FILE COPY

US Army Corps
of Engineers
Construction Engineering
Research Laboratory

USACERL Technical Report P-89/19
September 1989
Management Tools for Civil Works Operations and Maintenance

AD-A214 129

Communications Alternatives for Transparent Data Transfer in Civil Works Computer Systems

DTIC
ELECTE
NOV 0 7 1989
S D
D

by
E. J. Japel
C. L. Raaymakers
K. Stephenson
W. J. Schmidt

Although microcomputers are becoming increasingly powerful, there are still applications for which mainframes are better suited. However, the relatively low operational costs and high level of friendliness make the microcomputer a more attractive machine for data input. Distributive processing systems have been created to allow data input at the microcomputer level and processing at the mainframe level. These systems require the computer operator to transfer the file or data from the micro to the mainframe and back; this task often can be both tedious and difficult. In addition to data/file transfer, distributive systems must be able to be updated.

One solution to this problem is a transparent communications system. Such a system frees the user from controlling and monitoring the communications link between mainframe and microcomputer. This report describes the development of two transparent communications systems to facilitate use of automated data processing tools used at Civil Works Districts. Included are a distribution (or update) system specific to a single application and a data transfer system.

Approved for public release; distribution is unlimited.

89 11 06 168

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official indorsement or approval of the use of such commercial products. The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED
DO NOT RETURN IT TO THE ORIGINATOR

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704 0188
Exp Date Jun 30 1986

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED

1b RESTRICTIVE MARKINGS:

2a SECURITY CLASSIFICATION AUTHORITY:

2b DECLASSIFICATION / DOWNGRADING SCHEDULE:

3 DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.

4 PERFORMING ORGANIZATION REPORT NUMBER(S): USACERL TR P-89/19

5 MONITORING ORGANIZATION REPORT NUMBER(S):

6a NAME OF PERFORMING ORGANIZATION: U.S. Army Construction Engr Research Laboratory

6b OFFICE SYMBOL (If applicable): CECER-FS

7a NAME OF MONITORING ORGANIZATION:

6c ADDRESS (City, State, and ZIP Code): P.O. Box 4005, Champaign, IL 61824-4005

7b ADDRESS (City, State, and ZIP Code):

8a NAME OF FUNDING / SPONSORING ORGANIZATION: HQUSACE

8b OFFICE SYMBOL (If applicable): CECW-OM-B

9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER:

8c ADDRESS (City, State, and ZIP Code): 20 Massachusetts Ave., NW. Washington, DC 203.4-1000

10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
	CW32251		

11 TITLE (Include Security Classification): Communications Alternatives for Transparent Data Transfer in Civil Works Computer Systems (Unclassified)

12 PERSONAL AUTHOR(S): Japel, E.J.; Raaymakers, C.L.; Stephenson, K.; Schmidt, W.J.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM _____ TO _____	1989, September	50

16 SUPPLEMENTARY NOTATION: Copies are available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP
12	06	

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number): microcomputers; minicomputers; data transfer; transparent communications systems; civil works

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Although microcomputers are becoming increasingly powerful, there are still applications for which mainframes are better suited. However, the relatively low operational costs and high level of friendliness make the microcomputer a more attractive machine for data input. Distributive processing systems have been created to allow data input at the microcomputer level and processing at the mainframe level. These systems require the computer operator to transfer the file or data from the micro to the mainframe and back; this task often can be both tedious and difficult. In addition to data/file transfer, distributive systems must be able to be updated.

One solution to this problem is a transparent communications system. Such a system frees the user from controlling and monitoring the communications link between mainframe and microcomputer. This report describes the development of two transparent communications systems to facilitate use of automated data processing tools used at Civil Works Districts. Included are a distribution (or update) system specific to a single application and a data transfer system.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT: ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS

21 ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL: Dana Finney

22b TELEPHONE (Include Area Code): (217)352-6511 (X389)

22c OFFICE SYMBOL: CECER-IMT-E

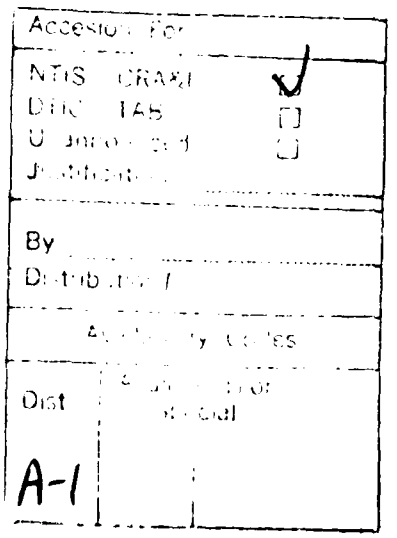
DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

FOREWORD

This research was conducted for the Directorate of Civil Works, Headquarters, U.S. Army Corps of Engineers (HQUSACE), under Civil Works Project 32251, "Management Tools for Civil Works Operations and Maintenance." The HQUSACE Technical Monitor was David Harmon, CECW-OM-B.

The work was performed by the Facility Systems Division (FS) of the U.S. Army Construction Engineering Research Laboratory (USACERL). Dr. Michael J. O'Connor is Chief, FS. The USACERL technical editor was Dana Finney, Information Management Office.

COL Carl O. Magnell is Commander and Director of USACERL, and Dr. L.R. Shaffer is Technical Director.



CONTENTS

TABLE

FIGURES

COMMUNICATIONS ALTERNATIVES FOR TRANSPARENT DATA TRANSFER IN CIVIL WORKS COMPUTER SYSTEMS

1 INTRODUCTION

Background

Technological advancements in the microcomputer have had tremendous impact on the way business is conducted, both in the private sector and the Government. Their comparative low cost and ease of use compared with mainframe computers have made micros the system of choice today.

Despite the microcomputer's capabilities, there are still some tasks for which mainframes are better suited. However, mainframes require complex skills for interaction and often are difficult to access due to the number of users logged in; these conditions tend to lower the efficiency for which most automated tools are designed. A popular solution to this dilemma is to create micro systems that mimic the mainframe application. The parallel system collects the same data in the same format while taking advantage of the microcomputer's speed, flexibility, and ease of access/use. Data are then transferred electronically to the mainframe.

A drawback to this approach is that, while data can be transferred using several different methods--most of which are classified as "distributive processing systems"--the process is seldom user-friendly. These systems have problems in three areas: (1) maintaining up-to-date personal computer (PC) programs, (2) actually transferring the data properly, and (3) providing a communications system that the user can operate.

One way to avoid these problems is to use transparent communications programs. These programs are responsible for completing the dialog necessary to transfer data from the microcomputer to the mainframe; the user only has to enter the commands to start the process and then everything else is done automatically. "Transparent" thus refers to the level of human involvement, in that the user need not know anything about how the communication is achieved, making the procedure "invisible."

The U.S. Army Construction Engineering Research Laboratory (USACERL) has been developing a family of integrated management tools to help U.S. Army Corps of Engineers (USACE) Civil Works managers budget resources optimally. One of these systems--the Automated Budget System (ABS)--is being implemented at many Civil Works District offices. A similar system, ECONPACK, was developed using many of the same principles, and is available throughout the Army on the Programming, Administration, and Execution (PAX) system.[1] ECONPACK supports economic analyses required for the Military Construction, Army (MCA) and related programs. Both systems were originally developed as mainframe applications and both now have PC versions requiring data transfer to the mainframe. In addition, it is anticipated that future systems developed in

[1] R. D. Neathammer and J. D. McLean, *Economic Analysis: Description and Methods*, Technical Report P-89/08/ADA204264 (U.S. Army Construction Engineering Research Laboratory, December 1988).

this family will be configured in the same way. Thus, some type of transparent communications program is needed to facilitate data transfer and enable Civil Works users to gain maximum benefit from these computer tools.

Objective

The objective of this work is to develop transparent communications systems capable of interfacing with mini-/microcomputers to transfer data to remotely located mainframes.

Approach

The method used to develop transparent communications systems consisted of three phases. Phase 1 involved gathering information on relevant communications topics to provide a basis for system design decisions. Several topics require careful consideration when a system is in the initial planning stages (see Chapter 2):

- Computing hardware
- Communications hardware
- Communications media
- Data/file type
- Communications standards
- Error checking protocols
- Communications problems
- Recovery mechanisms
- Prepacked programs/application-specific programs
- Data transfer, electronic update distribution (EUDS), or both.

Phase 2 was system design and phase 3 was the actual coding of the system. Chapter 3 details the design and problems encountered during development of the transparent communications systems.

Mode of Technology Transfer

The two transparent communications systems described in this report are being used in the field--one to transfer data for the ABS and the other serving ECONPACK. Civil Works District personnel have received training along with documentation from USACERL on use of the ABS software. ECONPACK has been turned over to the USACE Huntsville Division for maintenance and the system is currently in use at all installations required to submit economic analyses. Huntsville Division conducts an annual PROSPECT course on use of ECONPACK, including the transparent communications software. For both systems, user groups have been established to facilitate transfer to the field.

2 ISSUES CONSIDERED IN SYSTEM DEVELOPMENT

As noted in Chapter 1, distributive processing systems have problems in keeping PC programs current and transferring data without complex interactions. These drawbacks have considerable impact on the system's success.

Maintaining up-to-date software is essential to the smooth operation of distributive systems. Upward reporting of data requires that all calculations be done using the same tables and formulas that would be used if the data were entered directly into the mainframe. This task could be accomplished by mailing updates on diskette; however, given the huge number of PCs involved worldwide and the frequency with which updates are necessary, this approach would be unrealistic in terms of both cost and manpower required. Electronic communications systems are generally accepted as being the most reliable and cost-effective method of updating software.

Manual transfer of files from microcomputers to mainframes often requires in-depth knowledge of programming. For the typical user, an instruction such as "Shuffle bits and bytes over asynchronous com lines at a rate of 1200 baud using X.25 (error checking protocol)" makes little or no sense. The complexity of this transaction is beyond the skills of most persons for whom the system was designed to use on a daily basis.

To be a useful transfer medium, the transparent communications system must overcome these drawbacks. Ideally, updating and data transfer would be automatic, requiring very little user intervention and no special understanding of the process. To design systems meeting these conditions, it was first necessary to consider several issues that would have an impact. This chapter describes these issues.

Computing Hardware

There are numerous issues to consider in selecting the computing hardware for a distributive processing system since the type of hardware will have major impact on the system. Each possibility and combination must be considered and tested. Hardware generally consists of central processing units (CPUs), operating systems, communications equipment, storage media, communications media, input devices, and output devices. For communications system development, only CPUs, operating systems, communications equipment, and communications media must be considered.

CPUs

CPUs are available in three classes: mainframe, minicomputer, and microcomputer. A distributive processing system will often use more than one of these hardware platforms. Data entry, word processing, and recordkeeping may all be done at the microcomputer level, whereas data analysis and report generation are accomplished at the mainframe level. Each class of CPU has a unique command set and procedures to complete identical tasks.

Mainframes. Mainframe CPUs often are grouped into a single category, but they are far from being the same. Access to software and hardware peripherals such as disk drives, communications ports, and printers varies among machines. This variability is a function of the mainframe operating system that controls all computer operations. The relevance of the operating system lies in the ability of the PC or mini to speak its language. All communication, whether spoken, written, or electronic, is dependent on both

parties' speaking a common language. Without a common reference, neither party will be able to distinguish the meaning of the other's statements.

To develop a transparent communications system, the system designer must first determine what mainframe(s) the system will be expected to access. The more host mainframes that will be involved, the more complex the system will be. From a developmental perspective, there are three possible scenarios:

- One mainframe accessed by all users (Figure 1)
- Multiple types of mainframes accessed by all users (Figure 2)
- Multiple mainframes of the same type, each with its own user set (Figure 3).

The single mainframe system is the simplest in terms of development. Since all users are accessing the same machine, the dialog between mainframe and PC will be identical in all cases. The scenario with multiple types of mainframes and multiple users raises the problem of syntax differences between operating systems. The final scenario is very similar to a single mainframe system. At first glance, both systems appear to be the same. However, the third configuration (multiple mainframes of one type, each accessed by its own user set) raises some very unique, often frustrating problems.

Mainframe systems cannot be considered constant since they are often customized to fit the user group's needs. An example would be working with the Harris 500s at 40 different Civil Works sites. Variable options may be configured differently, the version of the operating system may vary between sites, and special commands may or may not be available. In short, mainframes can be as variable as PCs and this variability can create problems.

Minicomputers. Unix-based minicomputers include those such as Sperry and Sun; non-Unix systems include those in the DEC VAX series. At one time, the distinction between minicomputers and mainframes was well defined. Mainframe systems had multiple processors available to do many tasks simultaneously. In contrast, the minicomputer had only a single processor to do one task at a time. Today mainframes and minicomputers are so similar that it is sometimes difficult to distinguish between them.

For the communications system developer, the mainframe and the minicomputer present the same sets of problems. In the scenarios described above, a minicomputer might be used in place of a mainframe, provided the developer adapts it to the new syntax.

Microcomputers. The microcomputer is the most variable of all CPUs. PCs are made by many companies, with each machine having its own way of processing information. A few examples are Commodore, Radio Shack, Apple, IBM, Compaq, Zenith, Toshiba, AT&T, and Wang. Among the business computers, most are compatible with the industry standard, IBM. Because of patent restrictions, manufacturers use different chips and designs to reach the same compatible end. As long as the CPU is truly IBM-compatible, this area of hardware is of no concern to the transparent communications system.

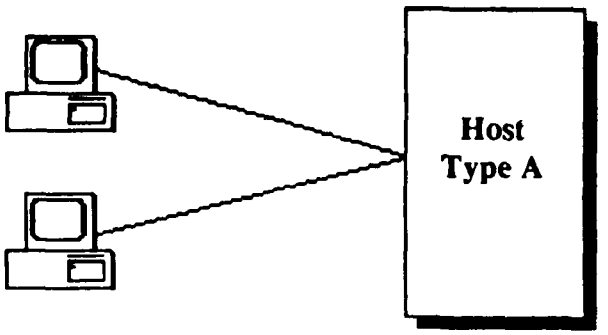


Figure 1. One mainframe accessed by all users.

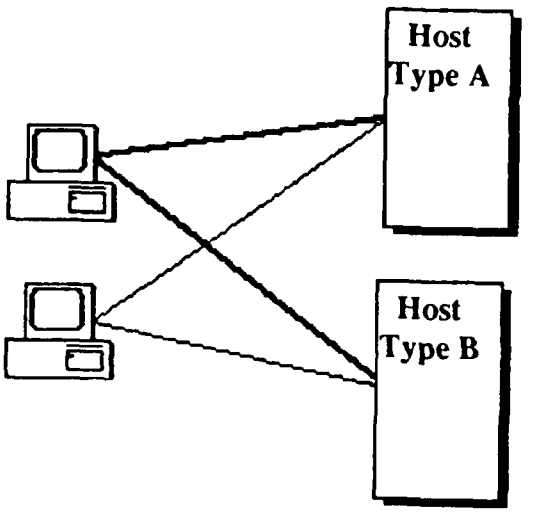


Figure 2. Multiple types of mainframes accessed by all users.

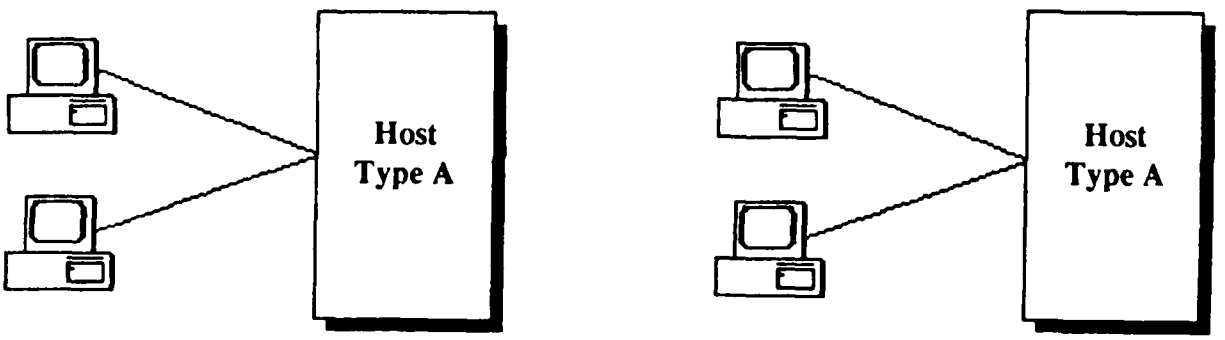


Figure 3. Mainframes of the same type, each with its own user set.

For communications, the most vital portion of the microcomputer CPU architecture is the chip known as the Universal Asynchronous Receiver/Transmitter (UART). The UART chip is responsible for managing both data input from and output to a communications port. This function relieves the programmer from the tedious task of assembling and disassembling bits.

The UART chip function is based on a transmit data clock. To output a byte, the individual bits are written to a hardware "latch," which then drives the interface to some outside device such as an RS-232 cable. Basically, the latch consists of a transmitter holding register, shift register, and transmit data clock. The byte is written into the holding register. It is then passed into the shift register. Data are formatted into a Serial Data Unit (Figure 4) for transmission and, finally, the byte is shifted, one bit at a time, out the port on the negative transition of the clock. Formatting involves adding the bits for start, parity, and stop. Receiving a byte essentially involves reading successive bits from a similar latch, stripping off the start, parity, and stop bits, and assembling the remaining data bits into bytes.

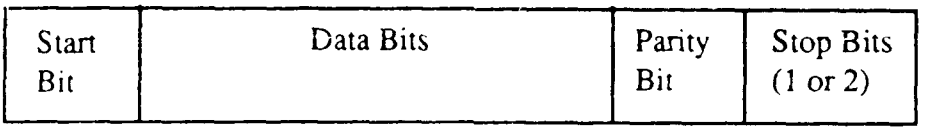


Figure 4. Serial Data Unit.

The start and stop bits are used to establish the delicate timing relationship between the transmitter and the receiver. The parity bit is used as a binary check value for error detection. There are five types of parity: even, odd, mark, space, and none. The UART calculates the parity bit based on the values of the received data bits. It then compares this calculated value with the received value. If the values do not match, an error bit is set for use by the communications program.

Many types of UART chips have been designed to accomplish these tasks. Although most of these chips are compatible with each other, there is always a potential for unusual discrepancies between them. Compensation can be made for these differences as they are detected. With the advent of IBM's new line of microcomputers, the PS/2s, there may be other unforeseen problems to overcome. The PS/2 models 50, 60, and 80 use a new, improved UART chip (16550). This new chip allows for baud rates of up to 19,200. The chip used in the XT and AT (16450) units only allowed for rates up to 9600.

The operating system for the IBM style microcomputer, MS-DOS, has undergone rapid changes over a very short time. Enhancements and improvements have given the developer access to many new features. In addition, the OS/2 version will provide even more capabilities to the communications programmer. However, since not all users in the field will have the most state-of-the-art equipment, the communications system

must specify a minimum DOS version as well as be upwardly compatible to higher DOS versions.

A few manufacturers do not provide IBM-compatible equipment, such as Apple. For example, the Apple MacIntosh does not attempt to meet IBM standards, but has a standard user interface common to all Mac programs. IBM has no such standard. A communications developer could conceivably be required to create and support code on both styles of machine. To avoid developing two completely separate sets of code, a Mac-like user interface for the IBM could be created using a highly portable language such as C. The developer would create two libraries containing identical routines--one for the Mac and one for the IBM--to perform nonportable operations such as video handling. The final product would be a single communications program.

Communications Hardware

The hardware that connects computers and allows them to communicate varies greatly. One of the reasons for this variation is that different types of communications media usually require different hardware. For example, a modem that allows a PC to communicate over telephone lines cannot be used to reach Ethernet. Another reason communications hardware varies so much is that it is often machine-dependent. For example, hardware that allows a PC to communicate over Ethernet generally cannot be used in a mini- or mainframe computer.

Modems

A modem is a device that translates computer data and makes it "voice-like" so that it can be transmitted over a telephone line. Some modems have an autodial capability, which allows the computer to dial a number and connect to another computer without human involvement. Modems without this capability make the goals of transparent communications more distant because someone must dial the telephone manually and make a connection.

One of the largest PC modem manufacturers, Hayes, has created a defacto modem "language" called the Attention (AT) Command Set. The PC uses this language to command the modem to, among other things, dial a new number or hang up. Modem manufacturers often claim "Hayes compatibility" for their modems, implying that their products understand most of the commands that a Hayes modem would. However, some of the omitted commands are crucial and can only be obtained (if at all) by writing special code tailored for each non-Hayes modem.

Networks

The concept of connecting computers into networks so that they can share data is very powerful and is becoming increasingly popular. Some networks are small, connecting several computers in the same room, whereas others span continents and connect thousands of computers. Networks can be server-based, in which the network itself offers file, print, and other services, or nonserver-based, in which it acts as a pass-through to another computer system. Configuring computers so that they will exchange data reliably can be a major task involving myriad communications issues such as parameters, protocols, and media. Networks often simplify this task. Networked computers can already communicate and many communications problems have already been tackled. The two types of networking considered for this study are worldwide and PC-based.

Worldwide Networks. Employing worldwide networks such as Tymnet and Telenet helps simplify transparent communications system development because the procedure for using the network remains constant, regardless of location. This consistency means that the code which logs in the user and communicates with the network can be used with little or no modification anywhere, decreasing development time.

Worldwide and PC-based networks offer similar services, but the way in which they are provided can differ greatly. For example, worldwide networks offer file storage services on the connected mainframes or minis, whereas these services are usually server-based in PC-based networks. Writing software that knows the type of network and services, and how to access those services can be quite difficult.

PC-Based Networks. PC-based local area networks (LANs) such as Banyan and 3Com are becoming increasingly popular. These networks offer both advantages and drawbacks with regard to transparent communications. For example, if data needs to be downloaded from a mainframe to all the PCs on a network, an advantage is that only one machine need actually receive the data from the mainframe. This machine can then share that information with other PCs via the network, greatly speeding data exchange. This same feature has a disadvantage, however, because machines wanting to access data on the network must somehow be informed of its location. Furthermore, special code needs to be written to do this network sharing--often for each different type of network.

Downloading and uploading data also introduce problems. The program trying to transfer data must know the data location or destination; a common solution is to ask the user. However, in a PC-based network environment, this solution may not always work. Users may not know if their data is being stored locally on their PCs or on the network. Or, if their data is on the network, they do not always know where it is or how to get to it. In addition, the system must be able to link the storage area. For different networks, this capability requires different command and error trapping sequences.

Since networks allow peripherals to be shared, it might be desirable for PCs wanting to send or receive data to use a common, shared modem. Although this practice makes efficient use of communications hardware, it introduces many new software-related difficulties. Software that communicates with a modem over Ethernet is more complex, more difficult to maintain, and must handle a much greater number of possible errors than software that communicates with a modem directly connected to a PC.

A further complication when sending data over a network is that network software sometimes intercepts special control characters, which can crash the system, throwing it into an infinite loop. Xon and Xoff are common examples of these special characters. The actual binary character definition may vary among machines; however, Xon and Xoff are commonly defined as hexadecimal 11 (control Q) and 13 (control S). The main difficulty in detecting this problem lies in the fact that most text files are written in American Standard Code for Information Interchange (ASCII), which does not contain such characters. Everything seems to work fine until a binary file containing control characters is sent and the PC crashes--seemingly at random--because the characters are not recognized. See **Data File Transfer** below for a more detailed explanation of this problem.

Hardwired Options

Hardwiring a PC to a mainframe can both simplify and complicate transparent communications. For a hardwired PC, connection to the mainframe is always possible; no special procedures or routing commands are needed. This configuration is used most

often when the desired mainframe is at the same physical location as the PC. However, in a distributive system with many users dispersed over a large area, it is impractical to require that all users be hardwired to the mainframe. A worldwide network can serve these users' needs much better.

It is possible to be hardwired directly to the network if a network gateway is available onsite. Transparent communications software must be able to differentiate between connection protocols, sending connection commands only when the system is not hardwired. One way to achieve this recognition is to run a "setup" program when the software is first installed. It would ask the user what type of connection exists and then save that information for future use. Another solution would be to have the software automatically determine what type of connection exists.

Communications Media

RS-232-C Interface

For two devices to communicate, there must be a standard interface between them. An example interface is the Recommended Standard 232-C (RS-232-C), a device or cable using a 25-pin connector, with each pin representing one of 25 standard configurations.

An RS-232-C interface permits serial data flow. Therefore, data characters must be converted to a serial bit stream and then framed by control codes. Since the data and control codes follow specific rules, there must be a method on either end of the data path to encode/decode data coming through the line. The most common example of an RS-232-C connection has two hosts, each connected to a modem. Communication is accomplished through a telephone line bridging the two modems (Figure 5). The software on the hosts opens the pathway to each host communication port and controls the amount of information sent to the modem at once. The RS-232-C interface supports speeds up to 20K bits per second (bps).

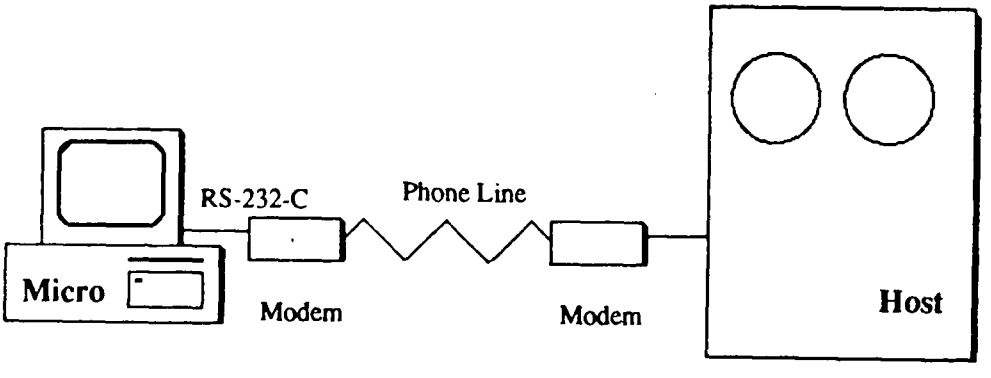


Figure 5. Modem connection.

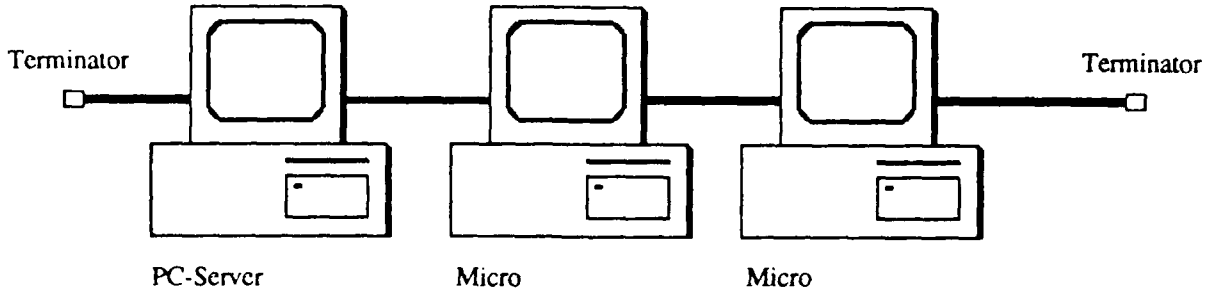


Figure 6. Linear bus.

Ethernet

Ethernet is a network protocol primarily run on "thick" or "thin" coaxial cable. The topology of the cable is a linear bus with PCs arranged along a single length of cable that can be extended at either end (Figure 6). Since coaxial cable is a shielded medium, it is largely immune to noise and can carry data at higher rates over longer distances than can twisted-pair cables. Connection to the LAN is via (1) a network interface card (NIC), which is inserted into each PC, and (2) coaxial cable.

With Ethernet, all transmissions are broadcast over the network. The most common Ethernet access method is Carrier Sense Multiple Access (CSMA). In a CSMA network, all stations have the ability to see activity on the network. When a PC is ready to transmit, it "listens" on the main channel. If the NIC senses activity on the channel, it waits a random length of time and retransmits. If transmitted packets collide on the cable, the NIC detects it and, again, waits a random time period before retransmission.

Token Ring Network

Token ring is a data-signaling scheme in which a special data packet (called a "token") is passed from one station to another along an electrical ring. Token ring networks are primarily run on twisted-pair cable. They use a deterministic token-passing system in which stations distribute the right to transmit on a channel by passing the token. A station that wishes to transmit waits until it receives a token from the previous station and attaches data information to the token. The receiving station strips the token of its data and passes onto the next station.

Data/File Transfer

Binary/ASCII Transfer

ASCII (or text) files use standard 7-bit character codes and contain a minimum of control codes. Control codes are essentially formatting functions that are machine-readable. They perform actions such as carriage return, end-of-file marker, beep, and form feed. The standard 7-bit ASCII codes include the alphabet (A through Z in upper and lower case), assorted punctuation marks, the digits 0 through 9, and a few control codes. Thus, an ASCII text file is human-readable as opposed to machine-readable.

In addition, IBM has developed 8-bit ASCII codes that include a set of 128 codes available for manufacturers to use as they wish. These 8-bit ASCII codes have become somewhat of a standard in the PC world. They include the same characters as are represented in 7-bit ASCII plus many graphics characters and some additional control codes. It is this "extended" character set (especially the control codes) that sometimes causes problems in data transfer.

Binary files often use the special 8-bit ASCII characters. In addition, most word processing programs attach control codes to the written text, making it necessary to treat the files as binary during transfer. (However, most word processors also allow reading and writing of ASCII text files. Unless both sender and receiver are using the same word processor, it is advisable to transmit documents as ASCII files.)

Some binary bit patterns duplicate a control-Z, which is the ASCII end-of-file marker. This marker is used in most communications programs to identify the completion of a file transfer. If this control code happens to be embedded in a binary file, it can cause the output file to close before the transfer has finished. Fortunately, not all communication applications use a control-Z. Some software packages transfer files by blocks of bits, or convert the file to a different format (such as a hexadecimal ASCII equivalent) before the transfer and then convert back to the original format after transfer. Alternative transfer protocols also solve this problem. The type and amount of data to be transferred must be considered when selecting a transfer protocol.

Data Transfer Rates

The data transfer rate is the speed of a transmission. This rate is either fixed or set by the transmitting device. By changing the hardware switches or the software controls, the transmission speed can be altered easily, within the limitations of the hardware or software. The data transfer rate is measured in bits per second and is usually referred to as baud rate, although the two terms do not have exactly the same meaning.

Bits per second is a measure of the data signaling rate (the average number of bps transferred from the source to the destination). Similarly, baud rate measures the modulation rate (i.e., the systematic change in frequency of a signal to encode and release information). For communications at 600 bps or less, the baud rate will equal the bits per second. However, higher speed transmission devices can sort bits into groups of 2, 3, or 4 bits and then each group is associated with a unique signal value and transmitted. For example, if data bits are divided into groups of 4 bits, a baud rate of 2400 can support 9600 bps, saving transmission time. In this case, the baud rate does not equal the bits per second.

The approximate time required to transfer a file can be calculated easily by Equation 1, given the baud rate and file size (in bytes). Keep in mind that this value is approximate; any retransmissions or collisions will increase the total time required to transfer the file.

$$\text{Time (min)} = \left[\text{[File Size (bytes) x 8 (bits/byte)]/Baud Rate (bps)}\right]/60 \text{ (sec)} \qquad \text{[Eq 1]}$$

Packing Files

A method of increasing the data transfer rate is to "pack" data files for transmission using an archiving/packing program such as PKARC. With PKARC, an archive can contain any number of files, all of which have been compressed or packed in some way. This process results in a 30 to 50 percent reduction in disk storage space and also reduces the file transmission time. Files can be easily "unpacked" using the companion program PKXARC. The recipient of the packed file must be able to unpack the file so that it can be used (in this case, a copy of the PKXARC program must be available).

Transmission Modes

Transmission modes are another important part of data communications. There are three modes of transmission: simplex, half-duplex (HDX), and full-duplex (FDX). The simplex mode is a unidirectional transmission. This mode allows data to be *only* transmitted or *only* received, but not both. Simplex transmission cannot switch directions (Figure 7). An example of simplex mode transmission is commercial (AM/FM) radio.

HDX mode is bidirectional, allowing transmission in either direction, but not simultaneously. Directions can be alternated to provide two-way data flow (Figure 8). Examples of devices using HDX mode are an intercom, speaker phone, and citizen's band (CB) radio.

FDX mode allows data transmission between devices in both directions at once; it requires at least two data channels or wires (Figure 9). A telephone is an example of FDX transmission.

Each byte (or character) of information transmitted or received is "framed" or surrounded by a start bit, a parity bit, and a stop bit. The parity bit follows each byte as part of an error detection technique called "parity checking." When an error occurs, the byte in question is retransmitted. The stop and start bits are used to indicate the beginning and end of a single character or byte.

Asynchronous/Synchronous Communications

This report has so far dealt with asynchronous data communications. Asynchronous (also called start-stop) refers to protocols that synchronize each byte transmitted by using stop and start bits. It is the software protocol that determines the method of data flow.

Synchronous communications are more complex and are usually reserved for mainframe computers. Synchronous communications involve larger "chunks" of data moving at higher speeds usually through a direct (hardwired) connection. Only recently have PCs become capable (through software) of synchronous communications.

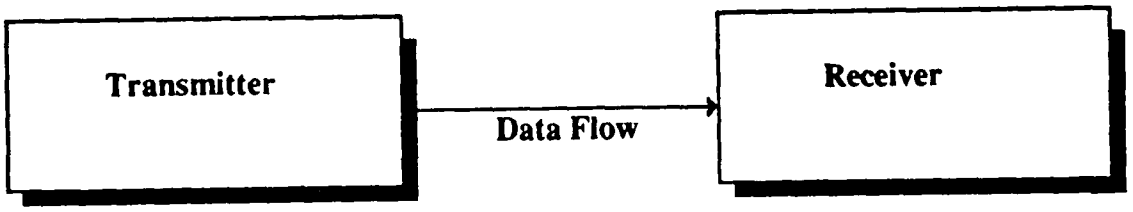


Figure 7. Simplex mode.

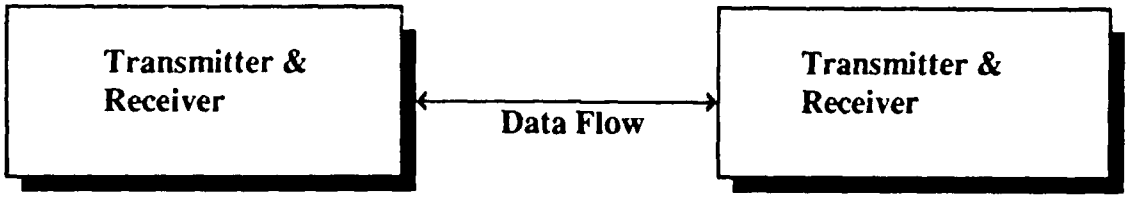


Figure 8. Half-duplex mode.

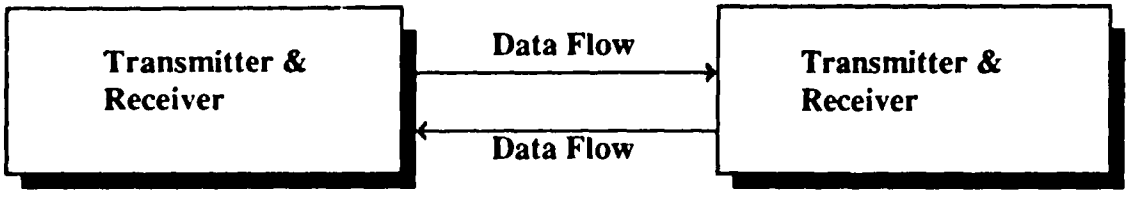


Figure 9. Full-duplex mode.

Both synchronous and asynchronous communications fall under the larger category of serial communications. Data is transmitted and received serially, or in sequence, 1 bit (8 bits/byte) at a time. Serial communications are possible over long distances, but at relatively slow speeds.

Parallel communications involve transmission of an entire byte of information at once. It can take place at high speeds, but over very short distances (typically just a few feet).

Communications Standards

Network integrators face an increasingly complex task in making dissimilar machines communicate over a network. This task is important because many organizations have met their computing needs by buying systems from several different vendors. Vendors sometimes offer proprietary solutions for connecting machines, but these protocols are frequently incompatible. The result is usually that groups of computers can communicate smoothly, but there is no network-wide communication.

The International Standards Organization (ISO), a worldwide standardization agency, began working on a network communications standard in 1977. This standard, the Open Systems Interconnect (OSI) model, is an important step toward eliminating interconnection problems. It provides a mechanism to standardize the set of protocols used in network communications. A protocol is a formal set of rules governing the format, timing, sequencing, and error control of transferred data. Fitting current network protocols onto the OSI model is an excellent way to put these protocols in perspective, promoting both understanding and comparison.

Many different organizations support the ISO model, including the American National Standards Institute (ANSI) and the National Bureau of Standards (NBS). NBS has written a specification called the Government Open Systems Interconnection Profile (GOSIP) which requires vendors to offer OSI-compatible systems when bidding for computer contracts. In addition, there are already many products available, including General Motors' Manufacturing Automation Protocol (MAP) and Boeing's Technical and Office Protocol (TOP).

The OSI Model

The OSI model is a standard, not a protocol. While a protocol defines strict rules that allow communication, the OSI model divides networking into seven logical "layers." The lower layers serve the upper layers, and each layer can communicate only with the layer directly above or below it.

The seven layers of the OSI model (from higher to lower) are (Figure 10):

7. Application. Provides the user with application services such as file transfer, electronic mail, and terminal emulation.

6. Presentation. Presents data to the application layer. Performs format and syntax conversions such as data compression and encryption.

5. Session. Performs services such as validating user identity and authority, translating names and addresses, and establishing or terminating connections.

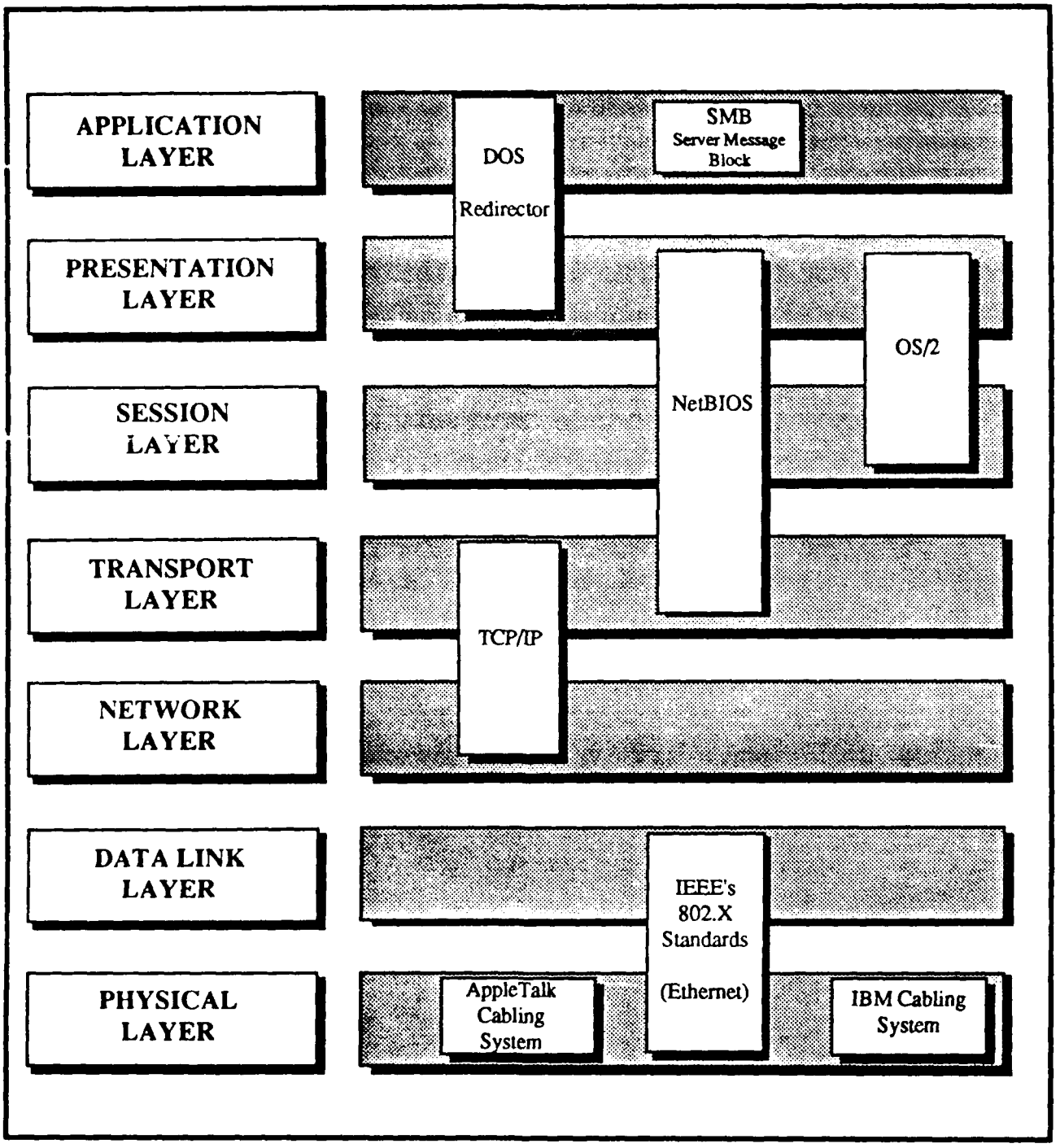


Figure 10. Open Systems Interconnect (OSI) model.

4. Transport. Supervises end-to-end transmission of data. Performs error detection/recovery, flow control, and disassembly/reassembly of session layer messages.

3. Network. Controls the routing and relaying of data as it travels through different networks. Can disassemble and reassemble transport layer messages.

2. Data Link. Formats data into "frames" and transmits it across LAN media. Does transmission error detection and correction. Defines the protocol used to transmit and receive data over the physical media.

1. Physical. Defines the physical means by which data traverses the media, such as cable and connector types, voltage and current levels, and signal modulation techniques.

Ethernet and Token Ring

The Institute of Electrical and Electronics Engineers (IEEE) began working on a networking standard in 1980. Two of the protocols it has established, Ethernet (802.3) and Token Ring (802.5), have become quite popular. Many network vendors offer a wide variety of Ethernet and Token Ring products.

Ethernet and Token Ring are low-level protocols, meaning that they deal with the lower (hardware) levels of the OSI model--specifically, the lowest two layers. They define a solid hardware foundation on which vendors can build when designing network products.

IBM's System Network Architecture

IBM has been developing its own network standard, the System Network Architecture (SNA), which was introduced in 1974. Because IBM has implemented SNA on many of its mainframes, other vendors have offered products to communicate with them. Consequently, SNA has become an industry standard even though some dislike its proprietary nature.

Although SNA has evolved significantly, its structure is still influenced by its mainframe origins. Thus, the networking layers that SNA defines differ significantly from those of the OSI model and are not interchangeable. However, it is important to note that while SNA groups protocols and services differently than OSI, it has the potential to cover the same full range of networking capabilities.

Error-Checking Protocols

A long-distance telephone connection may sound unchanging to your ear; however, its electrical characteristics vary constantly. Many conditions can affect the quality of telephone connections, from a wire swaying in the wind to the way in which calls are rerouted automatically through several switches when direct circuits are full.

When "noise" is introduced into a telephone line through any of these events, computer communications may begin to deteriorate when modems are used. Modems have two challenges: first, to not interpret such changes as data, and second, to maintain the quality of the line at a high enough standard to support its use for high-speed transmission.

Higher speed modems often have built-in active equalizers that compensate for changes in the line. Some also have a fallback option: when a connection becomes too poor to support a high data rate, the modem automatically falls back to a lower rate that is less sensitive to line vagaries. However, modem hardware alone is not sufficient to ensure error-free data transmission.

Error-checking protocols are schemes designed to ensure error-free data transmission. Most error-checking protocols detect discrepancies in set-length groups of characters, called "blocks," which are transmitted through digital systems. Each block consists of a predetermined number of bytes. For error checking (or error detection), the bytes are summed and a value representing the sum is transmitted along with the block of data.

In the computer on the receiving end, the bytes in the block are again summed and compared with the transmitted sum; any difference reflects an error. If a potential error is noted, the recipient's computer will transmit a message to the transmitting computer requesting that the data be retransmitted.

Many different error-checking protocols are used, such as Transmission Control Protocol/Internet Protocol (TCP/IP), Xmodem, Kermit, Microcom Network Protocol (MNP), and X.25. The simplest, most commonly used protocol in PC communications is Xmodem. MNP and Kermit are also popular. The Ymodem protocol is an extension of Xmodem, and X.PC is a newer protocol that was developed for use with packet-switched public data networks (PDN).

In general, the more rigorously a protocol protects against errors, the more slowly it works. Both ends of a communications channel must be equipped to handle the same error-checking protocol (or lack of one) for successful communications to be established.

TCP/IP. In the early 1970s the Department of Defense Advanced Research Projects Agency developed TCP/IP. It is currently popular as a proven nonproprietary standard supported by many vendors.

TCP/IP fits quite well into the OSI model and provides a good working example of the benefits gained by doing this. The TCP corresponds to the Transport (fourth) layer of the ISO model, and the IP to the Network (third) layer. In practice, TCP's job is to accept data from the application, divide it into blocks that IP can handle, then add control information so that everything can be reassembled at the other end. It then passes these blocks, along with a destination, on to IP. IP's sole task is to send the block of data to its destination. Because TCP and IP are separate and independent layers, one can be modified or improved without affecting the other. Furthermore, creating network applications is easier because of TCP's consistent networking services.

TCP/IP defines several protocols for the Application (seventh) layer, including terminal emulation, file transfer, and electronic mail. Although TCP/IP does not cover layers one, two, five, and six of the OSI model, the layers that it does cover make it a truly powerful standard. Moreover, the fact that TCP/IP matches the OSI model, but does not cover all its layers, can sometimes be a plus. For example, it can simplify the implementation of TCP/IP on existing protocols that cover only the Physical and Link layers, such as Ethernet and Token Ring.

X.25. The International Telegraph and Telephone Consultative Committee (CCITT), in collaboration with the ISO, has created a protocol called X.25. It covers the following layers of the OSI model: Data Link and Network. X.25 is a connection-oriented protocol. It establishes a connection between machines, then sends the data

sequentially over that connection. In contrast, TCP/IP is a connectionless protocol. The data it sends can travel many different paths and does not have to arrive in the order it was sent.

Connection vs. connectionless differences may introduce problems for system integrators. For example, as X.25 becomes more popular, some users might need to implement TCP over the X.25 protocol instead of the IP. The TCP normally works with the IP, which means it must ensure that data is reassembled in the correct order. However, since X.25 is a connection-oriented protocol, it already delivers data in the correct order. The resulting redundancy is only one of the many problems facing ISO as it strives to create a worldwide networking standard.

Xmodem. This protocol provides excellent file transfer and error detection for communications on high-quality lines. Xmodem forms data into 128-byte packets and inserts a single "checksum" error detection byte that is the sum of all ASCII values in the data block (packet). As a result of this simple design, Xmodem has efficient data transfer and very low overhead (in terms of time spent transferring data). However, this design also means its use can be very limited.

Xmodem's design requires that it operate at relatively slow transfer speeds (2400 bps or less) and on high-quality lines. In environments where many errors in transmission are occurring, Xmodem begins to slow considerably; also, its error-checking procedures are not infallible. For example, as a result of its single checksum error detection routine, it is possible for multiple bit changes to result in an undetected transmission error. Also, Xmodem uses a single character to acknowledge the status of a block transfer. Line noise can damage this character and, consequently, cause unnecessary retransmissions or abort the transmission altogether.

Another major limitation of Xmodem is its stringent communications parameter requirement of 8 data bits. Many PDNs and large computer systems operate with only 7 data bits; as a result, Xmodem cannot be used in conjunction with these networks or systems.

Xmodem is inherently an HDX protocol. Since packet numbers cannot be included with the data block, the sending computer must wait for an acknowledgment or negative-acknowledgment (ACK/NAK) from the receiver before sending the next block. Thus, Xmodem has limited use with packet-switched PDNs. Packet-switched networks such as Tymnet involve considerable overhead in terms of time needed for packet formation and submission to the network--often as much as 2 or 3 seconds per packet.

In the HDX mode of operation, a block of data is passed from the sender to the PDN, formed into packets suitable for transmission on the PDN, and then passed to an exit station on the PDN, reformed to the original block and passed on to the recipient. The recipient will then return an acknowledgment character through a similar path. Without some mechanism for sending additional blocks of data prior to receiving an acknowledgment from the recipient, the additional overhead of the PDN packet formation is added to every block of data transmitted and communications rates slow to a crawl. This delay can reduce transmission speeds as much as three or four times below the rated throughput of the data communication equipment involved.

Ymodem. Ymodem was developed after Xmodem and improves on several of Xmodem's shortcomings. First, Ymodem is capable of faster, more efficient transfer because it uses 1024-byte data blocks (it will automatically reduce to 128 bytes if line

quality is bad). Rather than the single-character abort signal used in Xmodem, a two-character code is used to stop transmissions in Ymodem. Thus, inadvertent aborts occur less frequently. Finally, Ymodem transmits file information at the beginning of each file transfer, allowing this information to be maintained and batch file transfers to be performed (not possible with Xmodem).

Two of the Xmodem limitations are still present in Ymodem: (1) it operates in an 8-bit world and (2) it is incapable of FDX operation.

Kermit. This protocol was established specifically for communication between dissimilar machines over high-speed lines and long distances. Kermit supports three types of error checking: 6-bit checksum, as used in Xmodem, 12-bit checksum, and CRC-16. Depending on line conditions and data requirements, Kermit users can select the type of error detection required. In the case of CRC-16, error-free transmission is possible. Kermit, capable of using both 7- and 8-bit data ports, can move groups of files. It includes packet information with each data block and can therefore send a stream of data blocks through a PDN with a reasonable turnaround time.

Kermit's enhanced capabilities bring some additional transmission overhead. In fact, for most PC-to-PC communications on good lines at speeds of 2400 bps or less, Kermit transmission will be considerably slower than a similar Xmodem transfer--perhaps as much as 40 percent slower. However, if you are transmitting vital financial data or a critical binary file, the added reliability of Kermit's error-checking success may be worth the additional overhead.

MNP and X.PC. The final two protocols discussed here are MNP and X.PC. Both protocols provide features very similar to Kermit with some slight improvements. Both can be implemented in hardware or software; however, their major difference lies in where they are implemented best. MNP is better suited for hardware/firmware implementations and X.PC is more adaptable to software implementation.

MNP provides some throughput gain (about 10 percent) by simulating a synchronous data stream. When modems at both ends of a communications link support MNP, the protocol strips off the start and stop bytes associated with each data byte and forms packets of continuous data that are terminated with a CRC error detection block.

The X.PC protocol has the advantage of being designed for use in conjunction with PDNs. X.PC converts all transmitted data into packets that include destination addresses and CRC error detection information. The inclusion of addresses in the data packets allows multiple communications sessions (up to 15 separate sessions).

Potential Communications Problems Beyond Program Control

Procedures developed for communications between microcomputer systems and mainframes typically are designed to provide specific responses in a predictable environment. Specifications for programs of this type require that the programmer evaluate the communication responses that may occur and provide programmed responses for the various circumstances that will be encountered during sessions. For example, if the task involves collecting information of a specific type from a mainframe data base, the programmer must provide alternative program sequences for times when the data base is in use by another person or program, procedures to handle transfers that exceed the capacity of the local microcomputer disk space, and procedures to report the occurrence of a "null" set of data when no cases are found on the remote system. The programmer must predict these outcomes and design the system to handle them without user intervention.

Unfortunately for the programmer, several circumstances are not predictable when creating an application-specific communication system. This section discusses unpredictable factors and provides suggestions for solving the related problems. In most cases, solutions for unpredictable problems require some type of recovery sequence. The results of completely unpredictable events require that the program sequence be stepped back and restarted at a convenient point instead of merely branching to a different sequence of input and output steps. Unpredictable problems include poor quality communication lines, power failures, and failures on the mainframe system.

Line Noise

Communication between two computers often is carried over telephone lines that are leased from a commercial carrier. These lines may be acceptable for voice communication, but not for data transfer. Line noise includes background pops and clicks classified as static. At many sites, static will occur on telephone lines during rainy weather or other wet conditions. Old telephone equipment may cause noise on lines because the switching systems are less precise than the digital switching systems now being installed. For voice communication, users can overcome a problem by restarting the conversation on a new line or just talking louder. Digital computers are not as adaptable and may require that the call be abandoned until the system is repaired.

In addition to transmission problems, it is also possible for data content to be modified as a result of line noise. Thus, data may differ between the sending and receiving computers because character transformations have occurred due to line variations.

A similar effect can occur when a telephone is equipped with a call-waiting notification system that sounds a mechanical click on the line when another call is placed to the number. These clicks are hardly noticeable during a voice session, but a mainframe system will presume that the line has been dropped by the remote system and hang up the telephone. Similarly, background clicks may cause the system at the other end to drop the telephone line at a random time in the program sequence.

When these conditions cause disconnection of a telephone line, the microcomputer system will be in an unknown state. As a result, if the continuing operation of the microcomputer system depends on a triggering character or sequence from the mainframe system, the program will potentially stop operating until the user intervenes.

Line loss also can occur at many points in the communication process. Typically, a local call in a community passes from the user's telephone or modem through a line to a central switching office. The signal is switched to another line and transferred back to the recipient's hardware. This process occurs on most systems even when the two instruments (modems or telephones) are located in the same room. In larger cities or when a call is long-distance, the number of switches and relays increases. Failure at any point in the process will result in line loss.

Power Outages

Power failure at either end of the communicating system results in the loss of communication lines. In addition, it usually shuts down the computer system. Normally, systems equipped with an uninterruptable power supply (UPS) proceed to back users out and cleanly execute shutdown procedures after the main power lines fail; however, even these systems may not continue to operate when a general power failure occurs.

Power failures at either end of the communication line can cause the modem to fail and lose the communication line even if the telephone system is powered separately and the line continues to work. Power failures at any of the switching stations also may cause line loss.

Power failures on the microcomputer side will be easily observable by the user and a restart of operations will be indicated (some automatic procedures may be programmed to restart cleanly after a local power failure). If the power failure occurs at any other location in the system, the program may not recover easily and the characteristics of the failure will be identical to those of a dropped telephone line.

Mainframe Failures

Failure of the mainframe system is another unpredictable event for application-specific communication programs. There are several ways that the mainframe system can fail. Most mainframe systems consist of a number of separate hardware components and peripherals, all of which may fail, resulting in the total failure of the mainframe system. In addition, since mainframe systems are typically multiuser, failures can be induced by other users since the operating system is complex on such systems. User software is another point of failure.

If failures occur on the mainframe system, depending on their location and the status of the user software on the microcomputer system at the time, the failure may be similar to that caused by random line loss or power failure.

The most insidious type of mainframe problem for the microcomputer programmer is the software failure. If the computer system stops sending expected information that is being used to trigger operations, it is possible for the user program to become "lost on the mainframe." For example, on one mainframe system, management will occasionally force all users through a different log-in sequence to bring attention to system announcements. Although this process may not be seen as a failure, it is possible that the sequence is neither anticipated by the user nor does it contain triggering words that cause program actions. As a result of this small change, anticipated information is not received and the program may hang up, abort, or loop infinitely.

Solutions

To counter unanticipated problems, programs that must guarantee the accuracy of transferred data use special protocols that test the data at each end using checksums and error correction algorithms to be sure that the data sent is identical to that received. Parity bits in each character may be used on some systems to indicate the occurrence of a bad transfer.

Communications with a mainframe system may provide correction codes, parity, and checksums compatible with microcomputer-based communication programs. As a result, a programmer can assure users that data transferred is correct. However, the setup sequences for data transfers such as mainframe program startup and log-in are not typically protected by the use of checksums, and rarely is a technique provided for the automatic correction of data sent from the mainframe as part of the operating system's user interface.

The programmer for an application-specific communication package must contend with the possibility that line errors on the telephone line will change data to a form that is not predictable or that the mainframe may even drop the line at unexpected times.

Protection against this type of problem requires that the programmer automatically track the progress of the communication program. For example, if programs are expecting feedback from the mainframe in a particular form, that form may never appear as a result of the line noise. If the programmer incorporates a "wait loop" to continually check all input for the existence of a particular phrase, the program may become caught in that loop until it is terminated manually.

Loss of a communications line, power or system operation on the mainframe side, or a "standard" session on the mainframe side all result in the same basic situation. The computer at the other end no longer responds to the microcomputer program. It is important for the application to detect this condition and act on it.

Software should be protected by the incorporation of time-out or count-out sequences that terminate the program after a reasonable number of attempts. The programmer can continue to cover the situation by providing a feature that tries to execute an operation repeatedly after timed intervals, and then eventually issuing a report to the user that the program has failed and must be restarted later. Since line noise is not predictable and may occur in the middle of a process, it will be necessary to make the user always restart a process at the beginning or to provide "restart" options. Restart operations typically back up the programmed process to the last recorded successful operation and restart at that point. The record of program progress is written to a disk at the local system.

If communication software supports special "interrupt" actions on specific events, it may be possible for the programmer to execute code that reconnects and restarts operations after the line is lost for any reason. Similarly, time-out interrupt sequences can protect application programs from long waits for system response after mainframe system failures. If an interrupt is not provided, some type of timer sequence should be incorporated to protect against excessive nonproductive connect time due to unanticipated errors.

Recovery Mechanisms

In a distributive processing communications system, it may be necessary to transfer one or more files in a single session. The reliability of the communications medium directly affects the probability that errors will occur during transmission. Time and file size are also factors in the error probability formula. The probability of encountering unrecoverable errors, such as losing the communications line, increases proportionally with the time and size of the file. Since unpredictable communications errors are a fact of life, recovery mechanisms must be built into transparent systems. Since there is no point in retransferring files that have already been transferred successfully, a recovery mechanism must record the success or failure of each file transmission and then, on the next restart of the system, transfer only those which have failed.

Recovery mechanisms can be difficult to design and incorporate. The key to a successful recovery scheme is anticipating all possible errors. This task is highly challenging. To recover, the system must recognize all possible results of the error that occurred. Some errors are difficult to reproduce, making it hard to determine the variations in responses.

The severity of the error also influences the design of the mechanism. Some errors affect the status of only a single file, while others, such as line loss, terminate the process completely. The recovery mechanism needs to solve both types of problem.

Below is an example recovery mechanism for an update distribution system designed to function in conjunction with the Ontyme mail system via Tymnet.

Say that a list of all files to be downloaded (download queue) is created on the micro by comparing micro and Ontyme version numbers associated with each module. After each file is transmitted, its status is checked for success or failure. The micro maintains a list file of all unsuccessful transmissions (Figure 11). At the end of the process, the user is informed that errors have occurred and the target program has not been completely updated. The next time the system is invoked, the list file is checked for entries. If they exist, these are added at the beginning of the download queue when the new list is created. The process begins again.

If a nonrecoverable interruption occurs, such as line loss or the user aborting the transmission, the system transfers all remaining file entries from the download queue to the failed file list for recovery next time.

There are other ways errors can be recovered; this is a single example of such a procedure. The mechanism used will depend greatly on the system and its purpose.

Prepackaged Programs/Application-Specific Programs

General communications programs, such as Crosstalk, Smartcom, Procomm, and Smartterm, offer the user a varied set of features. Many of these are highly user-friendly and easy to learn, but some are not. There are both advantages and disadvantages to using general communications programs. The main advantage is that the program functions are available to the user in any sequence at all times. For example, a user may wish to run a mainframe program as well as check mail in a single session. However, the user is responsible for entering all commands to control both mainframe and microcomputer. Transferring a single file requires access rights to the appropriate storage areas on both machines. This option may or may not be available to an

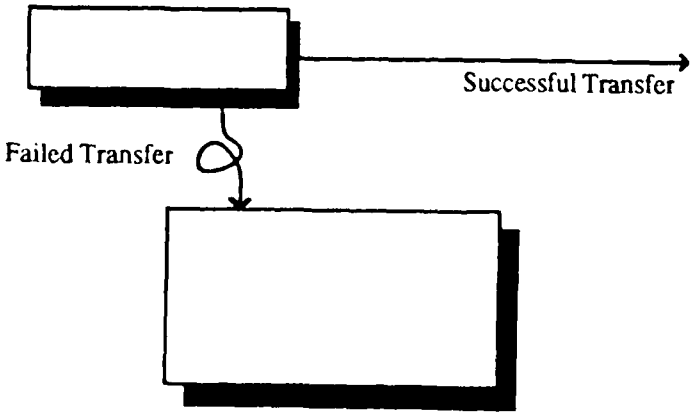


Figure 11. Recovery mechanism.

individual user. The PAX system, for example, restricts users to specific areas. To use a prewritten program for file transmission, the user must know:

- The logon procedure
- How to access the appropriate mainframe storage area
- Which file transfer protocols are available on the mainframe
- How to start the chosen protocol
- Which files need to be uploaded/downloaded
- Where the files are located
- The targeted storage location
- What to do if an error occurs.

Many of the better communications packages offer a scripting (or macro) language to help automate some simple exchanges between mainframe and micro. A common use for this scripting is the logon procedure, in which the commands that will be needed to start the system are stored in sequence in a script file for later use. Often, creating a script is like writing a small program. In addition, the scripting is seldom extensive enough to handle complicated tasks. Error recovery mechanisms and multiple files with variable names would be beyond the capability of most scripting programs.

3 DEVELOPING APPLICATIONS-SPECIFIC TRANSPARENT COMMUNICATIONS

After studying all the equipment and configurations available for creating transparent communications systems, it was decided to develop applications-specific software to transfer files exclusively from the Civil Works management system, ABS, and the Army-wide program, ECONPACK. Application-specific programs are written for the sole purpose of updating a specified set of program files or moving data from one machine to another. They can provide the user with a completely transparent system.

Adhering to the OSI Model

A good application-specific communications system will adhere to the OSI model. The system will span levels five (Session) through seven (Application) of the model. Although the system may be specific to a single application, it should not be tied to a single mainframe system. The way to avoid this limitation is to make the system modular. The user interface is a module that can be constant to any mainframe. In contrast, the command module to control the mainframe will vary according to the system used. Figure 12 shows the Main Menu for the ABS. Each option represents a separate module that is executed independently from the others.

Automated Budgeting Distribution System

The ABS is a compiled dBase III-Plus program designed to assist in the budgeting process for Civil Works projects. The program, used by 40 Civil Works districts nationwide, provides calculations and reports required by Headquarters, U.S. Army Corps of Engineers (HQUSACE). Appendix A contains examples of the screens used to enter and check data needed by the districts. These menus are accessed through the editor module of the system.

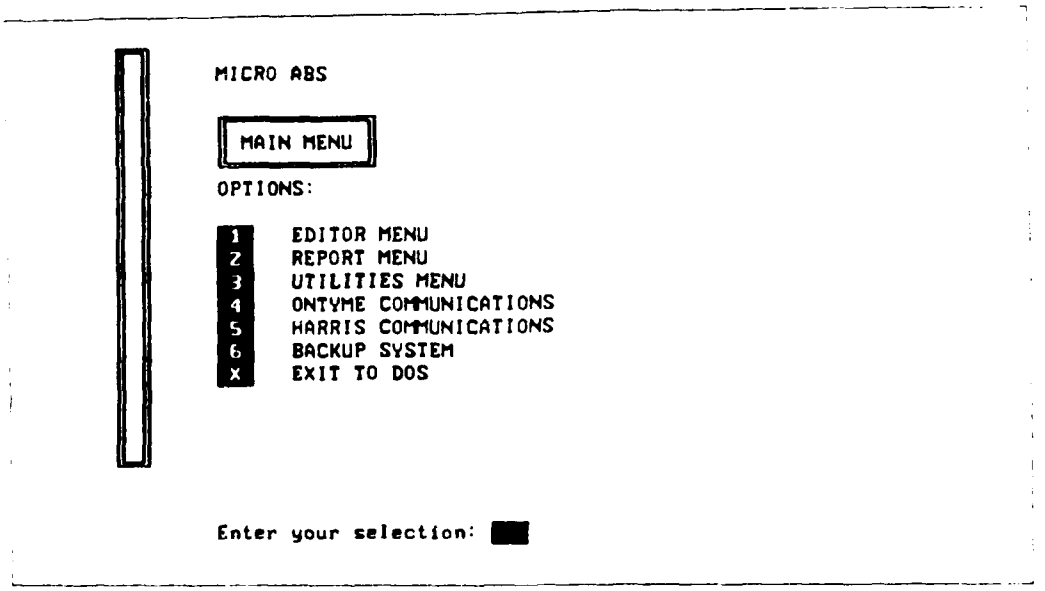


Figure 12. System modules for the Automated Budgeting System.

During program development, the need for a mechanism to send updates to users became increasingly apparent. The cost for manpower, disk media, and mailing charges was potentially very high. The solution to be developed, an electronic distribution system, was named the Automated Budgeting Distribution System (ABDS).

PC System Specifications

The first consideration was to determine the mainframe configuration model. Since all users already had access to Ontyme, it was selected. Ontyme is an IBM mainframe located in Dallas, TX. It is accessed through a worldwide broadband network known as Tymnet and administered by McDonnell Douglas Corporation. Once the mainframe configuration model was determined, the program design could begin to take shape. Mantra, a forth generation communications language created by McDonnell Douglas,* was chosen for its ability to easily handle the communications aspects of the system. X.PC and Xmodem were the file transfer protocols available for use on Ontyme. The other program specifications were:

- DOS Level: 2.0 and up.
- Hardware: IBM-compatible PC with minimum 640K random access memory (RAM), hard disk, and either Hayes-compatible modem or serial port for communications.

There are two sets of users for the ABDS, the user in the field and the distribution administrators. A detailed model was developed to describe the two related system functions. The user function has only one feature--downloading new program updates. Figure 13 shows the model of the user function. The administrative function has several features:

- Upload to Ontyme (Figure 14)
- Download from Ontyme (same as user function)
- Update distribution list
- View distribution list
- Edit user Id list.

Connection

Since ABDS is a communications system, the first operation performed is making the link with Ontyme. Before the actual link can be made, it is necessary to establish the communications parameters including modem type, port, baud rate, parity, duplex, and telephone number. Appendix B contains the documentation describing how to do this procedure. Once established, they need not be set again unless the hardware changes.

*Mantra can be obtained by contacting the sales representative at McDonnell Douglas Information Systems Group, Suite 200, 2070 Chain Bridge Road, Vienna, VA 22180; telephone (703) 734-0088.

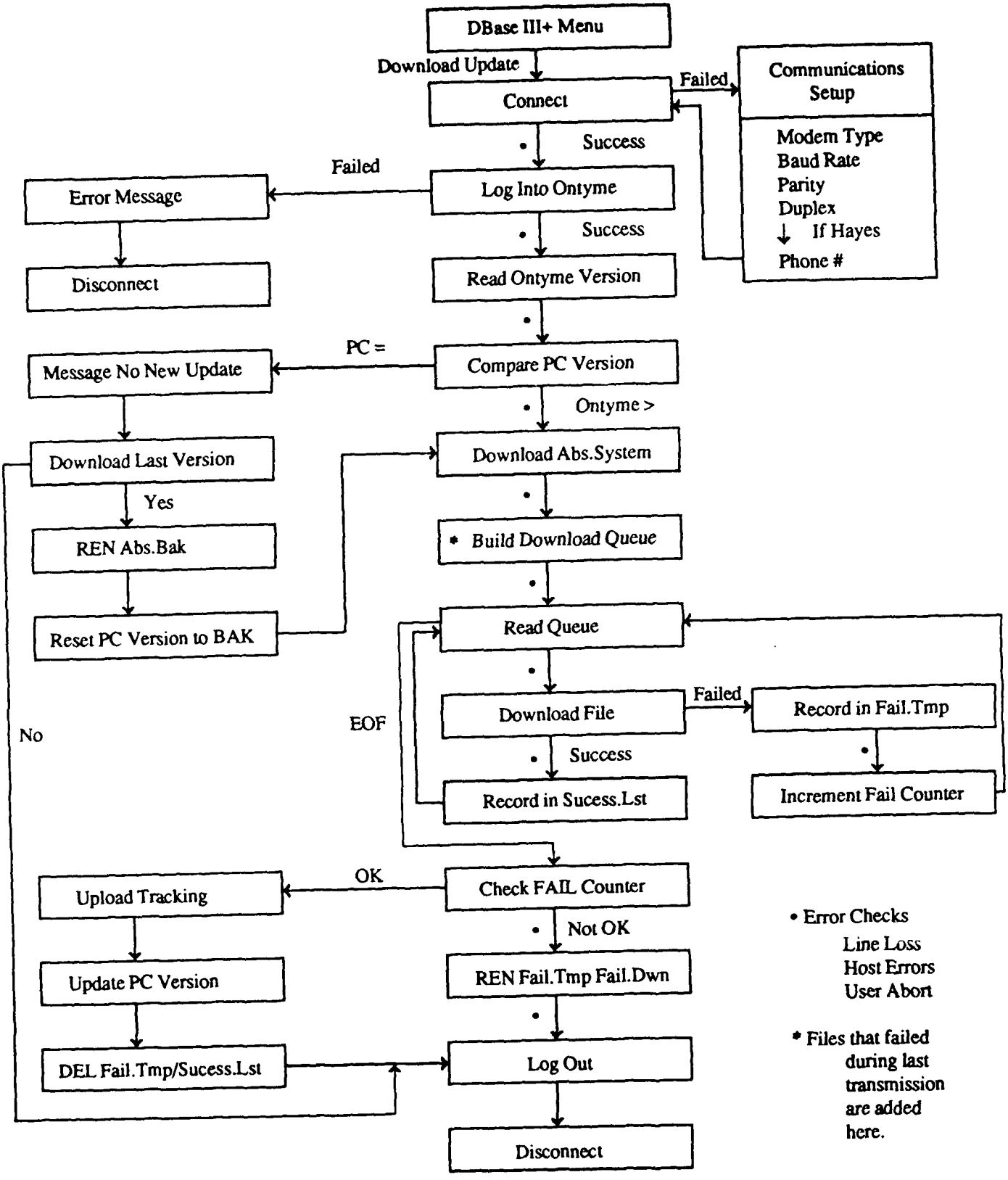


Figure 13. ABDS user function download feature.

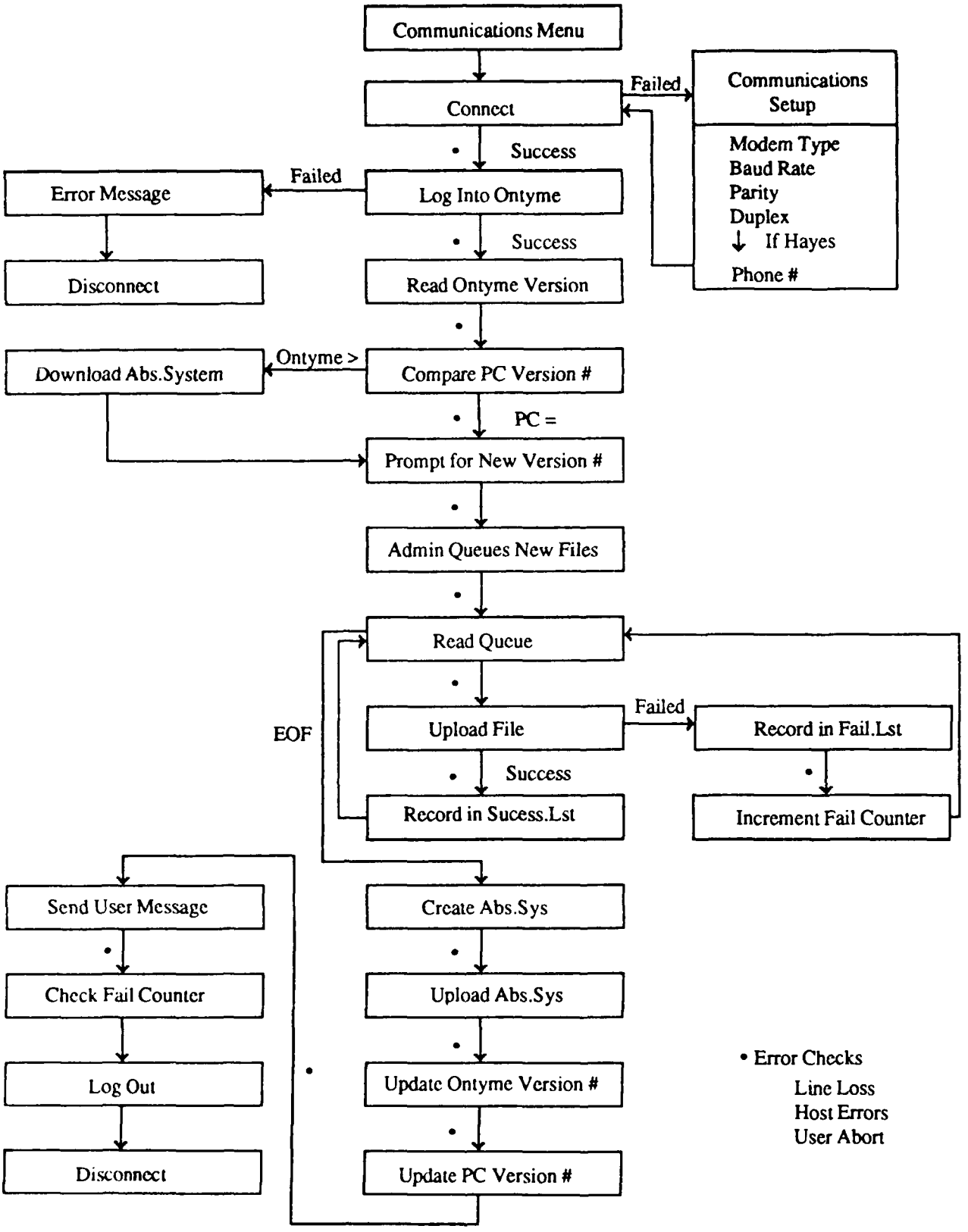


Figure 14. ABDS administrative function upload feature.

The Tymnet Engine is a gateway into the Tymnet Network. There are three methods of access to Tymnet (Figure 15):

- Telephone modem
- LAN or data switch
- Hardwired directly into Tymnet.

Modem Option. ABDS was designed to use a Hayes-compatible modem. The modem commands take the form of the Hayes AT command set.

Network Option. Many networks and data switches are in use by the Civil Works districts. Automating the command sequence is impossible without having access to each of the various network configurations. The ABDS, therefore, provides the user with a "Watch Me" mode when the access method is network. The Watch Me mode acts as an action recorder. After the user teaches the computer the sequence to access Ontyme, it is played back automatically each time the user accesses ABDS through the network.

Hardwire Option. Very few sites have a Tymnet Engine available for direct wiring.

Updates

ABDS tracks the files updated. Downloading an entire dBase system every time a small change is made to the code would waste both time and money. In addition, the longer it takes to download a file, the higher the probability of failure. ABDS, therefore,

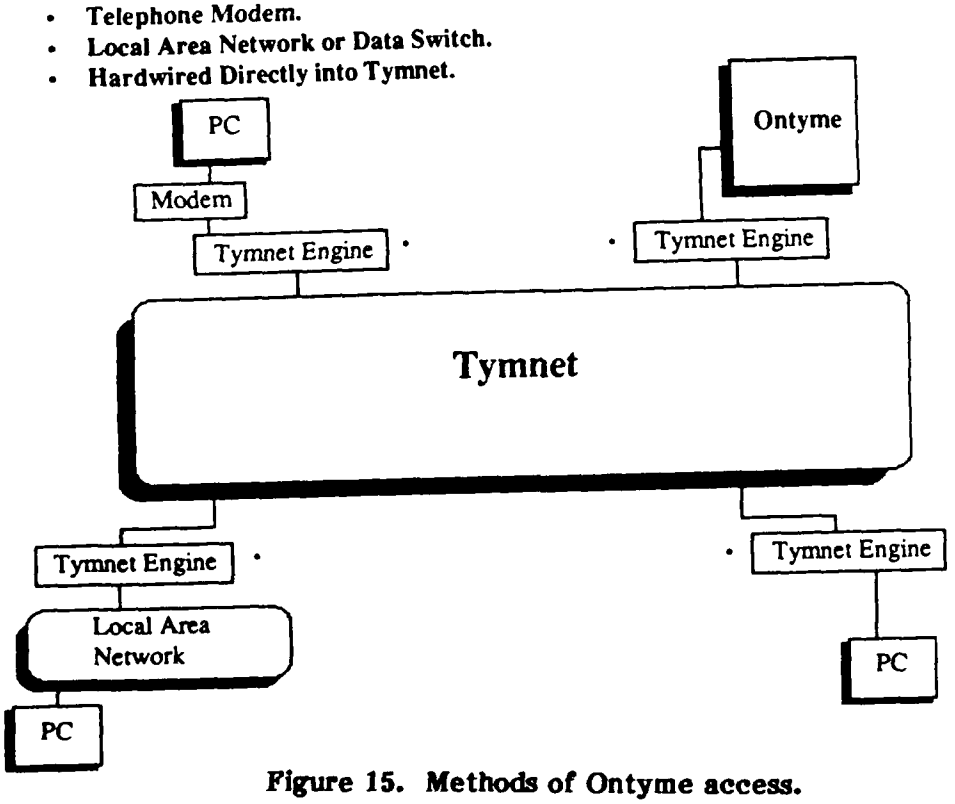


Figure 15. Methods of Ontyme access.

transfers only those files which have been changed since the last user update. A complex scheme of version and release numbers was devised to accomplish this task.

Each program module has an associated version number (version.release). All necessary information about each updated module (file name, size, date created, directory location, and version number) is stored in the ASCII file ABS.SYS (Table 1).

In addition to information stored about the individual modules of the program, ABDS stores data about the program as a whole. The ASCII file ABS.VSN records district ID, user last name, and system version number. The files ABS.SYS and ABS.VSN are critical to the upload, download, and tracking aspects of ABDS.

Downloading Program Updates

When the download feature is selected by the user or administrator and connection to Ontyme is achieved, the Ontyme version number is compared with the version.release number stored in ABS.VSN. If the Ontyme version or release number is higher than the PC version or release number, ABS.SYS, stored on Ontyme, is downloaded. The download queue is built by comparing the system version.release number to the number associated with each module stored in the ABS.SYS file. The system then begins to download the updated modules and stores them in the appropriate subdirectories on the PC. Files are transferred using Xmodem protocol (layered over X.PC protocol for users with telephone modems).

Uploading New Program Updates

It is the system administrator's responsibility to place the new program updates on Ontyme for users to retrieve. The upload feature on the administrator's menu provides this capability. Its function is similar to the download feature described above. Once connection is acheived, the version.release numbers are compared. If the Ontyme numbers are higher than the PC numbers, ABS.SYS is downloaded. Since it is possible for updates to be made from more than one PC, it is very important that the administrative system work with the most up-to-date system files. The system then displays the old version.release number and prompts the administrator for the new number. The administrator builds the upload queue by either entering or selecting the file name and directory for each module. Once the queue is complete, the system takes over, uploading

Table 1

Example of Storage File (ABS.SYS) Content

File Name	Size	Date	Directory	Version
SETUP	74	1-05-88	\ABS\COM	V02.02
BAUD	30	6-03-88	\ABS\COM	V02.01
COMLINE	23	6-03-88	\ABS\COM	V01.11
DISTRICT.TBL	651	1-05-88	\ABS\COM	V02.02

each file and storing it on Ontyme. The final steps are to update the system file (ABS.SYS) to reflect the new version.release numbers, upload it to Ontyme, update the Ontyme and PC system version.release numbers, and notify users of the update. As in the download feature, file transfer is via Xmodem protocol (layered over X.PC protocol for telephone modems).

File Recovery Mechanism

When dealing with file transmission, there is always a possibility that the file will not reach its destination intact. Using error-checking protocols such as Xmodem and X.PC ensures that the packet is retransmitted when errors occur. There are times, however, that the packet cannot be transmitted successfully. Any error-checking protocol has a limit to the number of retransmissions it will perform. The communications system therefore must provide a safety net to catch files that fail. Recovery mechanisms must cover all possible failures, from line loss and user abort to single file failure.

The ABDS records all failed files in a temporary listing file and uses a counter to keep track of them. When the transmission process (uploading or downloading) is complete, the counter is checked. If it is higher than zero, the temporary file is renamed to indicate a failed condition and the user is issued a message reflecting the failure. Next time the system is run, the files recorded in the failed file listing are added to the beginning of the new transmission queue. The process is repeated until all modules have been transferred successfully.

Tracking Users

In a system such as ABDS, it is important that all users have the most up-to-date modules to process their data. For the administrators monitoring the project, it is important to know which districts are not maintaining their systems. To handle these functions, a tracking report was implemented in ABDS.

ECONPACK Data File Transfer System

ECONPACK is a mainframe/micro economic analysis package developed by USACERL. It is an example of a system that uses distributive processing; the micro version of the system parallels the mainframe version. This configuration allows the user a choice of operating environments. The data, however, must still be submitted to the mainframe for upward reporting. To provide the micro user with a painless method to transfer the data files, a transparent communications system specific to ECONPACK was developed. It is called the ECONPACK Communications System.

PC System Specifications

As in the ABDS, the first consideration was the mainframe configuration model. Since the mainframe version of the system resides on Tymshare, an IBM machine located on the east coast with access by all users, this model was the only logical choice. Tymshare is the host mainframe for the PAX system, which supports several widely used applications.

System specifications are:

- Language: Mantra
- File transfer protocol: Mate
- DOS Level: 2.0 and up
- Hardware: IBM-compatible PC with minimum 640K RAM, hard disk, and either Hayes-compatible modem or serial port for communications.

The ECONPACK Communications System functions much like ABDS. Therefore, the programming concerns and design resemble those of ABDS. ECONPACK, however, does not have the updating feature, but involves mainly uploading and downloading data files.

Connection

The first step in any communications system is to link the two machines. The communications parameters modem type, port, baud rate, parity, duplex, and telephone number must be established and stored. This step need be done only once unless the hardware changes.

Access to Tymnet is by the same three options described earlier (modem, network, or direct hardwiring). Upon connection to the Tymnet Engine, a prompt is displayed: "Please log in:". Since Tymnet supports several hosts, the login sequence entered determines packet destination and the host accessed (Figure 16).

Modem Option

Because the ECONPACK Communications System is used worldwide, variations between telephone services offered in different countries had to be considered. For example, in Europe, the format of telephone numbers is quite different than in the United States. The U.S. number format is uniform, having an area code (3), Exchange (3), and Number (4) in the sequence XXX - XXX - XXXX. No matter what destination is being dialed, the format is the same. In Europe, however, the format varies by county, city, and country. The general number format is country code (1 or 2), city code (1 or 2), exchange/number (6 or 7) in the sequence XX - XX - XXXXXXX. The programming difficulty arises due to the variable number of digits in each code. The program must provide for the maximum number of digits to accommodate these differences.

In addition to communications line and access differences, some countries have strict regulations on data communications equipment. For example, in West Germany, only Bundestpost-approved modems can be used legally over the telephone lines. Many of these modems are manual-dial type, making completely transparent communications impossibile.

PAX Menus

Once the user is logged into PAX, the program must deal with the PAX menu system. The PAX menu includes variable series of announcements and messages followed by the user menu. The menu also varies among users. Designing for an automated system to pass through this maze is a difficult process. One solution is to bypass the menu system completely. Because the system is unattended, bypassing the menu system

does not detract from user functionality. To incorporate this feature, USACERL created a special project code in conjunction with the USACE Huntsville Division and McDonnell Douglas. This project code, embedded in the program, is entered into the PAX system during the logon sequence. This step starts an EXEC(ute) process that bypasses the menuing system and links the user with the appropriate disk and program areas. The system is now at command level and is ready to begin the requested file transfer.

File Transfer

Data files are uploaded and downloaded using the Xmodem protocol. If data files are ASCII text, transfer time and disk space can be reduced by using a compression routine before transmission. However, ECONPACK does not use compression due to the lack of a corresponding expansion routine on the mainframe.

Recovery Mechanisms

Normally, transparent communications systems require extensive recovery mechanisms to handle problems such as line loss, mainframe failures, and line noise. However, since ECONPACK transfers only one file per connection, there is no need for such a complex recovery system. When a problem occurs, the user need only restart the system to recover the file.

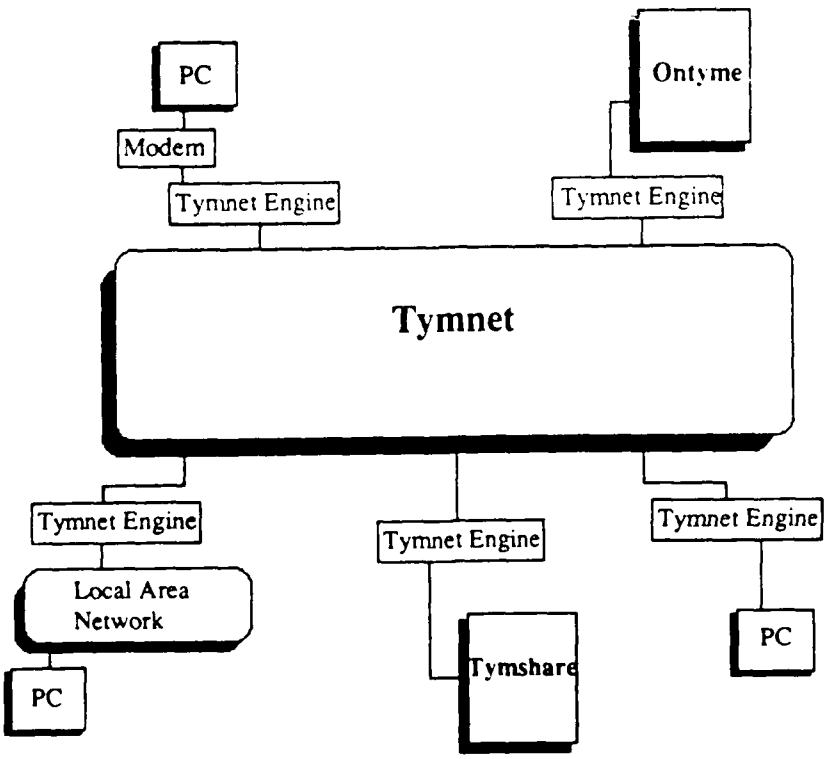


Figure 16. Methods of accessing PAX.

4 CONCLUSION

Transparent communications systems have been developed for transferring data from PC to mainframe in the Civil Works automated management system ABS and in the Army-wide system ECONPACK. The ABDS is a distribution (or update) system that allows uploading by system administrators and downloading from the field. The ECONPACK Communications System allows data transfer of single files to or from the mainframe.

File updating and transfer via transparent communications saves considerable money and time by completing the operations automatically with very little user intervention. Previous methods of completing these functions included mailing updates to users on floppy disks and logging onto the mainframe to transfer files using complex programming procedures.

It is possible that distributive processing systems like those developed in this study could evolve into a stand-alone package capable of transferring all types of information in a variety of environments. Extensive research would be needed to configure such a system, but the potential impact on mainframe data transfer is great both in Government and industry.

CITED REFERENCES

Neathammer, R.D., and J.D. McLean, *Economics Analysis: Description and Methods*, Technical Report P-89/08/ADA204264, (U.S. Army Construction Engineering Research Laboratory, December 1988).

UNCITED REFERENCES

Campbell, J., *C Programmer's Guide to Serial Communications* (Howard W. Sam and Co., 1987).

Connor, N., "TCP/IP Training," *Lan Magazine* (Feburary 1988).

"Data Communications Glossary," *Data Communications*, Vol 17, No. 3 (March 1988).

Derfler, F. J., "Networking Acronyms and Buzzwords," *PC Magazine*, Vol 7, No. 11 (June 14, 1988).

Jordan, L., and B. Churchill, *Communications and Networking for the IBM PC and Compatibles* (Brady Books, 1987).

Lefkon, D., "Seven-Layer Cake Made Simple," *Lan Magazine* (Feburary 1987).

Roux, E., "Protocols'/OSI's Final Frontier: The Application Layer," *Data Communications*, Vol 17, No. 1 (January 1988).

Schwartz, M., *Telecommunication Networks Protocols, Modeling and Analysis* (Addison-Wesley, May 1987).

Seyer, M. D., *RS-232 Made Easy* (Prentice-Hall, 1984).

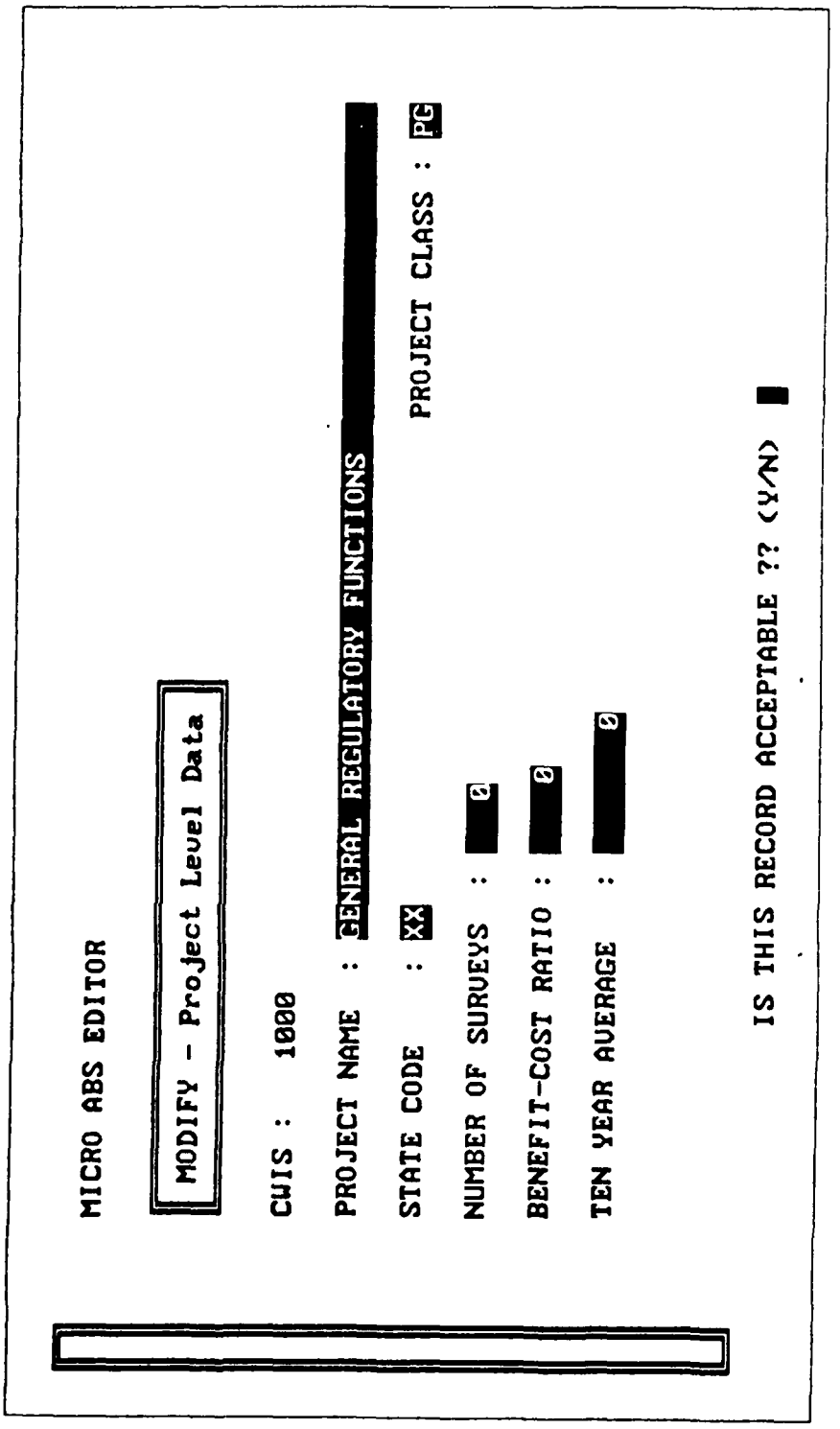
Stallings, W., "Protocols/Is There an OSI Session Protocol in Your Future?" *Data Communications*, Vol 16, No. 12 (November 1987).

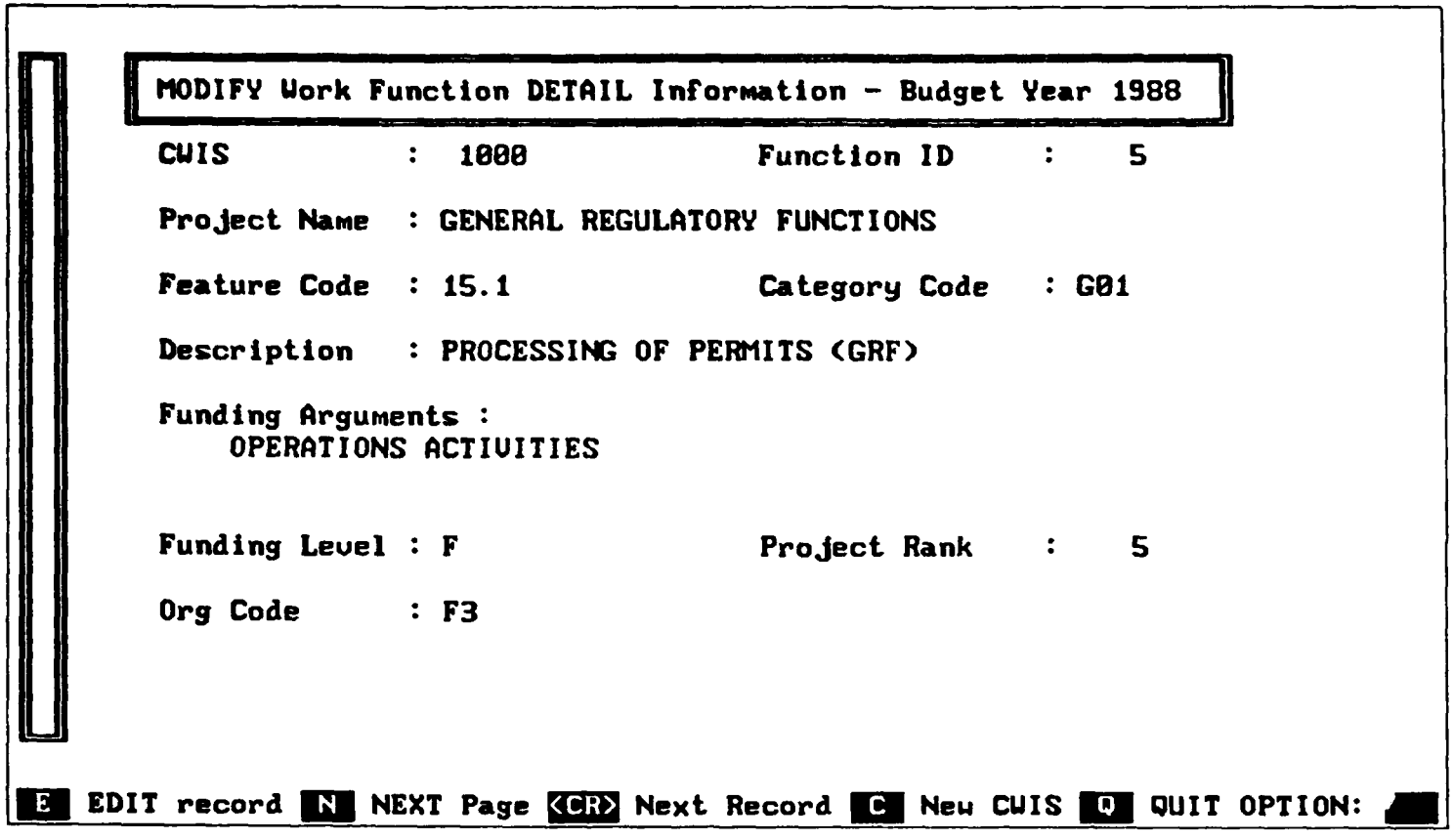
Wakid, S., "Network Features/Coming to OSI: Network Resource Management and Global Reachability," *Data Communications*, Vol 16, No. 13 (December 1987).

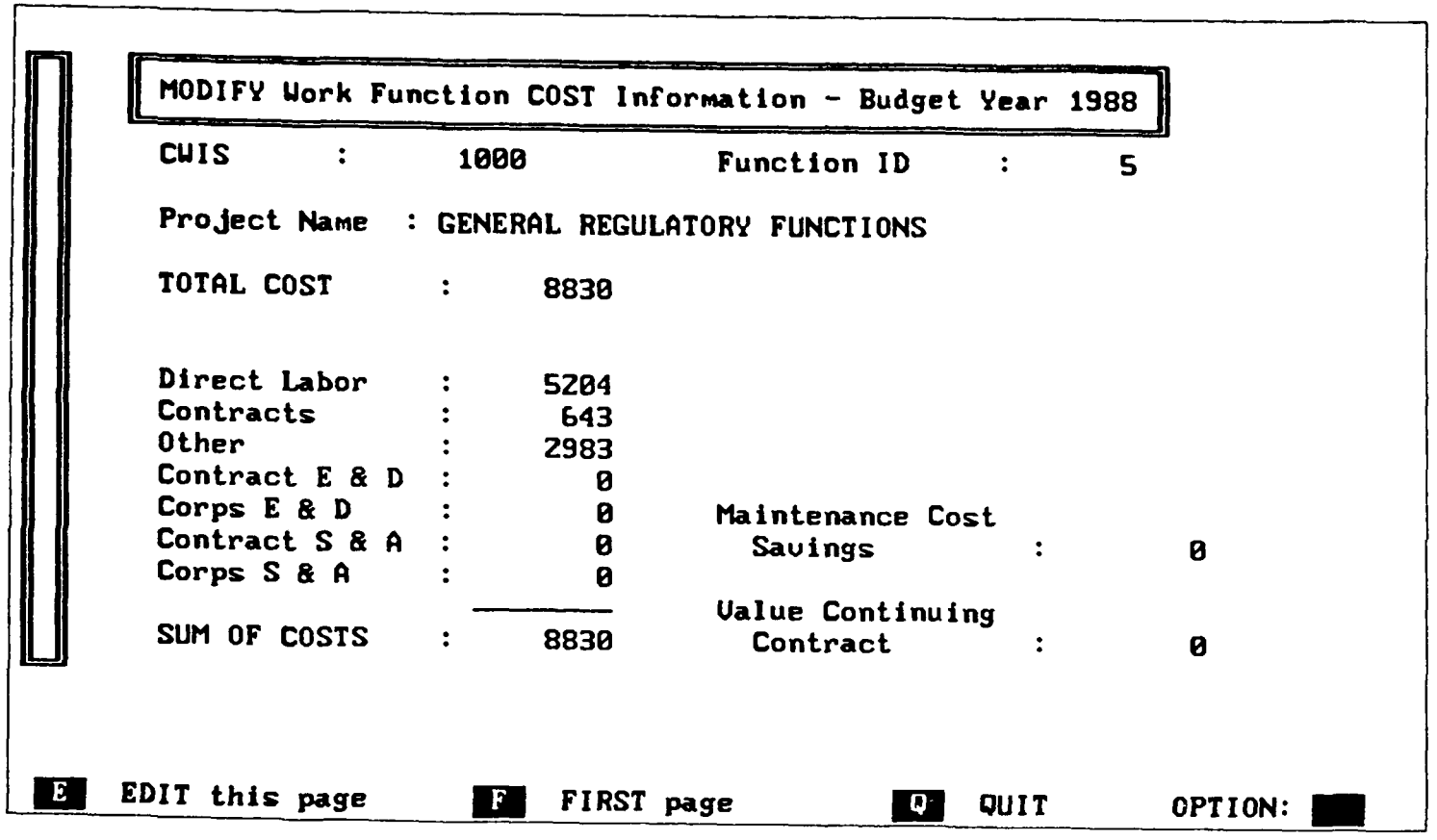
Wenig, R., and T. D. Pardoe, *Introduction to Data Communications* (Digital Educational Services, 1987).

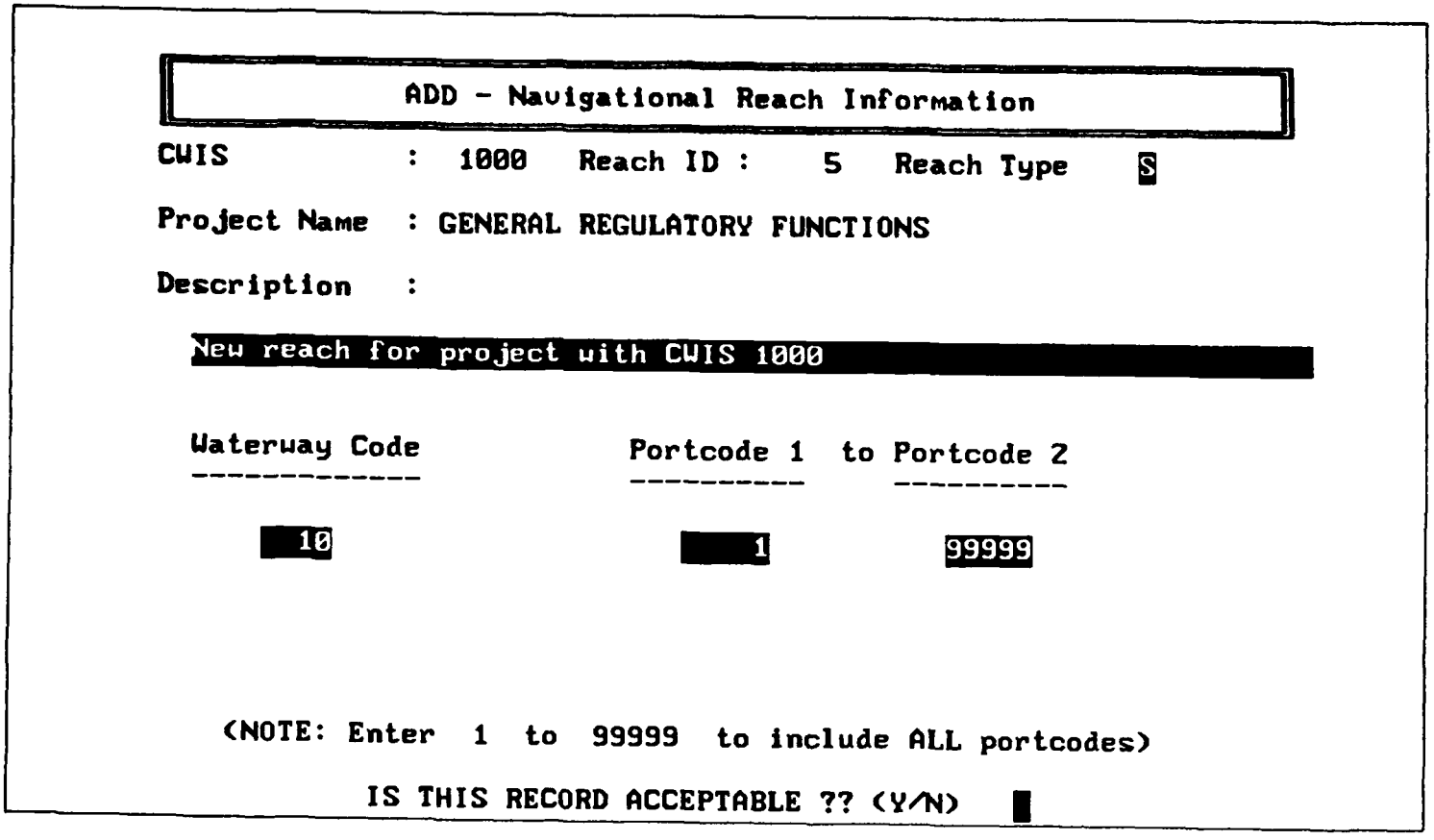
APPENDIX A:

EXAMPLE SCREENS FROM THE ABS EDITOR MODULE




MODIFY Work Function DETAIL Information - Budget Year 1988
CWIS : 1000
Function ID : 5
Project Name : GENERAL REGULATORY FUNCTIONS
Feature Code : 15.1
Category Code : G01
Description : PROCESSING OF PERMITS (GRF)
Funding Arguments :
OPERATIONS ACTIVITIES
Funding Level : F
Project Rank : 5
Org Code : F3
E EDIT record N NEXT Page <CR> Next Record C New CWIS Q QUIT OPTION:


MODIFY Work Function COST Information - Budget Year 1988
CWIS : 1000
Function ID : 5
Project Name : GENERAL REGULATORY FUNCTIONS
TOTAL COST : 8830
Direct Labor : 5204
Contracts : 643
Other : 2983
Contract E & D : 0
Corps E & D : 0
Contract S & A : 0
Corps S & A : 0
SUM OF COSTS : 8830
Maintenance Cost Savings : 0
Value Continuing Contract : 0
E EDIT this page
F FIRST page
Q QUIT
OPTION:


ADD - Navigational Reach Information
CWIS : 1000 Reach ID : 5 Reach Type S
Project Name : GENERAL REGULATORY FUNCTIONS
Description :
New reach for project with CWIS 1000
Waterway Code
Portcode 1 to Portcode 2
10
1
99999
(NOTE: Enter 1 to 99999 to include ALL portcodes)
IS THIS RECORD ACCEPTABLE ?? (Y/N)

APPENDIX B:

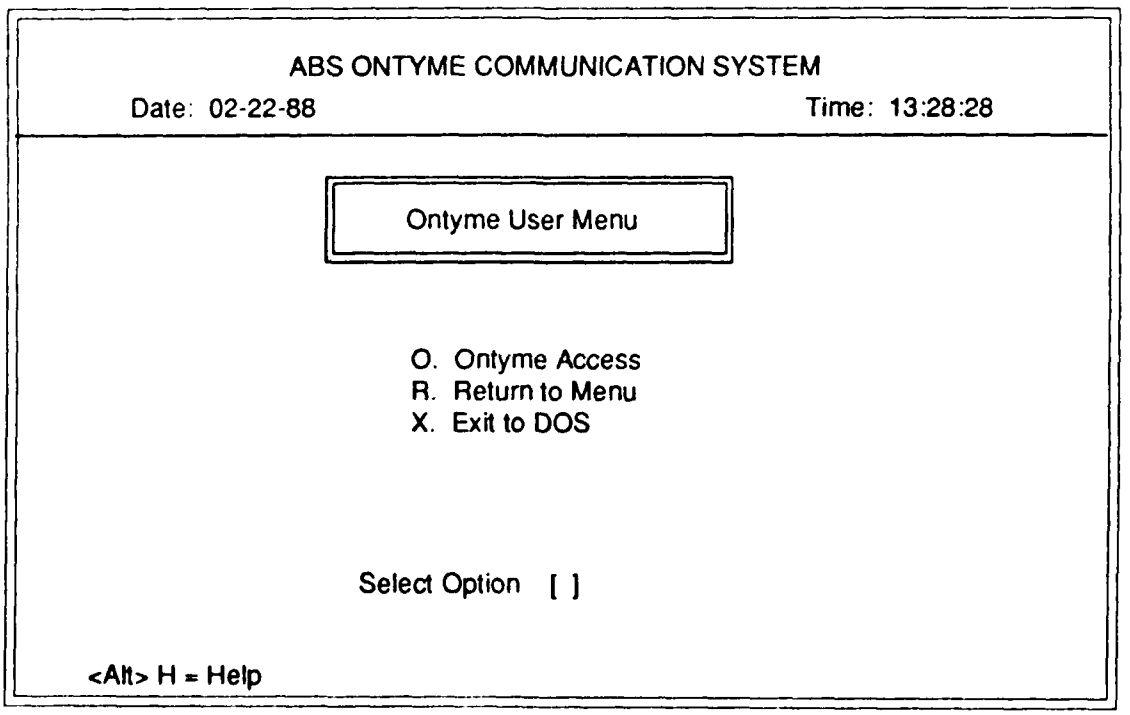
ONTYME COMMUNICATIONS INITIALIZATION FOR ABDS

INTRODUCTION

Updates to the Micro Automated Budget System (ABS) are distributed electronically, via Ontyme. When a program or file is changed, it is uploaded to Ontyme by HQUSACE or USACERL. The district users will receive an Ontyme mail message saying that an update of the system is available instructing them to select the Ontyme Communications option on the Micro ABS Main Menu to download the file(s). (Note: It is required that the Micro ABS Master PC be connected to a network/dataswitch or modem for using this procedure.) This handout provides documentation on initially setting up the parameters used for connecting to Ontyme and downloading an update.

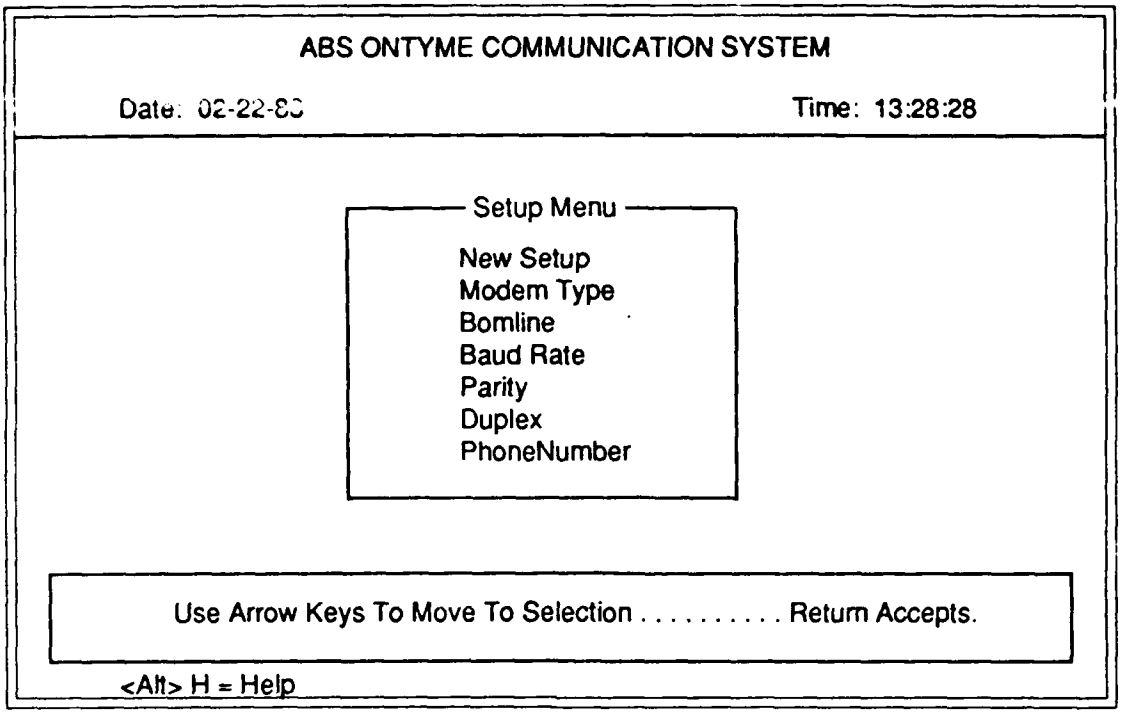
ONTYME ACCESS

The first time you select the Ontyme Communications option, you will be prompted for your three letter district id and last name. This information is stored on the PC and Ontyme, and is important for tracking what update version each district has at any given time. The following screen is then displayed.



SETUP PROCEDURE

Before option 1 is selected for the first time, it is suggested that you type **<Alt> S** to initiate a Setup menu, as shown below:



If you select **New Setup**, you will be shown a list of options for each of the parameters displayed and asked to select the correct one for your machine. The default parameters are:

Network	**Modem**
communications line 1	communications line 2
baud rate 4800	baud rate 1200
parity even	parity even
phone - none	phone - 0

Once the parameters have been set, they will remain for later use, unless you choose to change them. Note: The parameters may be changed from the Ontyme Communication Menu by typing **<Alt>S**. This procedure will place you in the setup routine described above.

The options for the parameters are listed below:

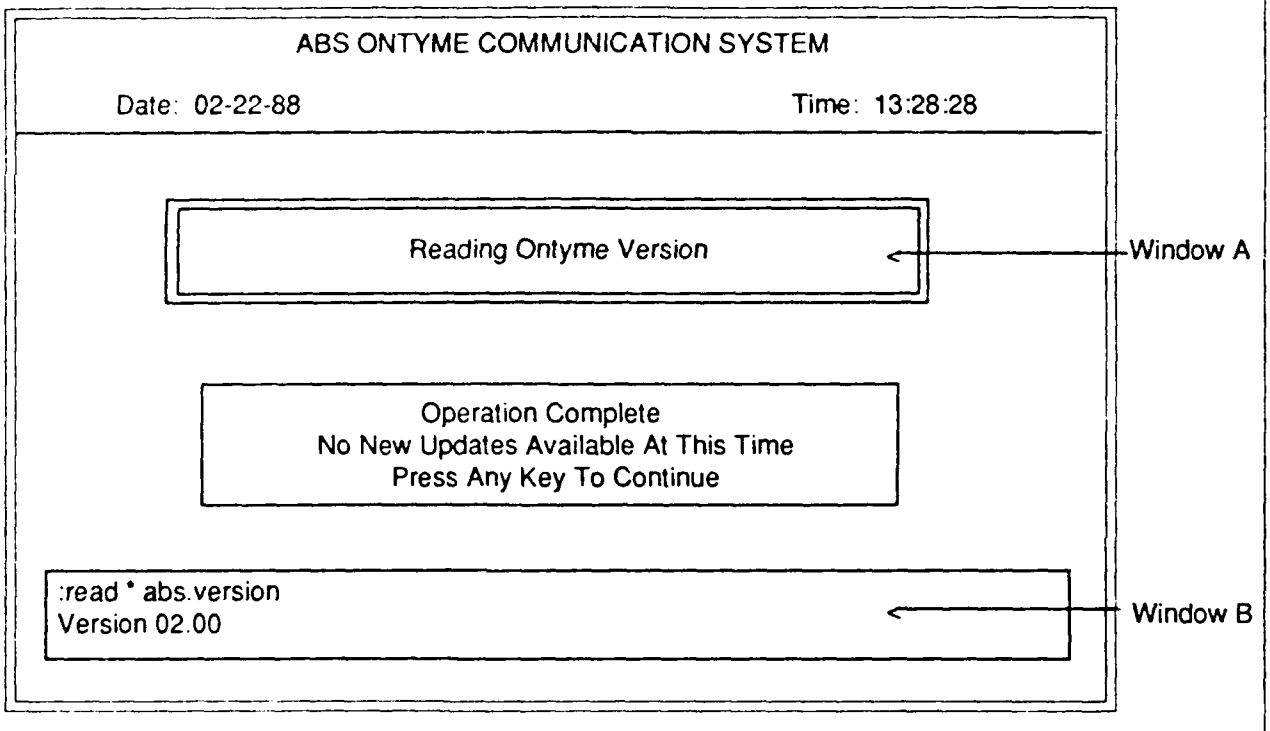
Modem Type	Network, Hayes Smart Modem
Comline	Comline1, Comline2
Baud Rate	300, 1200, 2400, 4800, 9600
Parity	Space, Mark, Even, Odd, None
Duplex	Full, Half
Phone Number	New Number, Delete Number Only necessary of using a modem, user adds numbers to list)

Note: When entering a phone number, the following prompt will be displayed: 0-0-000-000-0000. This allows for a 9 and a 1, if necessary. You should always enter the number right justified, for example: 0-0-009-356-1234.

DOWNLOADING

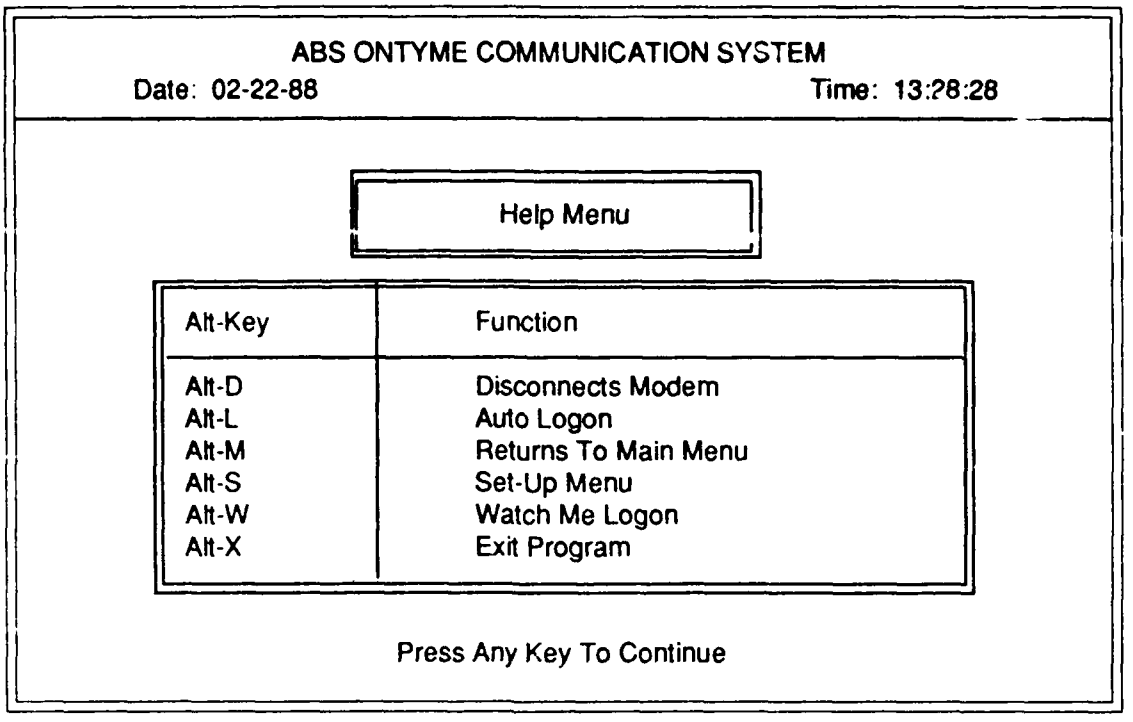
After proceeding through setup, you should select Ontyme Access in order to log onto Ontyme. At this point, the program is similar to other communications packages and you should follow the usual steps for connecting Ontyme. If connection is not made and you have trouble getting to Ontyme, double check the setup parameters by typing **<Alt> S**.

Once the message *please log in*: appears, instead of entering a logon, type **<Alt> C**. The program will continue to automatically log on to Ontyme under a specific id and download all program updates available since the last time you updated your system. If no updates are available, a message will be displayed and you will be logged out.



While you are logged onto Ontyme receiving updates, messages will be displayed in Window A, for information. Window B will echo the commands issued and Ontyme responses. In general, you can disregard any messages displayed in Window B.

HELP MENU



There is a help screen available to you at most places in the system by typing <Alt> H. It lists the special keys and their uses.

KEY	USE
Alt-D	Disconnects network and modem after connection has been achieved. It is prompted for by the system when it is appropriate.
Alt-H	Help screen. This key combination will display a help screen and is available at most places in the system.
Alt-L	Auto-Logon. Alt-L will automatically sign the user onto the appropriate host computer. May be invoked anytime the user must sign onto a host system, once Alt-W has recorded the logon sequence once.
Alt-M	Menu return. Alt-M returns the user to the Main Menu.
Alt-S	Setup Menu. This key combination can be used only from the Main Menu. It is used to bring up the Setup Menu. The setup routine is used to alter the communications parameters of the system.
Alt-W	Watch Me. This mode may be invoked when the user is signing onto a host computer system. It will "learn" the logon procedure and store it for later use, to be recalled using <Alt> L.
Alt-X	Exit. Press Alt-X at most times to exit the system.

USACERL DISTRIBUTION

Chief of Engineers
ATTN: CECW-OM
ATTN: CEIM-R
ATTN: CEIM-RC-A
ATTN: CEIM-RT-T
ATTN: CEIM-P
ATTN: CEIM-PD
ATTN: CEIM-PP
ATTN: CEIM-PE
ATTN: CEIM-S
ATTN: CEIM-SS
ATTN: CEIM-SL
ATTN: CECC-P
ATTN: CECW
ATTN: CECW-O
ATTN: CECW-P
ATTN: CECW-RR
ATTN: CEEC
ATTN: CEEC-C
ATTN: CEEC-E
ATTN: CERD
ATTN: CERD-C
ATTN: CERD-M
ATTN: CERM
ATTN: DAEN-ZCE
ATTN: DAEN-ZCI
ATTN: DAEN-ZCM
ATTN: DAEN-ZCZ

CEHSC, ATTN: Library 22060
ATTN: CEHSC-FU
ATTN: CEHSC-S
ATTN: CEHSC-SS
ATTN: CEHSC-SP
ATTN: CEHSC-SI
ATTN: CEHSC-SF
ATTN: CEHSC-SH

US Army Engineer Districts
ATTN: Library (41)
ATTN: IM (41)

US Army Engineer Divisions
ATTN: Library (13)
ATTN: IM (13)

US Military Academy 10966
ATTN: Facilities Engineer
ATTN: Dept of Geography & Computer Science
ATTN: MAEN-A

CEWES 39180
ATTN: Library
ATTN: IM

Information Systems Command
Headquarters, Fort Huachuca, AZ 85613
Info Systems Engr Command 85613
11th Signal Brigade 85613
5th Signal Command (Europe) 09058
7th Signal Cmd, Fort Ritchie, MD 21719
1st Signal Brigade (S. Korea) 96301
Western Command, Fort Shafter, HI 96858
Info Systems Cmd, Japan 96343

Defense Technical Info. Center 22314
ATTN: DDA (2)

158
+5
10/89