used to write terminal emulationsoftware, fax software, smart-card reader software, and soon. Developing good software usually means having some clearly defined interfaces. The high level diagram of the API interface layers are shown in this figure.

In the article we will show you how to use javax. Comm. to communicate with a serial device based on RS-232. We 'll also discuss what the javax.comm. API privides and what it doesn't provide. We'll present a small example program that shows you how to communicate to the serialport using

this API. At the end of the article we'll brifly detail how this javax.comm. API will work with other device drivers, and we'll go over the requirements for performing a native port of this API TO A specific OS.

Unlike classical drivers, which come with their own models of communication of asynchronous events ,the javax.comm API provides an event-style interface based on the java event model (java.awt.eventl package). Let's say we want to know if there is any new data sitting on the input buffer. We can find that out in two ways – by polling or listening. With polling , the lprocessor checks the buffer periodically to see if there is any new data in the hbuffer. With listening, the processor waits for an eventto occur in the form of new data in the input buffer.
As soon as new data arrives in the fuffer , it sends a nitification or event to the processor.

Dialer management and modem amangement are additional applications that can be written using the javax.. comm. API Dialer management typically provides and interface to the modem managfement's AT command interface,
. Almost all modems have an AT command interface. This interface is documented in modem manuals, perhaps a little example will make this concept clear,. Suppose we have a modem on COMI and we want to dial a phone number. A java dialer anagement application will query for the phone number and interrogatethe modem. These commands are carried by javax. Comm.. which does no interpretation. . To dial the number 918003210288, for example , the diale management probably sends an "AT" hoping to get back an "OK" followed by ATDT918003210288. One of the most important tasks of dialer management and modem amangement is to deal with errors and timeouts.

GUI for serial port managemen Normally, serial ports have a dialog box that congigures the serial [prts. Allowing users to set parameters suchas baud rate, parity, and so on . The following diagram depects the objects for X, Y, andZ modem protocols. These protocols provide support error detection and correction .

The programming basics

Too often, lprogrammers dive right into a project and code interactively with an API on the screen without giving any thought to the problem they are trying to solve. To avoid confusion and poptential problems, gather the following information before you start a project. Remember, programming

devices usually requires that you consult a manual. Get the manual ffor the device and read the section on the RS-232 interface and RS-232 protocol

Most devices have a protocol that must be followed. This protocol will be carried by the javax.comm API and delivered to the device. The device will decode the protocol, and you will have to pay close attention to sending data back and forth . Not getting the initial set-up correct can mean your application won't start, so take the time to test things out with a simple application. In other words, create an application that can simply write data onto the serial port and then read data from the serial port using the javax.comm. API .
Try to get some code samples from the manufacturer . Even if they are in another language, these examples can be quite useful. Find and code the samllestexample you can to verify that you can communicate with the device. In the case of serial devices, this can be very painful – you send data to device connected to the serial port and nothing happens . This is often the result of incorredt conditioning of the line. The number one rule of device programming (unless you are writing a device driver) is to make sure you can communicate with the device . Do this by finding the simplest thing you can communicate with the device. Do this by finding the simplest thing you can do with your device and getting that to work. If the protocol is very complicated, consider setting some RS-232 line analyzer software.

This software allows you to look at the data moving between the two devices on the RS-232 connection without intering with the transmission. Using the javax.comm API successfully in an application requires you toprovide some type of interface to the device protocol using the seril API as the transport mechanism. In other words , with the exception of the simplest devices, there is usually another layer required to format the data for the device. Of course the simplest protocol is "vanilla" meaning there is no protocol. You send and receive data with no I nterpretation.

9.3 OVERVIEW OF SUGGESTED STEPS FOR USING JAVAX.COMM

In addition to providing a protocol , the ISO layering model used for TCP/IP also applies here in that we have an electrilcal layer, followed by a very simple byte transport layer. On top of this byte transport layer you could put your transport layer. For example, your PPP stack ckould use the javax.comm API to transfer bytes back and forth to the modem. The role of

the javax.comm layer is quite small when looked at in this context: Give the javax. commAPI has to know about it . Open the decice and condition the line. You may have a device that requires a baud rate of 115 kilobits with no parity, Write some data and/or read data following whatever protocol the device you are communicating with requires . For example, if you connect to a printer, you may have to send a special code to start the printer and /or end the job. Some postscript printers require you to end the job by sending CTRL-D 0+03. Close the lport initializing the javax.comm API registry with serial interface ports

The javax.comm API can only manage ports that it is aware of . The latest version of the API does not require any ports to be initialized. On start up, the javax.comm API scans for ports on the particular host and adds them automatically. You can initialize the serial ports your javax.comm API can use. For devices that do not follow the standard naming convention writing and reading data fo javax.comm, this different than any other read and write method call to the derived output stream.

For write:

```
try  {
output.write( outputArray,0 .length );
```

For read:

```
Try  {
Int b = input.read()
```

Closing the port:

Closing the port with javax.comm is no different than with other requests to close device . This step is very important to javax.comm because it attempts to provide exclusive access. Multiplexing multiple users on a serial line requires a Multiplex or protocol.

```
Try  {

Inout.close();
Output.close();
} …
```

Chapter-10

10. OVERVIEW OF VOICE TRAMSMISSION OVER INTERNET

10.1 UNDERSTANDING JMF

javaTM Media Frame work (JMF) provides a unified architecture and messaging protocol for messaging the acquisition, processing, and delivery of time-based media data. JMF is designed to support most standard media content types, such as AIFF, AU,AVI,GSM,MIDI, MPEG,QuickTime, delivers the promise of " Write Once, Run AnywhereTM" to developers who want to use media such as audio and video in their java programs.JMF provides a common cross-platform javaAPI for accessing underlying meda frameworks. JMF implementations can leverage the capabilities of the underlying operating system, while developers can easily create portable jaba programs that feature time –based media by writing to the JMF API .

With JMF , you can easily create applets and applications that present, capture , manipulate, and store time-based media. The framework enables

advanced developers and technology providers to perform custom processing of raw media data and seamlessly extend JMF to support additional content types and formats, optimize handling of supported formats, and create new presentation mechanisms.
High –Level Architecture Devices such as tape decks and VCRs provide a familiar model for recording, processing , and presenting time-based media. When you play a movie using a VCR , you provide the media stream to the VCR by inserting videotape. The VCR reads and interprets the data on the tape and sends appropriate signals to your television and speakers.

JMF uses this same basic modil. A data source encapsulates the media stream much like videotape and a player provides processing and control mechanisms similar to a VCR . Playing and capturing audio and video with JMF requires the appropriate input and output devices such as microphone , cameras , speakers, and monitors. Data sources and players are integral parts of JMF's high-level API fo managing the capture ,presentation, and processing of time-based media. JMF also provides a lower level API that supports the seamless integration of custom processing components and extensions .

This layering provides java developers with an easy –to-use API for incorporating time-based media into java programs advanced media applications and future media technologies

10.1.1 TIME MODEL

JMF keeps time to nanosecond precision.A particular point in time is typically represented by a Time object, though some classed also support the specification of time in nanoseconds. Classes that support the JMF time model implement Clock to keep trach of time for a particular media stream. The Clock interface defines the basic timing and synchronization operations that are needed to control the presentation of media data.
A Clock uses a TimeBaseto keep track of the passage of time while a media stream is being presented . A timeBase provides a constantly ticking time source, much like a crysrtal oscillator in a watch. The only information that a TimeBase provides is its current time, which is referred to as the time-base time. The time base time cannot be stopped or reset. Time base time is often based on the system clock. A Clock objects media time represents the current position within a media stream –the beginning of the stream is media time zero, the end of the stream is the maximum media time for the stream . The duration of the media stream is the elapsed timefrom start to finish – the

length of time that it takes to present the media stream. (Media objects implement the Druation interface if they can report a media stream's duration) To keep track of the current media time , a Clock uses: The time-base start –time the time that its time Base reportswhen the presentation begins . the media start –time –the position in the media stream where presentation begins. The playback rate—how fast the Clock is running in relation to its Time Base. The rate is a scale factor that is applied to the TimeBase. For example , a rate of 1.0 represents the normal playback rate for the media stream , while a rate of 2.0 indicates that the presentation will run at twice the normal rate. A negative rate indicates that the Clock is running in the opposite direction from its TimeBase—for example, a negative rate might be used to play a media stream backward. When presentation begins, the media time is mapped to the time –base time and the advancement of the time base time is used to measure the passageof time. During presentation, the current media time is calculated using the following formula:
Media Time = Media Star Time + Rate(TimeBaseTime - TimeBaseStartTime)

When the presentation stops, the media time stops, but the time base time continues to advance. If the presentation is restarted, the media time is remapped to the current time-base time. Managers the JMF API consists mainly of interfaces that define the behavior and interaction of objects used to capture , process, and present time –based media . Implementations of these jinterfaces operate within the structure of the framework. By using intermediary objects called managers, JMF makes it easy to integrate new implementation of key interfaces that can be used eeamlessly with existing classes. JMF uses four managers: Manager—handless the construction of players. Processors, DataSources, and DataSinks. This level of indirection allows new implementations to be integrated seamlessly with JMF . From the client perspective, these objects are always created the same way whether the requested object is constructed from a default implementation or a custom one. PackageManager –maintins a registry of packages that contain JMF classes, such as custom Players, processors, DataSources, and DataSinks, CaptureDeviceManager—maintains a registry of available capture devices. PluginManager—maintains a registry of available JMF plug –in processing components, such as multiplexers, Demultiplexers, Codecs,Effecets, and Reindeers.

To wiite programs based on JMF , you 'll need to use the Manager create methods to construct the players, Proicessors, ddataSources, and DataSinks

for your application, If you're capturing media data from an input device , you'll use the captureDeviceManager to find out what devices are available and access information about them. If you 're interested in controlling what processing is performed on the data, you might also query the PlugInManager to determine what plug-ins have been registered. If you extend JMF functionally by implementing a new plug-in, you can register itwiht the PlugInManage to make it available to Processors that support the plug-in API . To use a custom Player, Processor, DataSource, or DataSink with JMF, you register your uniquepackage prefix with the PackageManager.

10.1.2 EVENT MODEL

JMF used a structured event reporting mechanism, to keep JMF –based programs informed of the current state of the media system and enable JMF –based programs to respond to media-driven error conditions, such as out-of data and resource unavailable conditions. Whenever a JMF object needs to report on the current conditions, it posts a MdeiaEvents.

MediaEvent is subclassed to identify many particular types of events. These objects follow the established java beans patterns for events. For each type kof JMF object that can post MediaEvents, JMF defines a corresponding listener interface. To receive notification when a MediaEvent is posted, yopu implement the appropriate listener interface and register your listener calss with the object that posts the event by calling its addListener method. Controller objects (such as Players and Processors) and certain control objects such as GainControl post media events.

RTPSessionManager objects also popst events. For more information, see RTP Events. DataModel JMF media playersw usually use DataSources to manage the transfer of media-content. A DataSource encapsulates both the location of media and the protocol and software used to deliver the media once obtained , the source
Cannot be reusedto deliver other media, a DataSource is identified by either a JMF MdeiaLocator or a URL (universal resource locator). A Media Locator is similar to a URL and can be constructed from a URL, but can be constructed even if the corresponsing protocol handler is not installed on the system. (In java, a URL can only be constructed if the corresponding protocol handler is installed on the system.) A DataSourcemanages a set of

SourceStream objects. A standard data source used a byte array as the unit of transfer. JMF defines several types of DataSource objects:

10.1.3 PUSH AND PULL DATA SOURCES

Media data can be obtained from a variety of sources, such as local or network files and live broadcasts. JMF data sources can be categorized according to how data transfer is jimitiated: Pull Data-Source—the client initiates the data transfer and controls the flow of data from pull datasourced. Established protocols for this type of data include Hypertext Transfer Orotocol (HTTP0 and FILE . JM defines two types of pull data sources: PullDataSource and PullBufferDataSource, which used a Buffer object as its unit of transfer. Push Data-Source—the server initiates the data transfer and controls the flow of data from a push data-source. Push data-sources include broadcast media, multicast media, and video-on-demand (VOD). For broadcast data, one protocol is the Real-time TransportProtocol (RTP) , under development by the Internet Engineering Task Force (IETF). The MediaBase protocol developed by SGI is one protocol used for VOD.

JMF defines two types of push data sources: PushdataSource and PushBufferDataSource, which uses a Buffer object as its unit of transfer. The degree of control that a client program can extend to the user depends on the type of data source being presented. For example, an MPEG file can be repositioned and aa client progaqm could allow the user to replay the video clip or seek to a new position in the video. In contrast, broadcastr media is under server control and cannot be repositioned. Some VOD protocols might support limited user control—for example, a client program might be able to allow the user to seek to a new position, but not fast forward or rewind.

10.1.4 SPECIALTY DATASOURCES

JMF defines two types of specialty data sources cloneable data sources and merging data sources. A cloneable data source can be used to create clones of either a pull or push DataSource. To create a cloneable DataSource, you call the Manager createcloneable DataSource method and pass in the DataSource you want to clone. Once a DataSource has hbeen passed to createCloneableDataSource, you should onlyinteract with the cloneable dataSource and its clones; the original DataSource should no

longer be used directly. Cloneable data sources implement the SourceCloneable interface , which defines one method, create Cline. By calling createClone, you can create any number of clones of the DataSource that was used to construct the cloineable DataSource. The clones can be controlled through the lconeable DataSourceused to create them—when connect, disconnect, start, or stop is called on the cloneable DataSource, the method calls are propagated to the clones.

The clones don't necessarily have the same properties as the cloneanble data source used to create them or the original DataSource. For example, a cloneable data source created for a capture device might function as a master data source for its clones—in this case, unless the cloneable data source is used, the clones won't produce any data. If you hook up both the cloneabledata source and one or more clones., the clones will produce data at the same rate as the master. A MergingdataSource can be used to combine the SourceStream from several DataSources into a single DataSource.

This enables a set of DataSources to be managed from a single point of control—when connect, disconnect, start , or stop is called on the MergingDataSource, the method calls are propagated to the merged DataSource,. To construct a Merging DataSource, you call the Manager createMerging DataSourcemethod and pass in an array that contains the data sources you want to merge . To be merged, all of the dataSources must be of the same type; The duration of the merged DataSource is the maximum of the merged DataSource objects' durations. The contenttype is application/mixed-media. Data formats the exact media format of an object is represented by a Format object. The format itself carries no encoding-specific parameters or global timing information, it describes the format's encoding name and the type of data tddhe format requires. JMF extends Format to define audio- and video-specific formats. An Audio Format describes the attributes specific to an audio format, such as sample rate, bits per samlple, and number of channels . A VideoFormat encalsulates information relevant to video data. Sevral formats are derived from videoFormat to describe the attributes of common video formats, including: Indexedcolorformat RGBGFormat YUVFormat JPEGFormat H261Format H263Format. To receive notification of format changes from a Controller, you implement the controllerListener interface and listen for

FormatChangeEvents. (For more information, see responding to Media Events.) controls JMF provides a mechanism for setting and querying attributes of anobject. A Control often provides access to a corresponding huser interface comlponent that enables user control over an object's attributes. Many JMF objects expose controls, including Controller objects, DataSource objects, DataSink objects and JMF plug-ins . Any JMF object that wants to provide access to its corresponding control objects can implement the controls interface. Controls defines methods for retrieving associated control objects. DataSource and PlugIn use the Controls interface to provide access to their control objects. Standard controls JMF defines the stardard Control interfaces shown in Figure 2-8; "JMF controls"

Caching control enables download progress to be monitored and displayed. If a Player or Processor can report its download progress, it implements this interface so that a progress bar can be displayed to the user. GainControl enables audio volume adjustments such as setting the level and muting the output of a Player or Processor . It also supports a limtener mechanism for volume changes.

DataSink or Multiplexer objects that read media from a DataSource and write it out to a destination such as a file can implement the StreamWritecontrol interface. This Control enables the user to limit the size of the stream that is created . FramePositioningControl and frameGrabbingContrtol enables precise frome positioning within a Player or Processor objects media stream. FrameGrabbingconrol provides a mechanism for grabbing a still video frame from the video stream . The FrameGrabbingControl can also be supported at the Renderer level. Objects that have a Format can implement the Format Control interface to provide access to the Format . Formatcontrol also provides methods for querying and setting the format. A TrackControl is a type of FormatControl that provides the mechanism for controlling what processing a Processor object performs on a particular track kof media data. With the Trackcontrol methods, you can specify what format conversions are performed on individual tracks and select the Effect, codes , or Renderer plug-ins that are used by the Processor . (For more information about processing media data, see Processing time – Based Media with JMF.

10.1.5 PLAYERS

A player processes an input stream of media data and rendes it at a precise time . A DataSource is used to deliver the input media – stream to the player. The rendering destination depends on the type of media being presented.

A Player does not provide any control over the processing that it performs or how it renders the media data. Player supports standardized user control and relaxes some of the operational restriction imposed by clock and Controller.

10.1.5.1 PLAYER STATES

A player can be in one of six states . The Clock interface the two primary states. Stopped and Started. To facilitate resource management, controller breaks the Stopped state down into five standby states: Unrealized, Realizing, Realized, Perfetching, and Prefetched .

In normal operation, A Player in the Unrealized state until it reaches the Started state: A Player in the Unrealized state has been instantiated, but does not yet know anything about its media . When a media Player is first created, it is Unrealzed .When realize is called , a player moves from the Unrealized state into the realizing state. A Realizing player is in the process of determining its resource requirements. During realization, a player acquires the resources that it only needs to acquire once. These might include rendering resources other than exclusive-use resources. (Exclusive-use resourcesare limited resources such as particular hardware devices that can only be used by one player at a time , such resources are acquired during prefetching.) a realizing player often downloads assets over the network.

When a Player finishes realizing., it jmoves jinto the realized state. A realized player known what resources it needs and information about the type of media it is to present. Because a Realized player known how to render its data, it can provide visual components and controls. Its connections to other objects in the system are in place, but it does not own any resources that would precvent another player from starting. When prefetching state. A prefetching player is preparing to present its media . During this phase , the player preloads its media data, obtains exclusive-use resources, and does whatever else it needs to do to prepare itself to play. Prefetching might have to recur if a player objects media presentation is

repositioned, or if a change in the player objects raterequires that additional buffers be acquired or alternate processing take place. When a player finishes prefetching, it moves into the prefetched state. A prefetched player is ready to be started. Calling start puts player into the started state. A started player objects time-base time and media time are mapped and its clock if running, though the player might be waiting for a particular time to begin presenting its media data. A player postsTransitionEvents as it moves from one state to another . The controllerListener interface privides a way for your program to determine what state a player is in and to respond appropriately. For example , when yourn program calls an asynchronous method on a player or processor, it needsto listen for the appropriate event to determine when the operation is complete. Using this event repoting mechanism, you can manage a player objects start latency by controlling when it begins Realizing and prefetching. It also enables you to determine whether or not the player is in an appropriate state before calling methods on the player.

10.1.6 PROCESSORS

Processors can be used to present media data. A processor is just a specialized type of player that provides control over what processing is performed on the input media stream . A processor supports all of the same presentation controls as a player .

In addition to rendering media data to presentation devices , a processor can output media data through a DataSource so that it can be presented by another player or processor, further mamipulated by another processor, or delivered to some other destination , such as a file . For more information about processors, see processing . Presentation Controls in addition to the standard presentation controls defined by Controller, a player or processor might also provide a way to adjust the playback volume. If so, you can retrieve its GainControl by calling getGainControl. A Gain Control object posts a GainChangeEvent whenever th gain is modified . By implementing the GainChangeListener interface, you can respondto gain changes. For example , you might want to update a custom gain control Component . Additional custom Control types might be supported by a particular player or processor implementation to provide other control behaviors and expose

custom user interface comlponents. You access controls through the getControls method. For example , the Cachingcontrol interface extends control to provide a mechanism for displaying a download progress bar. If a player can report its download progress, it implements this interface. To find out if a player supports Caching control, you can call getControl(CachingControl) or use interface components A player or processor generally provides two standard user interface components, a visual component and a control-panel component. You can access these components directly through the get visualComponent and get ControlPanelcomponent methods.

10.1.7 PROCESSING

A processor is a player that takes a DataSource as input , performs some user-defined processing on the media data, and then outputs the processed media data.

A processor can send th output data to a presentation device or to a DataSource. If the data is sent to a DataSource, that Datasource can be used as the input to another player or processor , or as the unput to a dataSink. While the processing performed by a player is predefined by the implementor, a processor allows the application developer todefine the type of processing that is applied to the media data. This enables the application of effects, mixing , and composition in real-time. The processing of the media data is split into several stages:

Demulplexing is the process of parsing the input stream . If the stream contains multiple tracks, they are extracted and output separately. For example , a a Quicktime file might be demultiplexed into separate audio and video tracks. Demultiplexing is performed automatically whenever the input stream contains multiplexed data. Pre-precessing is the process of applying effect algorithms to the tracks extracted from the input stream. Transcodding is the process of converting each track of media data from one input format to another . when a data stream is converted from a compressed type to an uncompressed type , it is generally referred toas decoding. Conversely, converting from an uncompressed type to a compressed type is referred to as encoding . tracks. Multiplexing is the process of interleaving the transcoded media tracks into a single output stream. For example , separate audio and

video tracks might be multilplexed into a single MPEG-1 data stream . You can specify the data type of the output stream with the processor setOutput contentDescriptor method . Rendering is the process of presenting the media to the user. The processing at each stage is performed by a separate processing component. These processing components are JMF plug-ins . If the processor supports Trackcontrols, you can select which plug-ins you want to use to process a particular track. There are five types of JMF plug-ins: demultiplexer—parses media streams such as WAV, MPEG or Quicktime . If the streamis multiplexd, the separate tracks are extracted . Effect—performs special effects processing on a track of media data. Codec—performs data dencoding and decoding.

Multiplexder—combines multiple tracks of input data into a single interleaved output stream and delivers the resulting stream as an output DataSource. Renderer—processes the media data in a track and delivers it to a destination such as a screen or speaker . Processor States A processor has two additional standby states, Configuring and configured, which occur before the processor enters the realizing state.

A processor enters the configuring state when configure is called . While the processor is in the configuring state, it connects to the Datasource, demupliplexes the input stream , and accesses information about the format of the input data. The processor moves into the configured state when it is connected to the DataSource and data format has been determined. When the processor reaches the Configured state, a Configured complete Events is posted. When realize is called, the processor is transitioned to the realized state. Once the processor is realzed it is fully constructed . While a processor is in the configured state, get Trackcontrols can be called to get the Trackcontrol object for the individual tracks in the media stream . These TrackControl objects enable you specify the media processing operations that you want the processor to perform. Calling realize directly on an Unrealized processor automatically transitions it through the Configuring and configured states to the realized state. When you do this , you cannot configure the processing options through the TrackControls –the default processor settings are used. Calls to the TrackControl method once the processor is in the Realized state will typically fail though some processor implementations might support them.

10.1.7.1 METHODS AVAILABLE IN EACH PROCESSOR STATE

Since a processor is a type of player , the restrictions on when methods can be called on a player also apply to processors. Some of the processor-specific methods also are restricted to particular states. The following table shows tdhe restrictions that apply to a processor . If you call a method that is illegal in the current state, the processor throws an error or exception.

10.1.8 PROCESSING CONTROLS

You can control what processing operations the processor performs on a track through the TrackControl for that track. You call processor getTrackControls to get the Trackcontrol , you can explicitly select th effct, Codec, and Renderer plug-ins you want to use for the track. To find out what options are available , you can query the pluginmanager to find out what plug-ins are installed . To control the transcoding that,s performed on a track by a particular Codec, you can get the controls associated with the track by calling the TrackControl getControls method.

This method returns the codec control. (For more information about the codec controls defined by JMF, see controls.) If you know the output data format that you want , you can use the setformat method to specify the Format and let the Processor choose an appropriate codec and renderer, Alternatively, you can specify the output format the processor is created by using a processorModel.

A processor Model defines the input and output requirements for a processor . When a processor Model is passed to the appropriate Manager create method , the Manager does its best to create a processor that meets the specified requirements. Data Output as a DataSource. This DataSource can be used as the input to another player or processor or as the input to a data sink. (For more information about data sinks see Media Data Storage and Transmission) A processor object’s output DataSource can be of any type: pushDataSource , pushBuffer DataSource, pullDataSource, or pullBuffer DataSource . Not all Processor objects output data –a processor can render the processed data instead of outputting the data to a DataSource. A processor that renders the media data is essentially a configureable player . Capture A multimedia capturing device can act as a source for multimedia data delivery. For example , a microphone can capture raw audio input or a digital video capture board might deliver digital video from a camera. Such

capture devices are abstracted as DataSource . For example , a a device that provides timely delivery of data can be represented as a pushdatasource . Any type of DataSource can be used as a capture DataSource: Push datasource. Any type of datasource can be used as a capture DataSource : pushdatasource, PushBufferDataSource,PulldataSource, or PullBuffer DataSource. Some devices deliver multiple data stream –for example , an audio/video conferencing board might deliver both an audio and a video stream. The corresponding DataSource can contain multiple SourceStreams that map to the data streams provided by the device . Media dataStorage and transmission.

A DataSink is used tko read media data from a DataSource and render thde media to some destination—generally a destination other than a presentation device A particular DataSink might write data to a file, write data across the network , or function as an RTP broadcaster , see transmitting about using a DataSink as an RTP broadcaster , see Transmitting RTP Data with a Data Sink.) Like player s, DataSink objects are construced through the Manager using a DataSource .

A DataSink can use a stream writer control to provide additional control over how data is written to a file . See Writing Media Data to file for more information about how DataSink objects are used . Storage Controls A DataSink posts a DataSinkEvent objects are used. Storage controls A DataSink posts a DataSinkEvent to report on its status. A DataSinkEvent can be posted with a reason code, or the DataSink can post one of the followingDtaSinkEvent subtypes: DataSinkErrorEvent, which indicates that an error occurred while the DataSink was writing data. EndOfStreamevent, which indicates that th entire stream has successfully been written.

To respond to events posted by a DataSink, you implement the DataSinkListener interface. Extensibility you can extend JMF by implementing custom plug-ins By implementing one of the JMF plug-in interfaces, you can directly access and manipulate the media data associated with a processor : Implementing the Demultiplexer interface enables you to control how individual tracks are extracted from a multiplexed media stream. Implementing the Codec interface enables your to perform the processing repuired to decode compressed media data, convert media data from one format to another , and encode raw media data into a compressed format . Implementing the Effect interface enables you to perform custom processing on the media data. Implementing the Multiplexer interface

enables you to specify how individual tracks are combined to form a single interleaved output stream for a processor, Implementing the renderer interface enables you to control how data is processed and rendered to an output device. Note: The JMF plug –in API is part of the official JMF API , but KMF players and processors are not required to support plug-ins. Plug-ins won't work with JMF 1.0 based players and some processor implementations might choose not to support them. The reference implementation of JMF 2.0 provoded by Sun Microsystems, Inc and IBM corporation fully supports the plug-in API custom codec, effect, and renderer plug-ins are available to a processor through the trackControl interface. To make a plug –in available to a default processor or a processor created with a processor Model , you need to register it with the plugInManager . Once you've registered your plug-in, it is included in the list of plug-ins returned by the PlugInManager getpluginList method and can be accessed by the Manager when it constructs a processor object. Implementing MediaHandlers and dataSource .

If the JMF plug-in API doesn't provide the degree of flexibility that you need, you can directly implement several of the ksy JMF interfaces: ckontroller, player processor DataSource, and DataSink. For example , you might want to implement a high performance player that is optimzed to present a single media format or a controller that messages a completely different type of time based media .

The Manager mechanism used tonconstruct player , processor DataSource, and DataSink objects enables custom implementations of theseJMF interfaces to be used seamlessly with JMF . When one of the create method is called , the Manager uses a well-defined mechanism to locateand construct the requested object. Your custom class can be selected and constructed through this mechanism once you register a unique package prefix with the package Manager and put your class in the appropriate place in the predefined package hierarchy. MediaHandler construction player processors, and a DataSink s are all types of MediaHandlers—they all read data from a DataSource. A MediaHandler is always constructed for a particular DataSource, which can be either identified explicitly or with a MediaLocater . When one of the createMediaHandler methods is called , Manager uses the content-type name obtained from the DataSource to find and create an appropriate Media Handler object.

JMF also supports another type kof MediaHandler , Mediaproxy. A Mediaproxy processes contain from one DataSource to create another Typically , a MediaProxy reads a text configuration file that contains all of the information needed to make a connection to a server and obtain media data. To create a player from a mediaproxy , Manager: constructs a DataSource for the protocol described by the medialocator uses the content-type of the DataSource to construct a mediaproxy to read the configuration file. Gets a new DataSource from the MediaProxy . Uses the content-type of the new DataSource to construct a player.

The mechanism that Manager uses to locate and instantiate an appropriate Media Handler for a particular DataSource is basically the same for all types of Media Handlers: Using the list of installed content package –prefexes retrieved from package Manager , Manager generates a search list of available MediaHandler classes . Manager steps through each class in the search list until it finds a class named Handler that can be constructed and to which it can attach the dataSource.

Chapter-11

11. REAL TIME PROTOCOL

11.1 INTRODUCTION

The real tidme transport protocol provides end-to-end delivery Services for data with real time characteristics, such as interactive audio and Video. Those services include payload type identification, sequence numbering, times stamping and delivering monitoring. Application typically run RTP on top OF UDP to make use of its multiplexing and checksome services, both protocol contribute parts of the transport protocol functionally. However RTP may be used with other suitable under lying nerwork or transport protocols.RTP supports data transfer to multiple destination using multicast distribution if provided by the network.

Note that RTP itself does not provide any mechanism to ensure timely delivery or provide other quality of service guarantees, but relies on lower layer services to do so. It does not guarantee delivery or prevent out of order delivery , nor does it assume that the under lying nerwork is reliable and

delivers packets in sequence. The sequence number included in RTP allow the receiver to reconstruct the sender's packet sequence, but sequence number might also be used to determine the proper locationof a packet , for example in video decoding, without necessary decoding packets in sequence . While RTP is primarily designed to satisfy the needs of multi participant multimedia conferences, it is not limited to that particular applicipant multimedia conferences, it is not limited to that particular application. Strong of continuous data, interactive distributed simulation, active badge, and control and measurement applications may also findRTP applicable.

RTP consists of two closely linked parts:

The real time transport protocol (RTP) , to carry data that has real time properties.

The RTP control protocol (RTCP), to monitor the quality of service and to convey information about the participants in an on-going session.

The later aspect of RTCP may be sufficient for "looselycontrolled" sessions, i.e., where there is no explicit membership control and set-up , but it is not necessarily intend to support all of an application's control communication requirements.

RTP is intended to be malleable to provide the information by a particular application and will often be integrated into the application processing rather than being implemented as a separate layer. RTP is a protocol framework that is deliberately not complete.

11.2 RTP USE SCENARIOS

The following sections describe some aspects of use of RTP. The example were chosen to illustrate the basic operation of applications using RTP, not to limit what RTP may be used for . In these examples RTP is carried on top of IP and UDP, and follows the vonventions established by the profile for audio and video.

11.2.1 SIMPLE MULTICAST AUDIO CONFERENCE

A working group of the IETF meets to discuss the latest protocol draft, using the IP multicast services of the Internet for voice communications . Through some allocation mechanism the working group chair obtains a multicast group address and pairs of ports. One port is used for audio data , and the other is used for control (RTCP0 packets.

This address and port information is distributed to the intended participants. If privacy is desired the data and control packets may be encrypted in which case an encryptin key must also be generated and distributed. The exact details of these allocations and distribution mechanisms are beyond the scope of RTP.

The audio conferencing application used by each conference participant sends audio data in small chunks of say 20 ms duration . Each chunk of audiodata is preceded by an RTP header and data are in turn contained in a UDP packet. The RTP header and data are in turn contained in a UDP packet. The RTP header indicates what type of audio encoding is contained in each packet so that senders can change the audio encoding is contained in each packet so that senders can change the encoding during a conference , for example , to accommodate a new participant that is

connected through a low – bandwidth link react to indications of network congestion.

The internet like other packet networks occasionally loses and records packets and delays them by variable amounts of time . To cop with these impairments, RTP header contains timing information.

This timing reconstruction is performed separately for each source of RTP packets in the conference. The sequence number can also be used by the receiver how many packets are being lost.

Since members of working group leave and join during the conference, it is useful to know who is participating at any momentand how well they are receiving thde audio data. For that purpose each instance of the audio application in the conference periodically multicasts a reception report plus the name of its user on the RTCP control port. The reception report tells how well the current speaker is being received and may be used to control adaptive encoding . In addition to the user name, other identifying information may be included subject to control bandwidth limits, A site sends the RTCP BYE packet when it leaves the conference.

11.2.2 AUDIO AND VIDEO CONFERENCE

If both audio and video media are used in a conference, they are transmitted as separate RTP session RTCP packets are transmitted for each medium using two different UDP port pair and multicast address. There is no direct coupling at the RTP level between the audio and video sessions , except that a user participating in both sessions should use the same distinguished name in the RTCP packets for both so that the sessions can be associated . One motivation for this separation is to allow some particilpants in the conference to receive only one medium if they choose . Despite the separation, synchronized playback of the source's audio and video can be achieved using timing information carried in the RTCP packets for both sessions.

11.2.3 MIXERS AND TRANSLATORS

So for we have assumed that all sites want to receive media data in the same format . However this may not always be appropriate .

Consider the case where the participants in one area are connected through a low speed link to the majority of the conference participants who enjoy high –speed network access. Instead of forcing everyoneto use a lower bandwidth, reduced quality audio encoding, an RTP level realy called a mixer may be placed near the low bandwidth area, This mixer resynchronizes incoming audio packets to reconstruct the constant 20 ms spacing generated by the senders, mixers these reconstructed audio streams in to a single stream, translates the audio encoding to a lower bandwidth one and forwards the lower bandwidth packet stream across the low speed link.

These packets might be unicast to a single recipient or multicast on a different address to multiple recipients . The RTP header includes a means for mixers to identify the sources that contributed to a mixed packet so that correct talker indicatiohn can be provided at the receivers . Some of the intended participants in the audio conference may be connected with high bandwidth links but might not be directly reachable via IP multicast. For example, they might be behind an application level firewall that will not let any IP packets pass. For these sites mixing may not be necessary, in which case another type of RTP level really called a translator may be used. Two translators are installed one on either side of the firewall, with the outside one funneling all multicast packets received through a secure connection to

the translator inside the firewall. The translator inside the firewall sends them again as multicast packets to a multicast group restricted to the site's internal nerwork . Mixers and translators mat be for a variety of purposes . An example is a video mixer that scales the images of individual people in separate video streams and composites them into one video stream to stimulate a group scene. Other examples of translation include the connection of group of hosts speaking only IP/UDP to a group of hosts that understands only ST-11 or packet-by –packet encoding translationof videostreams from individual sources without resynchronization or mixing.

11.3 DEFINITIONS

. RTP Payload: The data transported by RTP in a packet, for example audio samples or compressed video data. The payload format and interpretation is beyoned the scope of this document.

. RTP Packet: A data packet consisting of the fixed RTP header, a possibly empty list of contributing sources and the payload data.

Some underlying protocols may require an encapsulation of the RTP packet to be defined . Typically one packet of the underlying protocols contains a single RTP packet, but several RTP packets may be contained if permitted by the encapsulation method.

. RTCP Packet: A control packet consisting of a fixed header part similar to that of RTP data packets, followed by structured elements that varies depending upon the RTCP packet type. Typically, multipleRTCP packets are sent together a compound RTCP packet in a single packet of the underlying protocol: this is enable by the length

Field in the fixed header of each RTCP packet.
Port: The abstraction that transport protocol used to distinguish among multiple destination within a given host computer.

. TCP/IP Protocols: identify ports using small positive integers. "The transport selectors used by the OSI transport layer are equivalent to ports. RTP depdends upon the lower layer protocol to provide some mechanism such as ports to multiplex the RTP and RTCP packets of the session.

. Transport address: The combination of a network address and port that identifies a transport level endpoint , for example an IP address and a UDP port. Packets are transmitted a source transport address.

.RTP session: The association among a set of participants communicating with RTP . For each partcilpant , a particular pair of destination transport address defines the session. The destination transport address may be common for all participants , as in the case of IP multicast network address plus a common port pair . In a multi cast session, each medium is carried in a separate RTP session with its own RTCP packets . the multiple RTP session are distinguished by different port mumber pairs and different multicast addresses. Synchronization source (SSRO): The source of a stream of RTP packet , identified by a 32 bits numeric SSRC identifier carried in the RTP header so as not to be dependent upon the network address. All packets from a synchronization source form part of the same timing and number space, so a receiver groups packet by a synchronization source for play back. Examples of synchronization sources include the sender to stream of packets derived from a signal source such as microphone.

The SSRC is a randomly chosen value meant to globally unique with a particular RTP session . If a participant gererates multiple streams in RTP session, e.g from separate video cameras, each must be identified as different SSRC.

. End System: An application hthat gereates the content to be sent in RTP packets and / or consumes the contents of received RTP packets. An end system can act as one kor more synchronization in particular RTP session, but typically only one.

. Mixer: An intermediate system that receives RTP packets from one more sources possibly changes the data format, combines the packets in some manner and then forwards a new RTP packets in some manner and then forwards a new RTP packets. Since the timing among nultiple inputnsources, the mixer will make timing adjustments and generate its own timing for combined stream. Thus all data packets originating from a mixer will be identified as having the mixer as their synchronization source

. TRANSLATOR: An intermediate system that forwards RTP packets with their synchronization with their souce identifier interacts Examples of

translators are replicators from multicast to unicast and application level filter in firewalls.

. MONITOR: An application that receives RTCP packets sent by participants in an RTP session , in particular the reception reports, estimates the current quality. The monitor function is likely to be built into the application participating jin the session and does not send or receive the KTP data packets. These are called hird party monitors.

. NON- RTP-means: Protocols and mechanisms that may be needed in addition to RTP to provide a usable service. In particular for multicast addresses ann keys for encryption formats that represents simple applications electronic mail or a conference database may also be used, The specification of each of such protocols and mechanisms is outside the scope of this document.

. BYTE ORDER, ALIGNMENT, and TIME FORMAT: All integer fields are carried in network byte order that is most significant byte first. This byte order is commonly known as big endian.

The transmission order is described in detail in. All header data is aligned to its natural lengthi.e 16-bit fields are aligned on even offsets, 32 bits fields aligned at offsets divisible by four ,etc. octectrs designated as padding havce tdhe value zero.
Wall clock time (absolute time) is represented using the time stamp format of the network time protocol (NTR) . the fuyll resolution NTP timestamps is a 64-bit unsigned fixed point number with integer part in its 32 bits and fractional part in the last the low 16 bits of the integer part and the high bits of the fractional part . The high 16 bits of the integer part be determined independently.

11.4 MULTIPLEXING RTP SESSIONS

For efficient protocol processing the number of multiplexing point should be minimized as described it the integrated layer processing design principle. In RTP multiplexing is provided by the destination transport address, which define an RTP session, For example in a teleconference composed of Audio and Video media encoded separately each medium shouls be carried in a separate RTP session with its own destination transport address. It is not intended that the audoio and video be carried in a

single RTP session and demultiplexed based on the payload type or SSRC fields. Inter leaving packets with various payload typed but using the same SSRC would introduce several problems:

If one payload type were switched during a session there would be no general means to identify which of the old values the new one replaced . An SSRC is defined to identify a single timing and sequence number space. Inter leaving multiple paypoad types would require different timing spaces if the media lock rates differ would require different sequence number spacees to tell which load type suffered packet loss.

The RTCP sender and receiver reports can only describe one timing and sequence number space per SSRC and do not carry a payload type failed. An RTP mixer would not be able to combine interleaved streams of incompatible media jinto one stream.

Carrying multiple media in one RTP session preclude:

The use of different network paths or network resource allocations in appropriate reception of a subset of the media if desired, if for example just

audio if video would exceed the Availbale bandwidth and receiver implementations that use separate processes for different media , where as using separate RTP sessions permits either single or multiple process implementations .

Count of sequence number cycles. Note that different receivers within the same session will generate different extensions to the sequence number if their start times differ significantly. It is expected that reception that reception quality feedback will be useful not only for the sender but also for other receivers and third party monirots . The sender may modify its transmissioins are local regional or global; network managers may use profile independent monitos that receive only the RTCP packets and not the corresponding RTP data packets to evaluate the performance of their networks for multicast distribution.

Cumulative counts are used in both the sender information and receiver report blocks so that differences may be calculated beyween any two reports to make measurements over both short and longtime periods and to provide resilience against the loss of a report The difference between the

last two reports received can be used to estimate the recent quality of the distribution . The NTP timestamps is included so that rates may be calculated from these independent of the clock rate for the data encoding, it is possible to implement encoding and profile independent quality monitors . An example calculation is the packet loss rate over the interval between two reception reports. The difference in the cumulative number of packets lost gives the number lost during that interval. The difference in the extended last sequence numbers received gives the number of packets expected during the interval. The ratio of these two is the packet loss fraction over the interval. This ration should equal the fraction lost field if the two reports areconsecutive, but otherwise not. The loss rate per second can be obtained by dividing the loss fraction by the difference in NTP timestamps expressed in seconds . The number of packets received is the number of packets expected minus the number lost. The number of packets expected may also be used to judge the statistical of any loss estimates. For example , 1 kout of 5 packets lost has a lower significance than 200 out of 1000.

From the sender information a third party monitor can calculate the average payload data rate and the average packet rate over an interval without receiving the data. Taking the ratio of the two gives the average payload size, if it can be assumed that packet loss is independent of packet size, then the number of packets received by particular receiver times the average payload size(or the corresponding packet size) gives gives the apparent throughput available to that receiver . In addition to the cumulative counts, which allow long term packet loss measurement using differences between report. The fraction lost field provides a short term measurement from a single report. This becomes more important as the size of a session scales up enough that reception state information might not be kept for all receivers or the interval between reports becomes long enough that only one report might have been recived from a particular receiver.

The inter arrival jitter field provides a second short term measure of network congestion packet loss tracks persistent congestion while the jitter measure tracks transient congestion. The jitter measure may indicate congestion before it leads to packet loss . Since the inter arrival jitter field is only a snapshot of the jitter at the time of a report , it may be necessary to

analyze a number of reports from one receiver over time or from multiple receivers, e.g, within a single network.

11.5 RTP PROFILES AND PAYLOAD FORMAT SPECIFICATION

A complete specification of RTP for a particular application will require one or more companion documents of two types dscribed here: Profiles, and payload format specifications'.

RTP may be used for a variety of application with somewhat differing requirements . the flexibility to adapt to the 09se requirements is provided by allowing multilple choices in the main protocol specification, then selecting the appropriate in a separate profile document. Typically an application will operate under only one profile so there is no explicit indication of which prefiles is un use. A profile draft ieft avt-profile for the second type of companion document is payload format specification, which defines how a particular kind of payload data, such as H261 encoded video should be carried in RTP . These documents are typically titled "RTP payload format for XYZ audio /video Encoding " Payload formats may be useful under multiple profile and may therefore be define independently of any particular

profile . The documents are then responsible for assigning a default mapping of that format toa payload type value if needed.

Within this specification the following items have been identified for possible definition within a profile but this list is not meant to be exhaustive.

RTP data header : the octet in the RTP data header that contains the marker bit and payload type field may be redefined by a profile to suit different requirements, for example with more or fewer market bits.

Payload types :Assuming that a payload type field is included, the profile will usually define a set of payload formats (E.G, media encoding) and a default a static mapping of those formats to payload type values . Some of the payload formats may be defined by reference to separate payload format specifications . For each payload type define the profile must specify the RTP timestamp clock rate to be use.

RTP data header additions Additional fields may be appended to the fixed RTP data header if some additional functionality is required across the profile 's class of applications independent of payload type.

RTP data header extensions ; the contents of the first 16 bits of the RTP data header extension structure must be defined if use of that mechanism is allowed under the profile for implementation-specific extensions.

RTCP report interval : A profile should specify that the values suggested for the constants employed in the calculation of the RTCP report interval will be used . Those are the RTCP fraction of session bandwidth the minimum report interval , and the bandwidth split between senders and receivers. A profile may specify alternate values if they have been demonstrated to work in a scalable manner.

.

Chapter-12

12. Communication API

This chapter covers the Java communication API 2.0, a standard extension available in Java 1.1 and later that allows Java applications (but not applets) to send and receive data to and from the serial and parallel ports of the host computer. The Java communication API allows Java programs to communicate with essentially
and device connected to a serial are parallel port, like a printer, a scanner, a level. It only understands how to send and receive bytes to these ports. It dose not understand anything about what these bytes mean. Doing useful work generally requires not only understanding the Java communications API (which is actually quite simple) but also the protocols spoken by the devices connected to the ports (which can be almost arbitrarily complex)

12.1 The Architecture of the Java Communications API

Because the Java Communications API is a standard extension, it is not installed by default with the JDK You have to download it from and install it separately.

The Java communications API contains a single package ,javax. Comm, which holds a b because the communicationaker' dozen of classes, exceptions,and interfaces. Because the communication API is a standard extension, the javax prefix is used instead of the java prefix. The Java Communication API also includes a DLL, or shared library, containing the native code to communicate with the ports, and a few driver classes in the com.Sun.comm package that mostly handle the vagaries of Unix or Wintel ports. Other vendors may needs to muck around with these if they're porting the Comm API to another platform (e.g.,the Mac or OS/2), but as a user of the API,you'll only conceal

Yourself with the documented classes in javax.comm.

Javax.comm is divided into high level and low level classes. High level classes or responsible for controlling access to and ownership of the comm ports and performing basic I/O.. The Comm port identifier class lets you fined and open the ports available on a system.

The Comm port class provides input and output streams connected to the ports Low level classes Java Comm serial port and Java comm. parallel port, for example manage interaction with particular kinds of ports and help you need and write the control wires on the ports. They also provide event based notification of changes to the state of the ports.

Version 2.0 of the Java Comm API understands RS-232 serial ports and IEEE 1284 type parallel ports. Future releases may add support for other kinds of ports like the universal serial Bus (USB). Fire Wire, or SCSI.

12.2 Identifying Ports

The Javax.Comm. CommportIdentifier class is the control room for the ports on a system. It has methods that list the available ports. Figure out which program owns them take control of a port. And open a port so you can perform I/O with it. The actual

I/O, stream-based or otherwise, is performed through an instance of Java. Comm.

Comm port that represents the port in question. The purpose of Comm port identifier is to mediate between different programs, objects, or threads that want to use the same port

12.3 Finding the ports

Before you can use a port, you need a port. Because the possible port identifiers are closely tied to the physical port on the system, you can not simply construct an arbitrary
Comm port identifier object. (For instance, Macs have no parallel port and I Macs don't have serial parallel ports.) Instead you use one of several static methods in Java. Comm. Comm ports Identifier that use native methods and non publics constructors to find and create the right port.
These include:
Public static Enumeration gets port identifiers ()
Public Static Comm Port Identifier gets port identifier (String port name)
Throws no such port exception

Public Static Comm port identifier get port identifier (Comm port)
Throws no such port exception

The most general of these is Comm port identifier. Gets port identifier ()
Which return a Java. Utile. Enumeration containing one Comm port identifier for each of the port on the system port lister
Import java Comm*
Import Java. Util*
Public class port lister
Public static void main(string[] args)
Enumeration e = comm port identifier. Get port identifier( )
While (e.has more elements)
System. Out. Print in (comm port identifier) e.next element ( )

You can also get a Comm port identifier by using the static method get port identifier to request a port a identifier,either by name or by the actual port object. The latter assumes that you already have a reference to the relevant port,which usually isn't the case. The former allows you to choose from Windows standard names like"COMI"LPT2 or Unix names

like”SerialA”and”SERIAL B.” The exact formatof a name is highly platform-and implementation-dependent. If you ask for a port that doesn’t exist, a NoSuchportException is thrown.following programme looks for serial and parallel port by starting with COMI and LPTI and counting up until one is missing.

```
Namedportlister
Import javax,comm,*:
Public class Namedportlister(
Public statie void main(Stringl) args) (

// List serial (COM) ports.

Try (

Int portNumber=1:

While (true) (

CommportIdentifier,getportIdentifier(“COM” + portNumber):

System.out,println(“COM”-portNumber):


PortNumber++:

Catch  (NoSuchportExceptione ) (

// Break out of loop.

List parallel (LPT) ports.

Try (

Int portNumber=1

While (true)

Commport. Identifier. GetportIdentifier (LPT portNumber)

System. Out. Printin (LPT-portNumber)

PortNumber++)
```

Example 17.2.Namedportlister (continued)

Catch (Nosuchportexception e)

// Break out of loop.

Once again,here's the output from a stock Wintel box :

D:/JAVA/16.>java Namedportlister

COM1

COM2

LPT1

LPT2 on Unix, because it relies on

Now you can see that I have two serial and two parallel ports. H owever,this same program would fail hard-wired port names

12.4 Getting Information about a Port.

Once you have a commportIdentifier identifying a particular port,you can discover information about the port by calling several accessor methods. These include:
Public string getname()
Public intgetporttype( )
Public string getcurrentowner( )
Public boolean iscurrentlyowned( )
The getName ( )method returns the platform –dependent name of the port, such as COMI(windows) " Serial A"(Solaris) or "modem"(Mac)* the getportType( )method returns one of the two mnemonic. Constants CommportIdentifier. PORT-SERIAL or
CommportIdentifier. PORT-PARALLEL:
Public static final int PORT-SERIAL=1
Public static final int PORT-PARALLEL=2

The isCurrentlyowned () method returns true if some process, thread,or application Enumeration e=comport currently has control of the port.Itretuns

false otherwise.If a port is owned by another java program. The getcurrentOwner() returns the name supplied by the program .the get currentOwner ()returns the name supplied by the program that owns it :Otherwise,it returns null.This isn't to useful because it does't handle the much more likely case that a non-Java program like Dial-Up Networking or ppp is using the port .A comments in the source code indicates that this should be fixed so that non –Java programs can also be identified.

Is a revision of the portlister in the initial programme that uses these four accessor methods to provide information about each port rather than relying on the inherited tostring()method.

```
pretty portlister
Import javax comm..*;
import java. Util.*;
Public class pretty portlister
Public static void main(string[ ] args)
Enumeration e=commportIdentifier.getport identifiers():
While (e.hasmoreEmemts( ))
Commportidentifier com=(commportidentifier)e. nextElement()
system.out.print(com.getName( ));

switch(com,getporttype( ))
Casecomportidentifier.PORT-SERIAL:
System,out,print('a serialport")
Break:
Casecommportidentifier,PORT-parallel
System,out,print("a parallel port")
Break:
Default
//important since other types of ports like USB
//and firewire are expected to be added in the future,system,out,print("a port of unknown type,"):
break if(com,isCurrentlyOwned)
system,out,printin("is currentlyowned by"
+com,get current Owner)+")
else
system,out,printin("is not currently owned")
Here,s the out put when run on astock wintel box
```

This output originally confused me because I expected one of the COM ports to be occupied by the Dial-Up Networking PPP connection on the internal modem COM2),However, thejsCurrentlyOwned ( ) method only notices other Java programs in the same VM occupying port.To detect whether anon-Java programn a controlling a port you must try to open the port and watch for portInUseExceptioins,as discussed in the next section:

12.5 Opening Ports

Before you can read from or write to a port,,you have to open it. Opening a port gives your application exclusive access to the port until you give it up or the [rogram ends,(Two different should not send data to the same modem or printer at the same time,after all) Opening a port is not guaranteed to succeed. If another program (Java or otherwise) is using the port a portInUseException will be thrown when you try to open the port. Surprisingly. This is not a subclass of IOEXception.Public class portinUseException extends Exception CommportIdentifier has two open () methods, they return a javax,comm,commport object you can use to read data from and write data to the port. The first variant takes two arguments, a name and atme-out value:public synchronized commport open string name, int timeout throws portinuseexception the name argument is a name for the

program that wants to use the port and will be returned by getCurrentOwner()while the port is in use.The time out argument is the mesimum number of milliseconds this method will block while waiting for the port to become available.If the operation does not complete with in tht time,a portInUseException is thrown.example17-1 is a variation of the portlister program that attempts to open each unowned port
example 17-4 portOpener
import, javax, comm
import java util
public class portopener
public static void main(string()args)(
enumeration theport=commportidentifier get port identifers()while(the ports,hasnoreElements()commportidentifier
com=(commportidentifier)theports,next elements()
system.out ,print(com,getName():
example17-4port opener(continued)switch(com,getporttype()
case commportidentifier ports serial:
system.out print(“a serial

break case commportidentifier port parallel
system.out,print(“,a parallel port)
break
default:
//important since other types a port like USB
//andfirewireare expected to be added in the future system, out print(“a port of unknown type)break
try

commport the port “com. Open(“port opener”10:
system.out print in”is not cruntly owned)
the portclose:catch(portin use exception
string owner=com,get currentowner if owner=null)owner=”unknown”
system.print in (“currently owned by +owner+)

In this example, COM2 is occupied though by a non-Java program that did not register its name.You also see that LPT2 is occupied ,which was some thing of a surprise to me-I did not thanks I was using any parallel ports.The second open ()method takes afile descriptor as an argument:
Public commport open (file Descripto)throws UnsupportedconnOperation exception

This may useful on opearting system like Unix ,where all devices,serial ports included are treated as files.On all other platforms,this methods throws att Unsupportedc ommOperation excemption
Public class Unsupportedcommoperation exception extends exception
There is no corresponding close()methods commportidentifier class.the necessary close()method is in cluded in the commpor class itsself.you should close all ports you”ve opened when you”re through with them.

Waiting for a port with port owner ship events
The commportidentifier class has two method that are used to received notification of changes in the owner of the ports ,These are a public void addportownership listener(comport owner ship listener port ownership events are fired to signal that a port has been opened aport has been closed. Or another application wants to take control of the port. To listen for owner ship changes on a particular port,you must register comport ownership

listener object with the commpor Identifier object representing the ports using the addport ownership listener ()method.
Public void addport ownership listener(comport ownership listener)
You can deregister the port ownership listener by passing it removeport ownership listerner():
Public void remove port ownership listener(commport ownership listener)
The Javax, comm,commport ownership listener is a subinterface of java. Util. Eventlitener that declares the single methoud ownership change():
Public abstract void ownership change (int type)
The commport ownership listener interface is unusual,unlike other events listener interfaces ,the listener methoiud is passed an int rather than and events this int will generally have one of three values that indicate particular changes in the ownership of the port .All three values are defined as mnemonic constants in javax,comm,commport ownership listener:
Commport ownership listener,PORT OWNED
Commport ownership listener PORT-UNOWNED
Commport ownership listener PORT-OWNERSHIP-REQUESTED
PORT OWNED means some application has taken ownership of the port, PORT-UNOWNED means some application has released ownership of the port.Finally, PORT-ONERSHIP-REQUESTED means some application has ,requested ownership of the port but does not yet have it ,because another application owns it .

If the owner of the port hears the events,.it can close the port to give it up to the requesting application example 17-5 is a program that watches for ports ownership change. It”s of limited use, since these events only appear to be fired when a Java program takes over or releases a ports not when other program do.
Example17-5port watcher
Import javax comm:
Public class port watcher implements commport ownership listener string ports name.
Public portwatcher(string port Name)throws Nosuchportexception this port name.
Commportibentifier.get port identifier(port name
Port identifier.addport ownership listener (this)
Public void ownership changes (int types)(switch types
Case comport ownership listener, ports OWNED system .out printin(portname*has become unavailable

Break case ownership listener,PORT-OWENED System.out printin port name*has become available case comport ownership listener.PORT – OWNERSHIP-REQUESTED system,out printin an application has requested ownership portname break defficult

System.out.println("Unknown port ownership event,type

12.6 Registering Ports

For comleteness, I"ll note the static commportIdeentifier, addportName method:
Public static void addportName (string portname, int porttype,commdriver)
This method registers a particular name,type and driver with the Comm API so that it cn be returned by commportIdentifier getportidentifiers () and similar menthods like the javax comm commdriver class that addport Name ()takes comm API,not for application programmers.

12.7 Communicating with a Device on a port

The open () methed of the commportIdentifier class returns a commport object. The javax .comm. ommportclass has methods for getting input

and output streams from a port and for closing the port .There are also a number of driver-dependent methods for adjusting the properties of the port.

12.8 Communicating with a port

There are five basic steps to communicating with a port:

1. Open the port using the open () method of commportIdentifier,if the portis available, this returns a commport object. Otherwise, a portInUseException is thrown.

2. Get the port"s output stream using the getoutoutstream () method of commport.

3. Get the port's input stream using the getInputstream () method of commpoirt.

4. Read and write data onto those streams as desired.

5. Close the port using the close () method of commport.

6. Through 4 are new However, they're not particulars complex. Onxce thehas been esablished,you simply use the normal methods of any input output stream to read and write data. The getInputstream () and output stream () methods of commport are similar to the name in the java,net.URL class. The primary difference is that with comm completeh responsible for understanding and handling the data to you .There are no content or protocol humdlers that perform any of the data.It devcic attached to the port requires a complicated protocol-for example, a fax modem- then you'll have to handle the protocol manually.

   Public abstract inputstream getInputstream throws ioException public abstract outputstream getoutputstream() throws 10 Exception

   Although these methods are declared abstract in comport, any instance of commport you retrieve from open () will naturally be a concrete subclass of comport in which these methods are implemented.

   Some ports are unidirectional,In other words, the port hardware only supports writing or reading, not both.Fol instance,early PC parallel ports only allowed the computer to sent data to the printer but could only send a small number of precisely defined signals back to the computer. This was fine for a printer but it meant that the parallel port wasn't useful for a device like aCD-ROM or a Zip drive. If the port you've opened doesn't allow writing, getoutputstream

   Returns null. If the port does't allow reading,getinputstream () returns null.

12.9 Port Properties.

The javax comm port class has a number of driver dependent methods for adjusting the properties of the port . These properties are most generic characteristics like

Size, that can be implemented software. More speceific properitcs of a particular type of port. Like the hund rate of a serial port or the mode of the parallel port,must be sent using a more specific subclass, like javax. Comm. Serial port or javax. Comm.

12.9.1 Parallelport.

The five generic properties are receive threshold, time out value, recive framing byte,input buffer size,and output buffer size.Four of these properties receive threshold,receive timeout,receive framing and input buffer size determine exactly howand when the input stream blocks .The receive threshold specifies the number bytes that must be available before a call to read () returns .The receive time –out specifies the number of milliseconds that must pass before a call to read() returns.The in put buffer size specifies how large a buffer will be provided for the serial port .If the buffer fills up these read () method return. For instance , if the receive threshold is set to 5,read() won't return until at least 5 bytes are available .If the receive time out is set to 10 milliseconds, read () will wait 10 milliseconds before returning however,if data becomes available before 10 milliseconds are up,read() return immediately .For example,if the receive threshold is set to 5 bytes and the receive time out is set to 10 milliseconds, then read () will wait until either 10 milliseconds pass or 5 bytes are available before returning.

If the in put buffer size is set and the receive threshold is set, the lower of the two values must be reached before read () will return Finally,if receive framing is enabled, all read return immediately , regardless of the other values. Table 17-1 summarizes.

Table 17-1,when does read

Receive	Receive		Input	
Receive Threshold	Time Out	Receive Framing	Buffer Size	read()Returns when
Disabled	disabled	disabled	B bytes	Returns when any data is available.

n bytes	tms	disabled	b bytes Returns after t milliseconds or when either n or b b bytes are avlable. Whichever is less.
Disabled	disabled	enabled	b bytes returns immediately
In bytes	disabled	enabled	b bytes Returns immediately
Disabled	tms	enabled	b bytes Returns immediately
In bytes	tms	Enabled	b bytes Returns immediately

The output buffer size is the number of bytes the driver can store for the output stream before it can write to the port. This is important,because it's cast for a fast

Program to write data faster than the port can send it out. Buffer overruns are a common problem,ewspecially on old PC's with slower serial ports.

Each of these properties has four methods one enables the property. One disables it

One checks whether the property is enabled,and one returns the current value. For instance,the receive threshold is adjusted by these four methods.

Public abstract void enable receive threshold (int size) throws Unsupported comm operationException

Public abstract void disable receive threshold()

Public abstract boolean is receive threshold enaboled()

Public abstract int get receive threshold()

The other three properties follow the same naming conventions. These four methods adjust the receive time out.

Public abstract void enable receive time out (int rcvtime out) throws unsupported comm operation exception

Public abstract void disable receive time out()

Public abstract boolean is receive time out inabled()

Public abstract int get receive time out()

These four methods adjust the receive framing property.

Public abstract void enable receive framing (int framing byte)

Throws Ynsupported comm operation exception

Public abstract void disable receive framing()

Public abstract boolean is receive framing enable()

Public abstract int get receive f raming byte()

These four methods adjust the input and output buffer size.

Public abstract void set input buffer size (int size)

Public abstract buffer size ()

Public abstract void set output buffer size (int size)

Public abstract int get output buffer size()

All drivers must support input and output buffer so there are no is input buffer enable() or disable output buffer () methods however other then the input and output buffer sizes drivers are not required to support these properties if a driver does not supports the given property then enabling it will throw an Unsupported comm operation exception you can

determine whether or not a driver supports a property by trying to enable it and seeing seeing whether or not an exception is thrown .example 17-7 uses this scheme to test the properties for the port of the host system .

Example: 17-7 port tester (continued)

```
Public class  porttester

Public static void main (string ) args

Enumeration the ports comm port identifiers get ports identifiers

While the ports has more elements()

Comm ports identifier com=(comm ports identifiers) the ports next element()

System . out print (com get name)

Switch(com.getportType)

Case Comm portIdentifier.PORT-SEAIAL:

System.out.printIn(a serial port)

Break;

Case CommporIdentifier.PORT-PARALLEL:


System.out.printIn( a parallel port:)

Break

Default:

//important since other types of ports like USB

//and firewire are expected to be added in the future

system.out.printin( a port of unknown type *)

break:
```

```
try
Comm port the port=Comm . open(port Tester”20)
test properties (theport)
the port close ()
catch (port  inure Exception e)
system .out. println (“pot in use ,can’t test properties”)
system .out. println ()
static void test reporters (Comm port  the port )
try
 the port enable receive threshold   (  10    ):
system .out .println (“Receive timeout not supported”)
```

Example 17-7.port Tester (continued)

```
Catch (Unsupported Comm operation Exception e)
System. out. println (receive framing not supported”)
Try
The port. Enable receive framing (10)
System .out .println  (“Receive threshold supported “)
Catch (unsupported Comm operation Exception e )
   System .out. println (Receive Threshold not supported)
```

Here’s the results for both serial and parallel ports from a Windows NT box running the Comm API 2.0:

D:\Java\16\Java port Tester

COMI, a serial port:

Receive threshold supported

Receive timeout supported

Receive framing supported

COM2, a serial port:

Port in use, can't test properties

LPTI, a serial port:

Receive threshold supported

Receive timeout supported

Receive framing supported

LPT2, a parallel port:

Port in use, can't test properties

12.9.2 SERIAL PORTS

The javax .Comm .serial port class is an abstract subclass of Comm port that provides various methods and constants useful for working with RS-232 serial ports and devices. The main purposes of the class are to allow the programmer to inspect, adjust and monitor changes in the settings of the serial port. Simple input and output is accomplished with the methods of the superclass, Comm port .Serial port has a public constructor, but that should not be used by applications , instead, you should call the open () method of Comm port identifier that maps to the port you want to communicate with then cast the result to Serial port ,For example

Comm port jden tjfjecpies Comm portifier get port identifier ("COM")

Chapter-13

13.1 The Java speech API.

.Java Speech API Specification—Defines the packages used to implement basic Java Speech capabilities, speech recognition and speech synthesis.

.Java Speech Programmer's Guide_Describe how to use the Java Speech API to develop speech-enabled application .

.Java Speech Grammar Format (JSGF) Specification-Describe thew JSGF and explains how it is used to create platform –independent speech

recognition grammars. These grammars identify the words that a user speaks and their meaning in particular program contexts.

.Java Speech Makeup Language (JSML)Specification—Describe the role of JSML and shows how it's used to mark text documents for use with speech synthesizers.
.Javax.speech.recognition-Provides classes and interfaces that support speech recognition.

.Java.speech.synthesis-Provides classes and interfaces that support speech synthesis.

The Java.speech packages consists of the following classes and interfaces:

.Central-class that provides central access (via static methods) to all capabilities of the speech API.

.Engine-interface that is implemented by speech recognition and synthesis engines.

.Engine Attributes-Defines the attributes that are supported by an engine object.
Engine Central-Provides the operating modes of a speech engine in terms of Enginmodedesc objects.

.Engine Mode Desc –Defines an engine operating mode

.Engine List- A collection of EngineModeDesc objects.

.Audio Manager – Interface that defines methods for controlling audio input and output and managing audio events.

.VocabManager - Interface. That defines methods for controlling words that are used by a speech engine.

.Word – Encapsulates speakable words.

.Speech Event – The super class of all speech events.

.Audio event – Subclass of speech event that is generated by speech engine objects based on audio input and output and output processing.

.Audio Listener – Defines methods for handling audio event objects.

.Audio Adapter – Implementation of the audio lister interface.

.Engine Event – Reports changes in speech engine status.

. Engine Listener – Defines methods for handling engine event objects.

. Engine Adapter – Implementation of the engine listener interface.

The following sections cover the Javax.speech.recignition and javax.speech

Synthesis packages.

13.2 CLASS HIERARCHY
13.2.1 CLIENT LISTENER SERVER SIDE
13.2.1.1 CLASS HIERARCHY FOR CLIENT LISTERER CLASSES

Class java. Lang. Object

Class java . awt. Component (Implements
Java. Awt.
Image.ImageObserver,java.awt.MenuContainer,
Java.io.serializable)
Class java.awt.Container
Class java.awt.Window (implements
Javax.accessibility.Accessible)

Class java.awt.Frame(implements
Java.awt.MenuContainer)
Class Client Listener

Class DB Manager

13.2.1.2 LIST OF ALL MEMBER FUNCTIONS

Check IP (String)- Method in class DBManager

This method is used chech the duplicate IP address in the database.

checkTelNo(String) – Method in class DBManager

This method is used chech the duplicate telephone number in the database.

Client Listener – Class clientListener.

This class is used to make the threads for the clients

ClientListener() – Constructor for class ClientListener

Constructor only sets the GUI and Action Listeners.

Con - Variable in class DB Manager

DBManager –class DBManager.

This clss is handling all database tansactions.

DBManager() – Constructor for class DBManager

Constructor onlynloads the driver necessary for the data base connections and then establishes the connection with the database.

deleteTelTuple(String) – Methods in class DBManager

This function is used to delete the IP address from the data base.

Display (String) – Method in class Client Listener

This function is only used to display the Stings on the GUI.

insertRequest(String,String)- Method in class DBManager

This function is called for any request of log on from the Internet user .

insertSequence(Strings,String, String) –Method in class DBManager

This function is called for any request of log on from the Interner user.

Main (String II) – Static method in class Client Listener

This function is only used to start the application

runServer() – Method in class ClientListener

It receives the requests of the clients and call the Server Thread class to handle it

13.2.1.3 CLASS CLIENTLISTERNER

Java.lang.object
Java.awt.component
Java.awt.container.
Java.awt.window

Java.awt.frame.
Client Listener

All Implemented Interfaces:

Javax.aaccessibility.Accessible,java.awt.image.imageObserver,
Java. Awt.Menucontainer,java.io.seriallizable

Public class ClientListener
Extends java.awt.frame
This class is used to make the threads for the clients. Clients of this hclass lphone Listener class and I User class

Version
1.0,9-Oct-2001
Author:
Ksum
See Also:
I User ,server tread, Seriallized Form

Inner classes inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Inner classes inherited from class java.awt.window
java.awt.Window.AccessibleAWTWindow

Inner classes inherited from class java.awt.container
java.awt.container.AccessinbleAWTContainer

Inner classes inherited from class java.awt.component
java.awt.Component.AccessibleAWTComponent

Fields inherited from class java.awt.frame.
CROSSHAIR-CURSOR,DEFAULT-CURSOR,E-RESIZE-CURSOR,HAND-CURSOR,INCONIFIED,MOVE-CURSOR,N-RESIZE-CURSE,NE-RESIZE-CURSOR,NORMAL,NW-RESIZE-CURSOR,S-RESIZE-CURSOR,
SE-RESIZE-CURSOR,SW-RESIZE-CURSOR,TEXT-CURSO,
W-RESIZE-CURSOR,WAIT-CURSOR

Fields inherited from class java.awt.component
BOTTOM-ALIGNMENT, CENTER-ALIGNMENT,
LEFT-ALIGNMENT,RIGHT-ALIGNMENT,TOP-ALIGNMENT

Fields inherited from class java.awt.image.ImageObserver
ABORT,ALLLBITS,ERROR,FRAMEBITS,HEIGHT,PROPERTIES,SOM
EBITS,
WIDTH

Constructor Summary
ClientListener()
Constructor only sets the GUI and ActionListeners.
Method Summary
Void display(java.lang.String s)
This function is only used todisplay the strrings on the
GUI.
Static voi main (java.lang.sstring[] arg)
D This function is only used to start the application
Void run Server ()
It receives the requests of the clients and claa the
ServerThread class to handle it.

Method Inherited from class java.awt.frame
Add Notify , finalize, getAccessibleContext, getCursorType,
getFrames, getlconlmage, getmenuBar, getstate, gettitle,
is resizable, paramstring, remove, removeNotify, sectcurso,
setlconlmage, setMenuBar, setResizable, setState , settitle

Method Inherited from class java.awt.Window
addWindowListener,alllyResourceBundle,
applyResourceBoundle, dispose, gerFocusOwner,
getGraphicsConfiguration, getInputcontext,getListeners,
getLocale, getOwnedwindows,getOwner,gettoolkit,
getWarningstring, hede, isShowing, pack, postenent,
processEnent, processWindowevent, removeWindowListener,
setCursor, show, toBach, toFront

Method Inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addlmpl,
countcomponents, deliverEnent, doLayout, findComponentat,
findComponentAt, getAlignmentX, getAlignmenty,
getcomponent, getComponent, getComponentAt, getcomponentsAt,
getComponentCount, getcomponents, getlnsets, getLayout,

getMaximumsize, getMinimumsize, getPreferredSize, insets,
invalidate, isancestorOf, layout, list, list, locate, minimumSize,
lpaint, paintcomponents, preferredSize, print, printComponents,
processcontainerevent, remove, remove, removeAll,
removecontainerListener, setfont , setLayout, update, validate,
validatetree

Method Inherited from class java.awt.Component
action, add, addcomponentListener, addfocusListener,
addHierarchyBoundsListener, addHierarchyListener,
addInputMethodListener, addKeyListener, addMouseListener,
addMouseMotionListener, addPropertychangeListene
addLPropertyChangeListener, bounds, checklmage, checkImage
coalesceevents, contains, contains, createImage, createImage,
disable, disableEvents, dispatchevent, enable, enable,
enableevents, enableinput methods, firepropertychange,
getbackground, getbounds, getboundsgetcolormodel,
getcomponentorientation, getcursor, getdroptarget, getfont,
getfontmetrics, getforeground, getgraphics, getheight,
getInputmethodrequests, getlocation, getlocatio,
getlocationonscreen, getname, getparent, getpeer, gertsize
getsize, gettreelock, getwidth, getx, gety, getfocus,
handleevent, hasfocus, imageupdate, inside, isdisplayable,

isdoublebuffered, isenabled, isfocustraversable, islightweight
isopaque, isvalid, isvisible , keydown, ksyup, list, list, list,
location, lostfocus, mousedown, mousedrag, mouseenter,
mouseexit, mousemove, mouseup, move, nextfocus, paintall,
prepareimage, prepareimage, printall, processcomponentevent,
processfocusevent, processhierarchyboundsevent,
processhierarchyevent, processinputmdethidevent,
processkeyevent, prorcessmousevent,
processmousemotionevent, removecomponentlistener,
removefocuslistener, removehierarchyboundslisterner,

removehierarchylistener, removeinputmethodlistener,
removekeylistener, removemouselistener,
removemousemotionlistener, removepropertychangelistener,
removepropertychangelistner , repaint, repaint, repaint, repaint,
requestfocus, reshape, resize, resize, setbachground,
setbounds, setbounds, setcomponentorientation,
setdroptarget, setenabled, setforeground, setlocate,
setlocation, setlocation, setname, setsize, setsize, setvisible,
show, size, tostring, transferfocus.

Method Inherited from class java.lang.object.
Clone, equals, getclass, hashcode, notify , notifyall, wait, wait, wait.

Method Inherited from class java.awt.MenuContainer
Getfront, postevent

Constructor Detail

ClientListener

Public ClientListener ()

Constuctor only sets the GUI and action Listeners.

Method Detail

Runserver

Public void runServer()

It receives the requests of the clients and call the serverThread class To handleit . it's clients are phoneListener class and IUser class.

See also:
Serverthread, IUser, ModemManager

Display

Public void display(java.lang.string s)

This function is only used to display the strings on the GUI.

Parameters:

S- string to be displayed

Main

Public static void main (java.lang. string [] arg)

This function is only used to start the application.

Parameters:

Arg[] – array of string from the command line , not applicable here.

13.2.1.4 CLASS DBMANAGER

Java. Lang. Object
+… DBManager

public class DBManager
extends java. Lang. Object
This class is handling all database transactions.
Version:
1.0, 9-Oct-2001
Author:
KUSM
See Also:
ServerThread

Field Summary
java. Sql. Connection con

Constructor Summary
DBManager ()
Constructor only loads the driver necessary for the data base Connections and then establishes the connection with the database.

Method Summary
In chechIP (java. Lang. string chech IP)
T This methid is used chech the duplicate IP address in the database.

In chechTelNo (java. Lang. string chech Tel)

T This method is used chech the duplicate telephone number in the ddatabase.

In deleteTelTuple (java. Lang. String anyTel)
T This function is used to delete the telephone number from the data base.

In deldeteTuple(java. Lang. string anyIP)

T This function is used to deldete the IP address from the data base.

In insertRequest(java. Lang. string name. java. Lang. string tel No,
T java.lang. String ipAdd)
This function is called for any request of log on from the Internet user.
In insert Sequence (java. Lang. string name, java. Lang . string telNo,
T java. Lang. String ipAdd)
This function is called for any request of log on from the Internet user.

Methods Inherited from class java.lang.object
Clone, equals, finalize, getcalss, hashcode, notify, notifyall,
Tostring, wait, wait, wait.

Field Detail

Con

P[ublic java. Sql. Connection con

Constructor detail

DBManager

Public DBManager ()
Constructor only loads the driver necessary for the data base
Connections and then establishes the connection with the database.

Method Detail

insertSequence

lpublic int insertSequence(java.lang.String name,

java, lang. String telNo,
java. Lang. String ipAdd)
This function is called for any request of log on from the Internet user.
First of all it is cheched that this ip or tel no is present in the data base
Or not if yes then they are deleted and new information inserted otherwiseinformationis inserted as it is

Parameters:
Name- name of the internet user

Tel No- telephone number of the internet user
ipAdd – IP address of the internet user

Returns:
Status of the request, 1 - > successful, 0 -> not successful
See Also:
ServerThread, checkup(java. Lang. String), #ChechtelNo,
Deletetuple(java. Lang. string).
Deletetel Tuple(java. Lang. String).
I nsert Request (java. Lang. String. Java. Lang. String. Java. Lang. Sting)

Delete Tuple

Public int deleteTuple(java. Lang. String anyIP)

This function is hused to delete the IP address from the data base. It may be in the case that user wants to log off or somehow he was not able to log off and new request for log on is of the same ip then previous entry will be deleted from the data base using this function.
Parameters:
anyIP – ip address to be deleted
Returns:
Result of the query, 1 -> success, 0-> success
See Also:
insertSequence(jaba. Lang. String, java. Lang. String, java. Lang. string)
serverthread

delete TelTuple

public int deleteTelTuple(java. Lang. string anytel)

This function is used to delete the telephone numner from the database. It may be in the case that user was not abl to log off and new

request for log on is of the same number , then previous entry will be deleted from the data base using this function.
Parameters:
Any tel- Telephone number to be deleted
Returns:
Result of the query, -> success, 0 -> no success
See Also:
insertSequence(java. Lang. String. Java. Lang. String, java. Lang. String.

Check TelNo

Public int chechTelNo (java. Lang. StringchechTel)

This method is used chech the duplicate telephone number in the database. It takes the telephone number to be cheched. End return the result.
Parameters:

chechTel – telephone number to be cheched
Returns:
Return the result of the query 1 -> success, 0 -> no success

See Also:
Insertrsequence(java. Lang. String, java. lang . String, java. lang. string)

chekIP

public int chechIP (java. lang. String chechIP)

This method is used chech the duplicate IP address in the database. It takes the IP address to be checked and return the result.
Parameters:
Chech IP – IP address to be checked
Returns:
Return the result of the query 1 -> success, 0 -> no success
See Also:
insertSequence(java. lang. String, java. lang. String, java. lang. String)

13.2.2 IUSER SIDE

13.2.2.1 CLASS HIERARCHY FOR IUSER CLASSES

Class java. lang. Object

Class java. at.Component (implements
Jaba. Awt. Image. imageObserver, java, awt . MenuContainer,
Java.io.Serializable)

Class java. awt.Container
Class java.awt.Window(implements
Javax. Accessibility. Accessible)
Class java. awt. Frame (implements
Java. awt. Menucontainer)

Class javax.swing.jFrame(implements
Javax. Accessibility. Accessible,
Javax. Swing.RoorPaneContainer,
Javax.swing. Windowconstants)

. class IUser

13.2.2.2 LIST OF ALL MEMBER FUNCTIONS

C

connectCL() – Method in class IUser
This function is used for logging on.

D

disConnectCL() – Method in class IUser
This function is used for logging off.
Display(String) – Method in class IUser
This function displays the data in the text area to the user.
IUser – class IUser
This class builds the GUI for the user and does the remaining all tasks.
IUser() – Constructor for class IUser
This constructor is used to build the GUI.

M

Main(StringII)- Static method in class IUser
This function is used to make the object of the class.

S

sendData(String) – Method in class IUser
This function flusher the data in the output stream.

W

Wait For Call() – Method in class IUser
This function initiate the session with the server.

13.2.2.3 CLASS IUSER

Java. lang. Object

+-- java.awt.Component

+-- java. awt. Container

+-- java. awt. Window

+-- java. awt. Frame

+-- javax. Swing. JFrame

+--IUSer

All Implemented Interfaces:
Javax. Accessibility .Accessible, java.awt.image.imageObserver,
Java.awt.MenuContainer,javax.swing.RootPanecontainer,

java.io.serializable, javax. Swing. Windowconstants
public class IUser
extemds kavax.swing.jframe
This class builds the GUI for thruser and does the remaining at tasks.
Version:
1.0, 9-Oct-2001
Author:
Ksum
See also:
IUManager, serverthread,SerializedForm

Inner classes Inherited from class javax.swing.Jframe.
Javax. Swing. Jframe. AccessibleJFrame

Inner classes Inherited from class java.awt.Frame

Java. awt. Frame. AccessibleAWTFrame

Inner classes Inherited from class java.awt.Window

Java. awt. windowAccesibleAWTWindow

Inner classes Inherited from class java.awt.Container

Java. awt. Container. AccessibleAWTContainer

Inner classes Inherited from class java.awt.Component

Java. awt. Component. AccessibleAWTComponent

Fields inherited from class javax.swing.Jframe.
accessibleContext, EXIT-ON-CLOSE,rootPane,
rootPanechechingEnabled
Fields inherited from class java.awt.Frame

CROSSHAIR-CURSOR,DEFAULT-CURSOR,E-RESIZE-CURSOR,
HAND-CURSOR,ICONIFIED , MOVE-CURSOR,
N-RESIZE-CURSOR, NE-RESIZE- CURSOR, NORMAL
NW-RESIZE-CURSOR,S-RESIZE-CURSOR,
SE-RESIZE-CURSOR, SW-RESSSIZE-CURSOR, TEXT-CURSOR,
W-RESIZE-CURSOR, WAIT- CURSOR
Fields inherited from class java.awt.Component

BOTTOM- ALIGNMENT, CENTER- ALIGNMENT,
LEFT-ALIGN MENT, RIGHT-ALIGNMENT, TOP- ALIGNMENT
Fields inherited from class javax.swing.Windowconstants

DISPOSE- ON- CLOSE, DO-NOTHING-ON –CLOSE,
HIDE-ON-CLOSE
Fields inherited from class java.awt.image.Imageobserver

ABORT, ALLBITS, ERROR, FROMEBITS, HEIGHT, PROPERTIES,
SOMEBITS, WIDTH

Constructor Summary
IUser()
This constructor is used to build the GUI
Method Summary
Void connectCL()

This function is used for logging on.
Void disconnect CL()

This function is used for logging off.

Void display(java. lang. stringshow)

This function display the data in the text area to the user.

Static void main(java. lang. String[] arg)

This function is used to make the object of the class

Void sentData(java. lang. string data)

This function flushes the data in the output stream.

Void waitFroCall ()

This function initiate the session with the server.

Methods inherited from class javax.swing.Jframe

Addlmpl, creatrerootpane, framelnit, getaccessiblecontext,
Getcontentpane, getdefaultcloseoperation, getglasspane,
Getjmenubar, getlayeredpane, getrootpane,
Is rootpanechechingenabled, paramstring, processkeyevent,
Processwindowevent, remove, setcontentpane,
Setdefaultcloseoperation, setglasspane, setjmenubar,
Setlayeredpane, setlayout, setrootpane,
Setrootpanechechingenabled, update

Methods inherited from class java.awt.Frame

Addnotify, finalize, getcursortype, getframes, getlconlmage,
Getmenubar, getstate, gettitle, isresizable , remove,
Removenotify, setcursor, seticonimage, setmenubar,
Setresizable, setstate, settitle

Methods inherited from class java.awt.Window

Addwindowlistner, applyresourcebundle,

Allpyresourcebundle, dispose, getfocusowner ,
Getgraphics configuration. Getinputcontext, getlisteners,
Getlocale, getownedwindows, getowner, gettoolkit,
Getwarningstring, hide, isshowing, pack, postevent,
Processevent, removewindowlistener, setcursor, show, tobach,

Methods inherited from class java.awt.Container

Add, add, add, add, add, addcontainerlistener,

Countcomponents, deliverevent, dolayout, findcompomentat,
Findfompomentat, getalignmentx, getalignmenty,
Getcompoment, getcomponentat, getcompomentat,
Getcomponentcount, getcomponents, getlnstes, getlayout,
Getmaxmumsize, getmimnimumsize, getpreferredsize, insets,
Invalidate, isancestorof, layout, list, list, locate, minimumsize,
Paint, paintcomponents, preferredsize, print, printcompomponents,
Processcontainereven, remove, removeall,
removeckontainerlistner , sertfont, validate, validatetree

Methods inherited from class java.awt.Component

action, add, addcomponentlistener, addfocuslistener,
addhirrarchyboundslistener, addhierarchylistener,
addinputmethodlistener, addkeylistener, addmouselistener,
addmousemotionlistener, addpropertychangelistener,
addpropertychangelistener, bounds, chechimage, checkmage,
coalesceevents, contains, contains, createimage, createimage,
disable, disableevents, dispatchevent, enable, enable,
enableevents, enableinputmethods, fireporpertychange,
getbackground, getbounds, getbounds, getcolormodel,
getcomponentorientation, getcursor, getdroptarget, getfont,
getfontmetrics, getforeground, getgraphics, getheight,
getinputmethodrequests, getlocation, getlocation,
getlocationonscreen, getname, getparent, getpeer, getsize,
getsize, gettreelock, getwidth, getx, gety, gotfocus,
handleevent, hasfocus, imageupdate, inside, isdisplayable,
isdoublebuffered, isenabled, isfocustraversable, islightweight,
ispaque, isvalid, isvisible, keydown, keyup, list, list, list,
location, lostfocus, mousedown, mousedrag, mouseenter,

mouseexit, mousemove, mouseup, move, nextfocus, paintall,
lprepareimage, prepareimage, printall, processcoponentevent,
processfocusevent, processhierarchyboundsevent,
processhierarchyevent, processinputmethodevent,
processmouseevent, processmousemotionevent,
removecompomentlistener, removefocuslistener
removehierarchyboundslistener, removehierarchylistener,
removeinpitmethodlostener, removekeylistener,
removemouselistener, removemousemotionlistener,

removepropertychangelistener, removepropertychangelistener,
repaint, repaint, repaint, repaint, requestfocus, reshape, resize,
resize, setbachground, setbounds, setbounds,
setcomponentorientation, setdroptarget, setenabled,
setforeground, setlocale, setlocation, setlocation, setname,
setsize, setsize, setvisible, show, size, tostring, transferfocus
Methods inherited from class java.lang.object

clone, equals, getclass, hashcode, notify, notifyall, wait, wait, wait,
Methods inherited from class java.awt.MenuContainer

getfont, postevent
Constructor Detail

IUser

Public IUser ()
This constructor is used to build the GUI. Italso sets the action Listeners for the LOG OFF and LOG ON butons.

Method detail

Send Data

public void senddata(java. lang. stringdata)
This function flusher the data in the output stream.
Parameters:
Data – String to be sent

Display

Public void display(java. lang. String show)
This function displays the data in the text area to the user.
Parameters:
Show – String to be shown

connectCL

public void connectCL()
This function is used for logging on . It sends request to the server Thread class for logging on.
See Also:
ServerThread

disConnectCL

public void disconnectCL()

This function is used for logging off. It sends request to the server thread class for logging off.

See Also:

Server Thread

waitForCall

public void waitForCall ()

This function initiate the session with the server. It creates the Server socket and listen for the response from the phoneListener. Then it makes the object of test class to initiate voice conversation.

See Also:

Test, SevrverThread

Main

Public static void main(java. lang. String[] arg)

This function is used to make the object of the class.

Parameters:

Arg[] – any command line argument,null in this case

13.2.3 MODEM MANAGER SERVER SIDE

13.2.3.1 CLASS HIERARCHY FOR MODEM MANAGER SERVER

class java. lang. Object

class CLManager

class IUManager

class ModemManager

13.2.3.2 LIST OF ALL MEMBER FUNCTIONS

C

CallCentre() – Method in class ModemManager

This function is the main function of the class which performs all the tasks.

CLManager – class CLManager.

This class is used for interaction with the Data Base.

CLManager() – Constructor for class CLManager

It opens the connection with the ServerThread class to get the IP of the user,

Command(String) – Method in class ModemManager

It is used to send the commands to modem through stream.

G

getTelNoIP(String) – Method in class CLManager

It passes the telephone number to the data base to get the IP of the Internet user from there.

giveNumber() – Method in class modemManager

It is used to extract the telephone no.

I

InitializeModem(CommPortI detifier, String) – Method in class ModemManager

This method is used to initialized the modem.

isUserLogon(String) – Method in class IUManager

This is the main function of the class. It checks whether the user is offline or kon line and tells him/her that there is a call for this lphone number.

IUManager – class IUManager

This class is used for interaction with the Internet User.

IUManager() – Constructor for class IUManager.

L

LookIn(char) – Method in class ModemManager

It is used to extract the only the telephone no.

M

Main(StringII) – Static method in class ModemManager

It is the main function to run the application

ModemManager – class ModemManager

This class is used to interact with the modem.

ModemManager () – Constrctor for class ModemManager

This constructor makes the object of the classes which are needed during the execution.

N

NumberBolo(char) – Method in class ModemManager

It is used to tell the dialed telephone no.

numberNikalo() – Method in class ModemManager

It is used to senf number Nikalo numbers to say it aloud to user.

P

Print(byteII) – Method in class ModemManager

It is used to print responses of the modem.

R

Response() – Method in class ModemManager

It is used to take the response from the modem through stream

T

telNoExtractor() – Method in class ModemManager

it is used to take the telephone number from user.

V

ValidateNo() – Method in clss ModemManager

It is used to validate the number which is entered by the user.

13.2.3.3 CLASS MODEMMANAGER

Java. lang. Object

+-- ModemManager

public class ModemManager
extends java. lang. Object

This class is used to interact with the modem. This class also uses objects of IUManager, CLManager and Message Player to accomplish the task.
Version:
1.0, 9-Oct-2001
Author:
Ksum

See Also:
IUManager, CLManager, MessagePlayer

Constructor summary.
Modem Manager()
This constructor makes the object of the classes which are needed during the execution.

Method Summary
Void callCenter ()
This function is the mauan function of the class which performs all the tasks
Void command (java. lang. String atComm)
It is used to send the commands to moder through stream.
Void giveNmber()
It is is used to extract the tepephone no.
Void initializeModem(CommPorldetifier portld,
Java, lang. String. COMport)
This method is used to initialized the modem.
Void lookIn (char c)
It is used to extract the only the telephone no.
Static void main(java.lang.String[] args)
It is the main function to run the application.

Void numberBolo(char awaz)
It is nsed to tell the dialed telephone no,
Void number Nikalo()
It is used to send number Nikalo numbers to say it aloud to user.
Void print(byte[] received)
It is used to print responses of the modem.
Byte[] reponse()
It is used to take the response from the modem through stream.
Void telNoExtractor

It is used to take the telephone number from user.
Void validateNo()

It is used to validate the number which is entered hby the user.

Methods inherited from class java.lang.object

Clone, equals, finalize, getclass, hashcode, notify, notifyall, tostring, wait, wait, wait

Constructor Detail

ModemManager

Public modemManager()

This constructor makes the object of the classes which are needed during the execution, secondly it also locates the ports available to find the modem.

Method Detail

initializeModem

public void initializeModem(commPortIdentifier lportId,java. lang. string COMport)

This method is used to initialized the modem. It first creates the streams with the modem and then give it AT commands to bring jin it online voice command mode to receive the telephone calls

parameters:
Porrld – id of the required port
COMport – port at which modem is connected

Command

Public void command (java. lang. String atComm)

It is used to send the commands to modem through stream

Parameters:
atComm – AT command to be send
See Also:
#initializedModem

response

public byte [] response ()

It is used to take the response from the modem through stream. It takes response in the form of Byte array.
Returns:
Response in the form of Bytrearray.
See Also:
Command(java. lang. String)

callCenter

public void callCenter()

This function is the main function of the class which performs all the tasks. It receives the telephone call, give responses to the user, call other objects to connect to database and internet user etc. Finally it makes the object of test class to start conversation.
See Also:
IUManager, CLManager, MEssagePLayer, Test, numberNikalo(), telNoExtractor(), #telNoBolo, validateNo()

print

public void print(byte [] received

It is used to print responses of the modem.
Parameters:
Received – Bytes to be printed.
See Also:
Response()

telNoExtractor

public void telNoExtractor()

It is used to take the telephone number from user.
See Also:
Give number ()

lookin

public void lookin (char c)

It is used to extract the only the telephone no. from the response of user.
See Also:
telNoExtractor ()

giveNumber

public void give Number ()

It is used to extract the telephone no. from the response of it.
See Also:
lookIn(char)

numberNikalo

public void number Nikalo()

It is used to send number Nikalo numbers to say it aloud to user .
See Also:
Number Bolo(char)

Number Bolo

public void numberBolo(char awaz)

It is used to tell the dialed telephone no. to user.
See Also:
Number Nikalo()

ValidateNumber

Public void validateNo()

It is used to validate the number which is entered by the user,
See Also:
Callcenter()

Main

Public static void main(java. lang. String. String[] args)

It is the main function to run the application.
See Also:
ModemManager()

13.2.3.4 CLASS IUMANAGER

Java . lang. Object

+--IUManager

public class IUManager

extends java. lang. Object

This class is usd for interaction with the internert user. This class checks whether IUser is online or offline.

Version:

1.0, 9-Oct-2001

Author:

Ksum

See Also:

ModemManager, IUser, CLManager

Constructor Summary

IUManager()

Method Summary

In isUserLogon(java. lang. String IP)

T This is the main function of the class .It chechs whether the user is offline or online and tells him/her that thee is a call for this phone number.

Methods inherited from class java.lang.object

Clone, equals, finalize, getclass, hashcode, notify, notifyall. Tostring. Wait, wait, wait

Constructor detail

IUManager

Public IUManager()

Method detail

isUserlogon

public int isUserLogon(java. lang. String IP)

This is the main function of the class. It checks whether the user is offline or online and tells him/her that there is a call for this phone number . It takes the IP of the Internet user from the CIManager and user it for checking purposes.

Parameters:

IP – IP of the Internet user

See Also:
CLManager, ModemManager

13.2.3.5 CLASS CLMANAGER

Java. lang. Object

+--CLManager

public calss CLManager
extends java. lang. Object
This class is used for interaction with the data base. This class is used to send the pelephone number to the database to get the ip address of the respective number from there.
Version:
1.0 9-Oct-2001

Author:
Ksum
See Also:
CleintLisstener, IUManager, ModemManager

Constructor Summary
CLManager()
It opens the connection with the ServerThread class to get the IP of the user.
Method Summary
In getTelNoIP(java. lang. String tel)
T It passes the telephone number to the data base to get the IP of the Internet user from there.

Methods inherited from class java.lang.object

Clone, equals, finalize, getclass, hashcode, notify, notifyall, toString, wait, wait, wait,

Constructor Detail

CLManager

Public CLManager()

It opens the connection with the Server Thread class to get the IP of the user.

See Also:
ServerThread

Method Detail

getTelNoIP

public int getTelNoIP (java. lang. String tel)

It passes the telephone number to the data base to get the IP of the internet user from there.

Parameters:
Tel – telephone number of the called party.
See Also:
ServerThread

13.2.4 MESSAGE PLAYER SIDE

13.2.4.1 CLASS HIERARCHY FOR MESSAGE PLAYER CLASSES

Class java. lang. Obdject

Class java. util. Dictionary

Class java. util. Hashtable (implements

Java . lang. cloneable, java. util. Map, java. io. Serializable)

Class SoundList

class MessagePlayer

class soundLoader

13.2.4.2 LIST OF ALL MEMBRS FUNCTIONS

getClip(String) – Method in class SoundList
This function gets the audio clip to be played.

M

MessagePlayer – class MessagePlayer.
This class is used to play the messages to the phone user.
MessagePlayer () - Constructor for class MessagePlayer

P

Putclip(AudioClip, String) – Method in class SoundList
This function puts the audio clp to be played.

S

SoundList - class SoundList.
This class loads and holds a bunch of audio files whose locations are specified.
SoundList (URL) - Constructor for class soundList
This constructor constructs the url and inturn the list of the files to be loaded.
SoundLoader – class soundLoader
This class is used to load the files of the mesages to be played for the phone user.
SoundLoader(SoundList, URL, Sting) – Constrcuctor for class soundLoader
This constructor does all the job of loading the sounds.
Start(string, int)- Method in class MessagePlayer
This function is used to srtart the clip once loaded.
startLoading(String) – Method in class soundList
This function start loading the files basing on the relative url by passing the information to Sound Loader class.
Stop() - Method in class MessafgePlayer
This function is used stop the clip.

13.2.4.3 CLASS MESSAGEPLAYER

Java. lang. Object

+--MessagePlayer

public class MessagePlayer
extends java. lang; Object

This class is used to play the messafes to the phone user. This class loads all the messafes before hand and palys them for the user on demand.
Version:
1.0, 9-Oct-2001
Author:
Ksum
SeeAlso:
ModemManager, soundList, soundLoader
Constructor Summary
MessagePlayer()
Method Summary
Void start(java. lang. String chosenFile. Int time)
This function is used to strart the clip once loaded.
Void stop()
This function is used stop the clip.
Methods inherited from class java.awt.lang.object

Clone, equals, finalize, getclass, hashcode. Notify, notifyall,
Tostring, wait, wait, wait
Constructor Detail

MessagePlayer

Public MessagePlayer()

Method Detail

Stop

Public void stop()
This function is used stop the clip.

Start

Public void start(java. lang. stringchosenFile,
Int time)
This function is used to start the clip once loaded.
Parameters:
chosenFile – name of the file to be started
time – interval for which execution will not proceed so that message can be played

13.2.4.4 CLASS SOUNDLIST

Java. lang. Object
Java. util. Dictionary
Java. util. Hashtable
+-- SoundList
All Implemented Interfaces:

java. lang. cloneabel, java. util. Map. Java. io. Serializable

lpublic class SoundList
extends java. util. Hashtable
this class loads and holds a bunch of audio files whose locations are specified. Relative to a fixed base URL.
Version:
1.0, 9-Oct-2001
Author:
Ksum
SeeAlso:
MessagePlayer, SoundLoader, Serialized form
Inner classes inherited from class java.util.map
Java. util. Map. Entry
Constructor Summary
SoundList(URL baseURL
This constructor constructs the url and inturn the list of the files to be loaded.
Method Summary
AudioCli getclip(java. lang. String relativeURL)
P This function gets the audio clip to be played.
Void putClip(Audioclip clip, java . lang. String relativeURL)
This function puts the audio clip to be played .

Void startLoading the files basing on the relativeurl by passing the information to soundLoader class.
Methods inherited from class java.util.Hashtable

Clear, clone, contains, containskey, containsvalue, elements, Entryset., equals, get, hashcode, isempty, keys, keyset, put, putAll Rehash, remove, size, tostring, values
Methods inherited from class java.lang.object

Finalize, getClass, notify, notfyAll , wait, wait, wait

Constructor Detail

SoundList

Public SoundList(URL baseURL)

This constructor constructs the hurl and inturn the list of the files to be loaded. For this process it used SoundLoaderclass

Parameters:

baseURL – base url of the file

See Also:

SoundLoader

Method detail

startLoading

public void start Loading (java. lang. String relative URL)

This function start loading the ffiles basing on the relative url by passing the information to SoundLoader class.

Parameters:

Relative URL – relative url of the file

Relative URL – relative url of the file to be loaded

See Also:

SoundLoader

getClip

public Audioclip getClip(java. lang. Stringrelative URL)

This function gets the audio clip to be played.

Parameters:

relativeURL – relative url of the file

Returns:

Audio clip of the relative URL.

putClip

public void putClip (Audioclip clip,

java. lang. String relative URL)

This function puts the audio clip to be palyed.

Parameters:

Relative URL – relative url of the file
Clip – clip to be put

13.2.4.5 CLASS SOUNDLOADER

Java. lang. Object
+--SoundLoader
public class SoundLoader
extends java. lang. Object
This class is used to load the files of the messages to be played for the phone user . This class helps in loading all the messages before hand and palyes them for the huser on demand.
Version:
1.0, 9-Oct-2001
Author:
Ksum
See Also:
ModemManager, SoundList, MassagePlayer
Constructor Summary
Sound Loader(SoundList soundList, URL base URL,
Java. lang. String relativeURL)

This constructor does all the job of loading the sounds.
Methods inherited from class java.lang.object

Clone, equals, finalize, getclass, hashcode, notify, notifyall, toString, wait, wait, wait

Constructor Detail SoundLoader
Public SoundLoader(SoundList soundList,
URL baseURL
Java. lang. string relativeURL)
This constructor does all the job of loading the sounds.
Parameters:
soundList – list of files to be loaded
baseURL – base URL of the file
relative URL – relative URL of the file

13.2.5 VOICE TRANSMISSION SIDE

13.2.5.1 CLASS HIERARCHY FOR VOICE TRANSMISSION

Class VTServer
Class VTClient
Class java. lang. Object
    Class test
Class MY controllerListener

13.2.5.2 LIST OF ALL MEMBER FUNCTIONS

C

Close() – Method in class MycontrollerListener
    This function is used to close the palyer.
Configure(int) – Method in class MyControlledListener
    This function is used to configure the palyer
controllerUpdate(ControllerEvent) – method in classMyControllerListener
    This function is used for event handling.
createManager(String, int, int, Boolean, Boolean) – Method in class VTClient
    This function is used to cerate the session with the Server

D

Devices actionPerformed(String) – Method in class VTServer

    This function is used to cerate the session with the client.

M

MycontrollerListener – class MyControllerListener.
    This class is used for transition of the player from onde state to other.

P

playToEndOfMedia(int) – Method in class MyControllerListener
    This function is used to detect end of media stream for the palyer.

Prefetch(int) – Method in class My ControllerListener
    This function is used for prefetching.

R

Realize(int) – Method in class MyControllerListener
    This function is used to realize the palyer.

T

Test – classTest.
This class use makes the objects of VTServerand VTClient
Test(String) – Constructor for class Test
It takes the IP address of the client.

U

Update(ReceiveStreamEvent) – Method in class VTClient

Update(RemoteEvent) – Method in class VTClient
This function is used to handle RemoteEvent.
Update(SendStreamEvent) – Method in class VTServer
This function is used to handle SendStreamEvent.

V

VTClient – class VTClient
This class initiate the Audio Session with the Server.
VTClient() – Constructor for class VTClient

VTServer – class VTServer
This class initiate the Audiosession with the client.
VTServer() – Constructor for class VTServer

13.2.5.3 CLASS TEST

Java. lang. Object

+--Test

public class test
extends java. lang. Object
This class use makes the objects of VTServer and VTClent
Version:
1.0, 9-Oct-2001
Author:
Ksum
See Also:
VTServer, VTClient
Constructor Summary

Test(java. lang. String ip)

It takes the IP address of the client.

Methods inherited from class java.lang.object

Clone, equals, finalize, getclass, hashcode, notify, notifyall, toString, wait, wait, wait

Constructor Detail

Test

Public Test(java.lang . String ip)

It takes the IP address of the client it passes the IP address to the VTServer method and to VTClient method createManager.

See Also:

createManager

13.2.5.4 CLASS VTSERVER

Constructor Summary

VTServer

Method summary

Public class VTServer

This class intiate the audio session with the client. It sends request to VTCleint to initiate the session. This clss implements sendStreamListener and RemoteListener interfaces.

Version:

1.0, 9-Oct-2001

Author:

Ksum

See Also:

VTClient

VTServer

Void devoces action Performed (java. lang. String ip)

D This function is used to cerate the session with the client.

Voi update(SendStreamEvent evt)

D This function is used to handle SendStreamEvent.

Constructor Detail VTServer

Public VTServer()
Method Detail

Devices actionPerformed

Public void devices actionPerformed(java. lang. String ip)

This function is used to cerate the session with the client. This is the main function of the class which performs all the task of the class.

Parameters:

Ip – IP address of the client with which session will be initialized.

Update

Public void update(SendStreamEvent evt)

This function is used to handle SendStreamEvent.

Parameters:

Evt –

13.2.5.5 CLASS VTCLIENT

VTClient

Public class VTClient

This class initiate the Audio session with the Server. It accepts request of VTServer to initiate the session. This class implements ReceiveStreamListener abd RemoteListener interfaces.

Version:

1.0, 9-Oct-2001

Author:

Ksum

See Also:

VTServer

Constructor Summary

VTClient()

Method Summary

SessionManager createManager(java, lang, String address,
R int dataport, int cport, Boolean listener,
Boolean sendlistener)

This function is used to cerate the session with the Sever.

Void update(ReveiveStreamEvent event)

Void update(Remoteevent revent)

This function is used to handle Remoteevent.

VTClient

Public VTClient()

createManager

public SessionManager createManager(java. lang. String address,
int dataport
int cport,
boolean listener,
Boolean sendlistener)

This function is used to cerate the session with the Server. This is the main function of the class which performs all the task of the class.

Parameters:

Address – IP address of the client with which session will be initialized.

Dataport – data port for the communication

Cport – control port for the communication

Listener-

Senlistener –

Returns:

SessionManager

Update

Public void update(ReceiveStreamEvent event)

Update

public void update(Remoteevent revent)

This tunction is used to hndle remoteEvent.

Parameters:

Revent –

13.2.5.6 CLASS MYCONTROLLERLISTENER

My ControllerListener

Public class MyControllerListener

This class is used for transition of the player from one state to other.
Version:
1.0, 9-Oct-2001
Author:
Ksum
See Also:
VServer
Method Summary
Void close()

This function is used to close the palyer.
boolean configure(int timeOutMillis)
This function is used to configure the palyer.
Void controllerUpdate(ControllerEvent ce)
This function is used for event handling.
Boolean playToEndOfMedia(int timeOutMillis)
This function is used to detect end of media stream for the palyer.
Boolean prefetch(int timeOutMillis)
This function is used for prefetching.
Boolean realize(int timeOutMillis)
This function is used tko realize the palyer.
Method Detail

Configure

Public Boolean configure(int timeOutMills)
This function is used to configure the palyer.
Parameters:
Int – Time to stay in that state
Returns:
Boolean of the player

Close

Public void close()
This function is used to close the palyer.

playToEndOfMedia

public Boolean playToendOfMedia(int timeOutMillis)
This function is used to detect end of media stream for the palyer.
Parameters:

Int – Time to stay in that state
Returns:
Boolean state of the stream

Realize

Public Boolean realize(int timeOutMillis)
This function is used to realize the palyer.
Parameters:
Int – Time to stay in that state
Returns:
Boolean state of the player

Prefetch

Public Boolean prefetch(int timeOutMills)
This function is used for lprefetching
Parameters:
Int – Time to stay in that state
Returns:
State of the palyer

controllerUpdate

public void controllerUpdatre(ControllerEvent ce)
This function is used for event handling.
Parameters:
Ce – event to uupdate the controller

13.3 JAVA SPEECH API

.Java Speech API Specification—Defines the packages used to implement basic Java Speech capabilities, speech recognition and speech synthesis.

.Java Speech Programmer's Guide_Describe how to use the Java Speech API to develop speech-enabled application .

.Java Speech Grammar Format (JSGF) Specification-Describe the JSGF and explains how it is used to create platform –independent speech recognition grammars. These grammars identify the words that a user speaks and their meaning in particular program contexts.

.Java Speech Makeup Language (JSML)Specification—Describe the role of JSML and shows how it's used to mark text documents for use with speech synthesizers.

The Speech API consists of the following three packages:
Java.speech-provides classes and interfaces that support audio connectivity and manage the use of speech processing engines.

.Javax.speech.recognition-Provides classes and interfaces that support speech recognition.

.Java.speech.synthesis-Provides classes and interfaces that support speech synthesis.

The Java.speech packages consists of the following classes and interfaces:

.Central-class that provides central access (via static methods) to all capabilities of the speech API.

.Engine-interface that is implemented by speech recognition and synthesis engines.

.Engine Attributes-Defines the attributes that are supported by an engine object.

Engine Central-Provides the operating modes of a speech engine in terms of Enginmodedesc objects.

.Engine Mode Desc –Defines an engine operating mode

.Engine List- A collection of EngineModeDesc objects.

.Audio Manager – Interface that defines methods for controlling audio input and output and managing audio events.

.VocabManager - Interface. That defines methods for controlling words that are used by a speech engine.

.Word – Encapsulates speakable words.

.Speech Event – The super class of all speech events.

.Audio event – Subclass of speech event that is generated by speech engine objects based on audio input and output and output processing.

.Audio Listener – Defines methods for handling audio event objects.

.Audio Adapter – Implementation of the audio lister interface.

.Engine Event – Reports changes in speech engine status.

. Engine Listener – Defines methods for handling engine event objects.

. Engine Adapter – Implementation of the engine listener interface.

The following sections cover the Javax.speech.recognition and javax.speech

Speech recognition is supported by the Javax.speech.recognition package, Which consists of 15 interfaces and 19 classes. These classes and interfaces makeup four major groups: Recognizer, Rule, and Result.

The Recognizer interface extends the engine interface to provide access to a speech recognition engine. Recognizer Attributes and recognizer Mode Desc are used to access the attributes and operational modes of the recognizer. Recognizer objects generate recognizer interface defines methods for handling these events. The recognizer Adapter class provide a default implementation of this interface. The audio level event is a granted as a result of a change in the audio level of a recognizer The recognizer audio listener interface defines methods for handling this event, and the recognizer audio adapter class provides a default interface implementation.

The grammar interface provides methods for handling the grammar used by a recognizer. It is extended by rule grammar and dictation grammar. Which support the grammar and dictation grammar.

The grammar syntax detail class is used to identify error in a grammar. The grammar event class is used to signify the generation of result object that matches a grammar. The grammar listener interface define methods for handling this event, and the grammar adapter class provides a default implementation of this interface.

The rule class encapsulates rules that are used with a rule grammar. It is extended by rule alternatives. Rule count, rulename, ruleparse, rulesequence, ruletag, and ruletoken. Which specify different aspects of grammar rules.

JSML allow text to be marked up using the following synthesis-related information

.Paragraph and sentence boundaries

.Pronunciation of words and other text elements

.Pauses

.Emphasis

.Pitch

.Speaking Rate

.Loudness

These capabilities may not have your computer reading poetry ,but they allow you to greatly enhance any speech that it generates.

The Java Speech API supports speech generation via the javax. Speech. Synthesis package. This package provides the following five interfaces:

- Synthesizer-Extends Engine to implement a synthesizer engine.
- Synthesizer Attributes-provide access to the control attributes of a synthesizer.
- Synthesizer Mode
- disc- Used to specify an operational mode for a synthesizer.
- Synthesizer Event- Event generated by a synthesizer as it changes state.

- Synthesizer listener –Defines methods for handling synthesizer event objects.
- Speakable-An interface for providing JSMI text to a synthesizer.
- Speakable event-Event generated by a speakable object that identifies the state of JSML processing.

- Speakable listener-Defines methods for handling speakable event objects.
- Speakable adapter- An implementation of the speakable listener interface.
- Voices-Allows the age and gender if a voice to be specify

13.4 JAVA TELEPHONY API

The JTAPI consists of the following 18 packages:-

- java . telephony –Provides the core classes and interfaces used by all telephony applications.
- Javax. Telephony capabilities_Provides support for basic call and connection capabilities.
- Javax. Telephony. Events ---Defines the basic events used in all telelphony applications.
- Javax . telephony callcenter---Provides support for developing call center applications.
- Javax. Telephony callcenter---provides support for developing call center applications.
- Javax. telephony. Callcenter. Capabilities---provides capabilities such as routing and automated call distribution used in call center applications.
- Javax telephony. callcenter .events—Defines the events used in call center applications.
- Javax. Telephony. Call controls—provides call control feature, such as call hold, call transferring and conferencing.

- Javax. Telephony. Call control. Capabilities---Extends the interfaces of the basic javax. telephony. capabilities package to support call control applications.
- Javax. telephony. Call control. Events---Defines the events used in call control applications.
- Javax. telephony. Media---Supports media streams (touch tone and non-touch tone) used in telephony media-exchange applications.
- Javax telephony. Media. Capabilities---Defines the Media terminal connection capabilities interface, which support media steaming applications.
- Javax. telephony. Media. Events—defines the events used with media streams.
- Java Telephony. Provides control over the physical features of telephone equipment.
- Javax. telephony. Phone capabilities—provides interfaces for controlling equipment components.
- Javax telephony. Phone. Events---defines the events used with the javax. telephone. Phone package.
- Javax telephone private data---provides classes for accessing telephone hardware switches.
- Javax. telephony private data. Capabilities---provides an interface that is used to access the capabilities provided by javax. telephony. Private data.
- Javax telephone private data. Events ----defines the events used with the javax. telephone private data package.

The terminal interface is used to provide access to a physical hardware device at the endpoint of telephone connection. For example telephone sets are accessed as terminal objects. The terminal connections interface

provides physical access to a telephone connection. While the connection interface models a logical connections between a call object and an address object.

An instance of the call interface represents an actual telephone call.The address interface is used with a telephone number or in internet telephony applications an IP address combined with other end point information. The provider interface is used to access a telephony /service provider soft ware element.

While the exact requirements vary from one speech engine to another. The faster the CPU and the more memory available, the higher the Accuracy for speech recognition and better the text-to-speech sounds.
A sound card, microphone, and speakers are also needed. Most speech engines will work with any sound card. Some system offload processing onto a DSP (digital).

14. CLASS HIERARCHY

JCOMM

Class. Java. lang. object

Interface. Javax. Comm.. CommPort

Class javax. Comm.. ParallelPort

Class javax. Comm.. SerialPort

Class. Javax. Comm.. comm.PortIdentifier

Interface. Javax. Comm.. commPortOwnershipListener(Extends
Java. util. EventListener)

Class . java . util. EventObject(implements java.io. serializable)

Class. Javax. Comm.. parallelportEvent

Class. Javax. Comm.. SerialPortEvent

Interface javax. Comm.. parallelportEventListener(extends
Java. util. EventListener)

Interface. Javax. Comm.. SerialPortEventListener(extends
Java. util. EventListener)

Class. Java. lang. Throwable(implements java.io. serializable)

Class. Java. lang. Exception

Class. Javax. Comm.. NoSuchPortException
Class javax. Comm.. portInUseException
Class. Javax. Comm.. UnsupportedCommOpeation Exception

15. Conclusion

On the whole, the project has been a success as it gave us a valuable chance to gain knowledge in the latest fields of Java telephony and e-

commerce.The experience gained during the course of this project will help us in our future endeavours.This project will indeed a milestone in our academic and professional careers.

While analyzing, designing and implementing this project, we tried to apply all we had studied in different courses through out this undergraduate degree program and specially has to clear our concepts in the subjects of Digital communication,analog communication,Line Transmission Theory,Programming in C++,Computer Architectureand Switching Networks. This exercise not only allowed us to implement the theoretical concepts of Telecommunication and Computer Science, but also provided us with an excellent opportunity to revise and refresh everything in great detail, making us ready for the challenges of practical life and abreast of the cutting edge technological breakthroughs.

16. Future Expansion Possibilities

There is a lot of room for future expansions in this system.Some of the other features that can be added to the automatic inquiry system include the following.

COMPLETE EXCHANGE FACILITIES

Major break through for the development of an automated exchange has been made and now it's just adding up of more characteristics.This can be used for connecting of calls,call holding,call busy,automated complaint system and other exchange facilities can very easily be added.

Excluding exchange facilities,we can add other facilities to it as well,which are as under

News Update

The news update will allow the users to listen to latest news via any touch-tone phone from anywhere.The news will be updated at the exchange regularly via Internet. The text-to-speech technology will be incorporated to convert the news in text form to speech, enabling the user to listen to the latest news update.

Weather forecast

Similarly the weather forecast will keep the users in touch with the latest weather forecast. The user will dial his code number and will choose the weather forecast option from the available options.Weather forecast will be constantly updated at the exchange.Text-to-speech technology will make the weather forecast available to the user in audible form via telephone.

Sports News

The sports update service will allow the users to listen to latest sports news and live commentary of important sports events.Again the user will dial the code number and will choose the sports option.The server will establish

connection with an online radio/sports station and will allow the user to listen to live commentary or sports update being broadcasted from that station.

Flight Timings

Flight timings service will inform the users about the flight schedules of the local as well as international airlines .The access and lookup procedure for this service will be more or less the same as in case of the news and sports update.

On Line transaction Processing

Online Transaction Processing (OLTP) service will allow the users to purchase or order products via any touch-tone phone. Online Transaction Processing will involve ordering products from a list of offered products by just dialing the code number.These may include placing orders to a restaurant or purchasing an item from a departmental store.

Call Routing

Many of the same telephony applications that provide voice mail or database access also allows incoming calls to be routed to other phone line. Because most contemporary call routing systems rely on DTMF(touch-line) to rout the call they ask for an extension number, but with speech recognition this could be very recommendable. So call routing can be easily Managed through programming in Java telephony API.

MultiLine Applications

Most telephony applications are designed to handle several phone lines comming into the same PC.Multi line telephony applications need to be

designed to handle the multiple input channels in such a way that one channel doesn,t slow down or harm another channel.Furthermore, Special devices are available in the market that allow up to 1**6** incoming lines.

Audio on Demand

We can record audio messages on individual subscriber's numbers through telephone and subscriber can easily acces these messages on arrival.

17. BIBLIOGRAPHY.

Resources.

1.http://java.sun.com/products/javacomm/

2.http://www.embedded.com/2002/toc9801.htm

3.http:ridgewater.mnscu.edu/classes/dc/io/

4.The book understanding data communications,by Gilbert Held and George.

5.http://www.clbooks.com/sqlnut/SP/

6.http:bbec.com/catalog/software/serialte.html

7.http://www.openbsd.org/

8.http://www.javaworld.com/javaworld/jw-12-2002/jw-12-javadeev.html.

9.The January issue of Javaworld ran the second article in the smart card series.:"Smart Card and the open card Framework."

10.http://www.javaworld.com/javaworld/jw-02-2002/jw-02-javadev.html.

11.http://www.javaworld.com/javworld/jw-01-2002/jw-01-javadev.html.

12.http://www.javaworld.com/javaworld/jw-03-2002/jw-03-javadev.html

13.http://www.javaworld.com/javaworld/jw-02-2002/jw-02-javacard.html

14.Book on Java 2(JDK 1.3 edition) by Ivor Horton.

15.The complete reference JAVA 2(Fourth edition) by Herbert Schildt
.
16.JAVA 2 introducing swing,by DEITEL and DEITEL.

17.JAVA I/O by Elliotte Rusty Harold(international edition).
…..