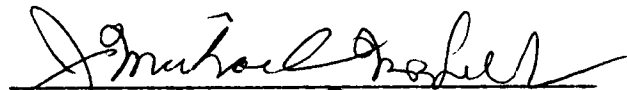
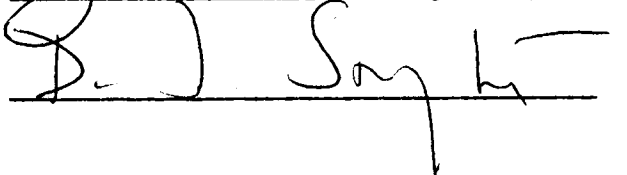


To the Graduate Council:

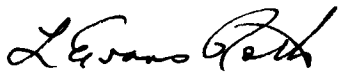
I am submitting herewith a thesis written by Charles L. McDowell entitled "A UCSD Pascal Communication Package." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.


Charles E. Hughes, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:


Vice Chancellor
Graduate Studies and Research

A UCSD PASCAL COMMUNICATIONS PACKAGE

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Charles L. McDowell

December 1980

3046699

ACKNOWLEDGMENTS

The author wishes to thank Charles Hughes, Mike Moshell, and David Straight for their time and suggestions as committee members. Also thanks to Mike Sappington, lab manager, for his assistance.

ABSTRACT

This paper describes a communication package containing three programs, SEND, RECEIVE, and TENCOM. Programs SEND and RECEIVE allow transfer of text or code files between any mini- or micro-computers running under the UCSD Pascal Operating System. Program TENCOM allows transfer of text files between a mini- or micro-computer running the UCSD Pascal Operating System and the DEC-10 at the UTCC. A description of implementation and user's guide is contained for each program.

TABLE OF CONTENTS

SECTION	PAGE
I. INTRODUCTION.	1
II. COMMUNICATION BETWEEN MINI-COMPUTERS.	4
1. General Description	4
2. User's Guide.	7
3. Timing Considerations	9
III. COMMUNICATION BETWEEN A MINI-COMPUTER AND THE DEC-10.	13
1. User's Guide.	13
2. TENCOM Implementation	20
IV. PROBLEMS AND SOLUTIONS.	28
1. Physical Connection	28
2. BIOS Difficulties	28
3. Input Buffers	29
4. Interactive File Specification.	29
5. Synchronization	30
6. Recognizing Proper Ascii Characters	31
7. Sending A Carriage Return to the DEC-10	32
V. CONCLUSION.	33
1. Modifications	33
2. Suggested Further Research.	33
LIST OF REFERENCES	34
VITA	36

I. INTRODUCTION

The mini-computer Lab at the University of Tennessee is operated by the Computer Science Department. It has facilities for use in course work and research projects. Included in these facilities are mini-computers, micro-computers, terminals, and printers of several varieties and a telephone modem for linkage to outside computers. There is also a plug-in board which allows the user to choose his system of computer, terminal and printer. The computers available in the mini-lab include a PDP-11/34, an Imsai 8080 and several Apple II's. Although there are multiple operating systems up on each machine, this project was concerned only with the UCSD Pascal Operating System.

[Bowles]

The UCSD Pascal Operating System runs on mini- and micro-computers, controlling up to 64K bytes of memory. It is machine independent in that all systems programs are written in the same language (an extended version of Pascal) except for an interpreter called the 'P-machine'. This P-machine must be written in the native assembler language of the host machine. The system contains file maintenance routines, a line oriented editor, a screen oriented editor, Pascal and Basic compilers, a linker, an assembler, and several utility programs.

Considering the highly machine independent nature of this operating system, programs developed in this environment are very portable. In fact, a program compiled on one machine can be run on another without recompilation, if it can be transported. This feature is the basis for this project. If two machines have the same type and style of mass storage device, then all that is required to transport the program is physically carrying the storage medium containing the program from one machine to the other. But in the mini-lab, machines using the UCSD system have three types of storage devices -- 5 1/4 inch floppy disk, 8 inch floppy disk and RK05 hard disk. For complete portability a communication package allowing transmission and reception of files was needed.

This project developed two programs, SEND and RECEIVE, which are designed to read a text or code file from one machine and move it to another machine, regardless of the mass storage device type of either machine.

Many programmers who have written programs in standard Pascal on the UTCC DEC-10 have expressed an interest in using their programs on the computers in the mini-lab. Although these cannot be executed under the UCSD system in their compiled form, the text can be recompiled under UCSD and then used. Programs written in UCSD Pascal may be run on the DEC-10 with modification (substitution for extended features) and recompilation. This project developed a

program, TENCOM, which will download text files from the DEC-10 to any computer under UCSD and also will upload text files from any computer under UCSD to the DEC-10.

Many features in these programs could have been made to run more quickly or efficiently by the addition of assembler language routines to perform input/output functions. But, a primary objective in this project was to achieve as high a degree of machine independency as possible, so all procedures were written in UCSD Pascal, allowing portability from machine to machine.

II. COMMUNICATION BETWEEN MINI-COMPUTERS

The first phase of this project was to establish communication between two mini-computers, each using the UCSD Pascal operating system. Two programs, SEND and RECEIVE, were developed to accomplish this task.

The programs are designed to execute simultaneously on the two machines between which communication is needed. SEND is executed on the source machine and RECEIVE is executed on the destination machine. These programs allow transmission of both text and code files at baud rates of up to 9600. The programs test for erroneous transmission and will retransmit bad characters, if necessary.

1. GENERAL DESCRIPTION

Remote input and output is achieved in UCSD Pascal by two serial ports called REMIN: and REMOUT:. These are considered to be files and are accessed by declaring a variable as a pointer into the file. They may be opened by using the RESET and REWRITE functions as follows:

```
RESET(A,'REMIN:');  
REWRITE(B,'REMOUT:');
```

where A and B are declared as variables of type INTERACTIVE.

On all machines using UCSD Pascal in the mini-lab, except the Apple, REMIN: and REMOUT: are readily available without any hardware modifications. The Apple uses a serial

interface card which does double duty -- in one slot it controls communication with the printer, in another slot it controls remote input and output ports. Short of buying a second serial interface card at a cost of approximately \$200, there is only one solution. That is to open the computer and move the card to the proper slot when needed for SEND and RECEIVE. In its normal position (used for printer) the card resides in slot one, the second slot from the left. When needed for remote input and output, the card must be moved to slot two, the third slot from the left.

There is no capability to test the port status in UCSD Pascal. This requires that the programmer must pay strict attention to synchronization of two programs which are attempting to communicate. A program which reads from REMIN: will wait until a signal is on the line. A program which writes to REMOUT: will continue execution. This situation can lead to problems if the input buffer of a machine is small. The input buffer on the Imsai 8080 is two characters, on the Apple it is one character. The reading program must empty its input buffer faster than the writing program fills it or else there will be garbled characters in the buffer resulting from the attempt to overfill it. Synchronization of SEND and RECEIVE was achieved by a combination of two methods:

- a) requiring the writing program to wait for an echo from the reading program before writing another character,
- b) building in wait loops to allow the other program to catch up on work such as disk access and translation.

These programs were designed to transmit both text and code files. Because code files may contain random control characters which can cause adverse reactions by the machines when read or written, and to prevent making a special case of code files, it was decided not to send the actual hexadecimal representations of the ascii characters in the files. Instead, each character is translated into the two characters which form the hexadecimal representation of the character. This assures that all transmitted characters will be in the range of 0-9, A-F. For example, if the character from a file is 'A', the hexadecimal value of its ascii code is 41. This would be translated to the two characters '4' and '1' and then transmitted as two separate characters. The receiving program would read this as two characters and then translate them back to 'A' before writing to the destination file. Control-G or BEL would be translated and transmitted as '0' and '7'. While this translation process is not necessary for all characters, there are several for which it is absolutely necessary (for example, Control-S).

In order for the transmission to take place, there must be a physical connection between the two machines. This is made possible by a three wire cable using RS-232 connectors. The connectors are wired as follows:

A	to	B
pin 7		pin 7
pin 2		pin 3
pin 3		pin 2

Pin 7 carries ground, pin 2 is output and pin 3 is input. There is a cable with these connections residing in the mini-lab.

These programs are compiled under the \$I- option which turns off all tests for errors in I/O operations. Therefore, the function IOCHECK has been added to both programs. During all disk access operations, IOCHECK is invoked to check for any errors that may have occurred. IOCHECK is also used to test whether a file exists. A RESET is done on the specified file name. If IOCHECK returns a value of zero (signifying no I/O error), this means that the file exists. If IOCHECK returns a non-zero value, the file does not exist.

2. USER'S GUIDE

Before executing the programs SEND and RECEIVE, the machines should be prepared as follows:

- a) If the Apple is involved, the serial interface card should be placed in slot two.
- b) Baud rates for both machines should be set to the same. The programs were designed to operate at 9600 baud, but slower rates will work as long as both machines use the same rate.
- c) Connect the communication interface cable to the ports of both machines.
- d) Obtain disks with the appropriate programs (SEND and RECEIVE) and boot Pascal on the machines.

Once these steps have been taken, the user is ready to execute the programs. The RECEIVE program will ask the user for a destination file name, the SEND program will ask for a source file name. The user should supply the RECEIVE program with the destination file name and press carriage return BEFORE doing so with the SEND program. This is important to allow the programs to synchronize themselves properly. The file names must be fully specified as follows:

<unit number>:<file name>.<extension>

The unit number may be omitted if the file is to be taken from or placed on the prefixed disk. The extension must be used for all files that have an extension.

No other information is required by either program except in the following cases:

- a) If the SEND program is given a non-existent source file name the user is asked to supply another name.
- b) If the RECEIVE program is given a destination file name that already exists, the user is asked if he wants to remove that file. If a 'Y' is entered, the program continues. If any other character is entered, the user is asked to supply another file name.

Thereafter, each time SEND reads, translates, and transmits a block, a short message is written to the screen. Each time RECEIVE receives, translates, and writes a block, it also writes a short message to the screen. Processing of each block (512 bytes) should require 30-35 seconds at 9600 baud. This time is variable on different machines due to different amounts of time required to read from or write to disk. Also, instruction times vary on different machines. The time quoted above was taken from transfers from an Apple to an Imsai 8080.

3. TIMING CONSIDERATIONS

The programs SEND and RECEIVE use more time than would seem necessary when using figures for burst baud rates. To transmit a block of 512 characters, actually 2048 characters

are sent due to translation of a character into its hex representation and echoing of each character. At 9600 baud, one bit is transmitted every 104 microseconds. This would cause a block to be sent in about 2 seconds. This is much less than the timed rate of just over 32 seconds.

This can be explained by the overhead time costs involved. A fixed time is required to read from disk and translate in the SEND program and to retranslate and write to disk in the RECEIVE program. Also, a fixed amount of time is required by the basic I/O system and program checks for each character. The only variable is baud rate. The following equation shows the relationship:

$$\text{Time/Block} = K1 + K2 + K3/B$$

where:

K1 is read, write, and translation time,

K2 is I/O and program overhead,

K3 is number of bits transmitted per block,

B is the baud rate.

It was originally thought that the transmission time was a function of K1 plus the baud rate, but using the following figures and graph, it was shown that K2 exists as a constant, leaving baud rate as the only variable in the function.

BAUD RATE	TRANSMISSION TIME
9600	23.0
4800	25.25
2400	29.5
1200	37.75
300	91.75

By using the above times (taken by stopwatch), the graph in Fig. 1 shows the relationship between the baud rate curve and the timed transmission curve. The distance between the curves represents the constant, K_2 .

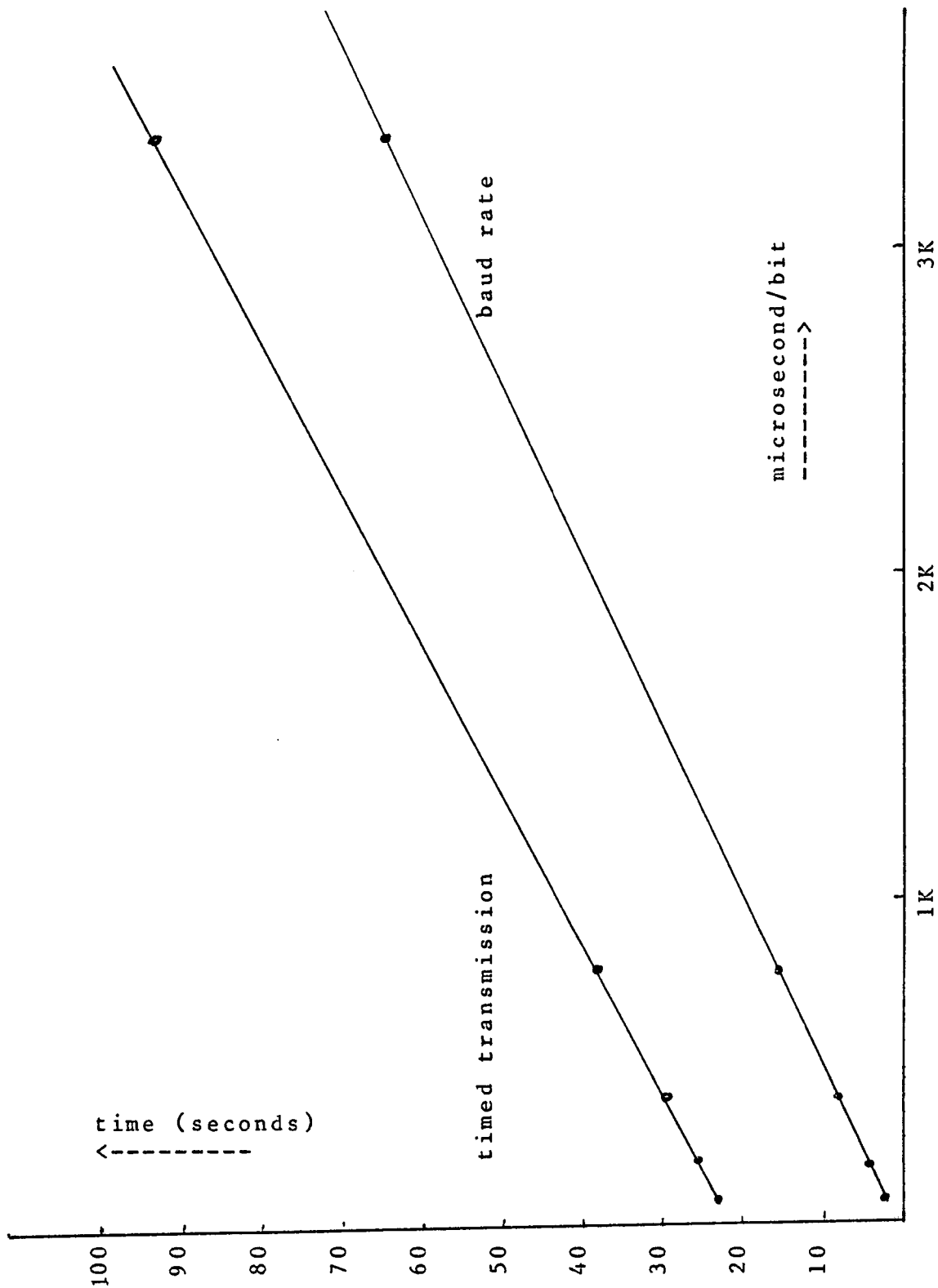


FIGURE 1. BAUD RATE VS. TIMED TRANSMISSION

III. COMMUNICATION BETWEEN A MINI-COMPUTER AND THE DEC-10

The second phase of this project was to establish communication between a mini-computer using the UCSD Pascal operating system and the DEC-10 at the UTCC. The program TENCOM was developed to accomplish this task.

The program was designed to log on to the DEC-10, issue monitor commands, recognize necessary monitor messages, upload and download text files. Transmission of code files was not considered because recompilation and in some cases, modification, is necessary before execution of the files on the new machine.

The monitor used on the UTCC DEC-10 at the present is the '7.01' monitor. This monitor is under field test and may not be in permanent use. If the monitor is changed in the future, the only modification necessary to the program TENCOM will be in the procedure that recognizes the log on message. This possible modification is discussed in the section on implementation.

1. USER'S GUIDE

Before executing the program TENCOM, the mini-computer should be prepared as follows:

- a) If the Apple is involved, the serial interface card should be placed in slot two.

- b) The baud rate should be set to 300. This is the maximum rate that can be used by the DEC-10 over the telephone modem.
- c) Connect the communication interface cable to the port of the machine and to the port on the telephone modem.
- d) Obtain a disk with the program TENCOM and boot Pascal on the machine.
- e) Dial the DEC-10 number on the telephone modem.

When these steps have been taken, the user is ready to execute TENCOM. TENCOM prompts the user for his project-programmer number:

WHAT IS YOUR PPN? --

The user should supply the PPN and terminate with a carriage return. If a mistaken character is typed, the user can correct it (before the carriage return) by typing backspace or rubout, and retyping the correct character. The user is then prompted for his password:

WHAT IS YOUR PASSWORD? --

The user should supply his password and terminate with a carriage return. The password will not be echoed on the screen to facilitate secrecy. A mistake may be corrected by a backspace or rubout.

The program will attempt to log on the the DEC-10 at this time. If the DEC-10 is not available, the program will print a message,

DEC-10 NOT AVAILABLE

TRY AGAIN WHEN IT IS UP.

abort execution, and return to the Pascal monitor. If the DEC-10 is available, the program will transmit the log on message, including 'LOG', PPN, switches for no mail and no notices, and the password. It then reads the message returned from the DEC-10 and determines whether the log on is successful or unsuccessful. If unsuccessful, the user is asked if he would like to reattempt the log on:

LOG ON IS UNSUCCESSFUL

WOULD YOU LIKE TO TRY AGAIN? Y OR N --

If the user's response is no then the program is aborted. If the response is yes, then the user is re-prompted for his PPN and password. Care should be taken to enter the correct characters.

When the logon has been successful, the user is prompted for his choice of action:

PLEASE ENTER OPTION -- R)ECEIVE OR T)RANSMIT --

If the user would like to download a file from the DEC-10 to a mini-computer, then he should enter the character 'R'. If he would like to upload a file from a mini-computer to the DEC-10, he should enter the character 'T'.

RECEIVE

If the user's option is receive, he is asked if he would like to inspect his DEC-10 directory:

DO YOU NEED TO SEE YOUR DEC-10 DIRECTORY? Y OR N --

If the response is 'Y', then the user's directory is displayed and execution continues. If the response is any other character, then the next prompt asks for a file name from the DEC-10:

WHAT FILE WOULD YOU LIKE TO DOWNLOAD?

FILENAME --

The user should respond with the complete file name, including extension (if any) and terminate with a carriage return. The program shows the user the filename selected and asks the user to inspect it for correctness:

THIS IS YOUR FILENAME - IS IT CORRECT? -- filename

Y OR N --

If the user enters 'N', then the process is repeated. Otherwise the program tests to see if the file actually exists. If the file is non-existent, then the user is given the message:

THE FILE filename DOES NOT EXIST

WOULD YOU LIKE TO CONTINUE? Y OR N --

If the response is 'N', then log off is done and the program is aborted. If the response is any other character, then the cycle is repeated, beginning with the prompt for directory.

If the file does exist, the user is prompted for the name of the new file to be created on the mini-computer:

WHAT IS THE DESTINATION FILE?

FILENAME --

The user should supply the unit number (unless the file is to be placed on the prefixed disk) and the file name (an extension is optional). The user should take care to make sure that the disk on which the new file is to be created has enough free blocks to accomodate the entire file. The program tests whether this file already exists. If it does then the message

REMOVE OLD filename ?

is displayed. If the user responds with a 'Y', then the old file is purged and a new file is opened under that filename. If any other character is entered, then the user is re-prompted for another destination file name.

The program then begins receiving the file and when complete displays the message:

THAT'S ALL FOLKS

terminates execution and returns to the Pascal monitor.

TRANSMIT

If the user's option is TRANSMIT, then the next prompt asks for the source file name from the mini-computer:

WHAT FILE WOULD YOU LIKE TO SEND?

FILENAME --

The user should fully specify the file, including unit number (unless the file is on the prefixed disk), file name, and extension (if there is one). If the file exists, execution continues. If it does not exist, a message is displayed:

THE FILE filename DOES NOT EXIST

WOULD YOU LIKE TO CONTINUE? Y OR N --

If the response is 'N', log off is done and the program is aborted. If the response is any other character, then the user is re-prompted for a source file name.

The user is next asked if he would like to inspect his DEC-10 directory:

DO YOU NEED TO SEE YOUR DEC-10 DIRECTORY? Y OR N --

If the response is 'Y', the directory is displayed. If the response is any other character, the user is then prompted for the destination file on the DEC-10:

WHAT IS YOUR DESTINATION FILE?

FILENAME --

The user should check his directory before attempting to up-load because creation of this file will destroy a previous file by the same name without any warning.

When transmission is complete, the program logs off the DEC-10, displays the message:

THAT'S ALL FOLKS

terminates execution and returns to the Pascal monitor.

In the event that the user's DEC-10 disk space is over his limit, the log off will not occur. Instead, the message:

YOUR DEC-10 DISK QUOTA IS EXCEEDED.

WOULD YOU LIKE TO DELETE A FILE? Y OR N --

If the user responds with 'N', then the program terminates. The user must then attach to the job via a regular terminal in order to prevent indiscriminate deletion of files by the DEC-10 operating system. If the user responds with 'Y', then his DEC-10 directory is displayed. The user should not attempt to use control-S to freeze the screen. The directory is shown at a fairly slow rate, so that the user can read the file names without stopping the display.

The user is then prompted for a file name:

WHAT FILE WOULD YOU LIKE TO DELETE?

FILENAME --

The user supplies a file name and it is deleted. The user's directory is displayed again for his inspection. He is again asked if he would like to delete a file. If the response is yes, then the directory, delete, directory sequence is repeated. If the response is no, then log off is attempted.

2. TENCOM IMPLEMENTATION

SENDING AND RECEIVING

The main procedures in TENCOM for sending to and receiving from the DEC-10 are TRANSMIT and RECEIVE. Here is an explanation of the way each works.

TRANSMIT, after verification of filenames, uses PIP on the DEC-10 to create the destination file. The routine reads a line from the source file on the mini-computer and transmits it one character at a time to the DEC-10. If the character is a control, then instead of sending it as it is, the program translates it to: space, '^', letter associated with control character, space. For example, control-H would be transmitted as ' ^H '. This allows the user to safely send control characters and to edit them once they are in his DEC-10 file. After each character transmission, the routine reads the echo and checks it against the sent character. If they are equal, execution proceeds. If they are not equal, a control-U is sent to cancel the line and transmission is begun at the beginning of the line again. If the echo is a control-G, (meaning buffer overflow in the DEC-10), a control-U is sent and the line is re-transmitted.

There is a limit to the number of times that a single line will be re-transmitted. At the beginning of execution, this limit is twenty times. If a single line is re-transmitted twenty times, then the program is aborted.

But, for each separate line that is re-transmitted, the limit is decremented by one. For example, if line A is re-transmitted, then the limit for ensuing lines (which began at twenty) now becomes nineteen, so that if line B must be re-transmitted, it may only be re-transmitted nineteen times before program termination. If any line is re-transmitted a number of times between one and the current limit, the limit is decremented. If any line is re-transmitted a number of times equal to the current limit, the program is aborted. If the current limit becomes zero and re-transmission of a line is attempted, the program is aborted.

Once the file has been completely sent, the routine sends a control-Z to PIP which closes the destination file. Then it sends a control-C which exits PIP. Finally, logoff is done and execution terminates.

RECEIVE uses the text editor SOS to download files from the DEC-10 one line at a time. After opening the destination file on the mini-computer, the routine runs SOS on the source file. This not only verifies the existence of the source file, but also allows line by line access to the file.

Once the file is entered using SOS, a line feed is sent. This prompts the DEC-10 to send one line from the file. RECEIVE stores each character from the input port into an input buffer until the carriage return at the end of

the line is recognized. It then checks for end of file by looking for

%No such line(s)

in the line just received. The SOS line numbers are stripped and the line is written to the destination file.

MESSAGE RECOGNIZERS

Several procedures in TENCOM require the use of string recognizers in order to understand messages sent by the DEC-10. Each recognizer is a Mealy type finite state machine. [Kohavi] This technique was adapted a thesis written in 1979 by Mukesh Sundarem. [Sundarem]

The function LINEINIT, which prepares the line by sending the DEC-10 select sequence, 'PD', must recognize either the 'Please log on' message or the 'System not available' message. The first message contains the characters '7.01' and the second contains 'Not a'. Fig. 2 shows a diagram of the finite state machine used to recognize these sequences.

S0 is the initial state, S4 and S8 are the final states. If S4 is reached then the DEC-10 is not available. If S8 is reached then the DEC-10 is ready for logon. 'X' represents any character other than one required to move the machine to a next higher valued state.

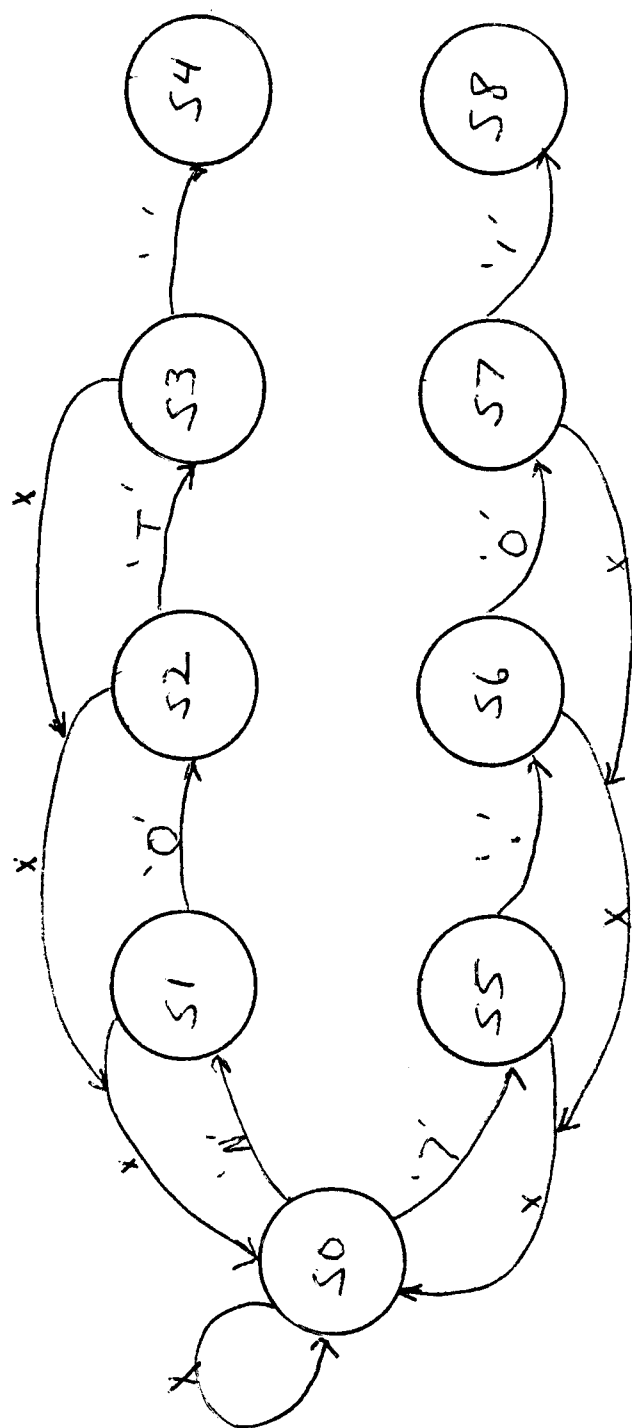


FIGURE 2. STRING RECOGNIZER

The message recognized by the above machine will be changed if the DEC-10 monitor is changed in the future. At present, UTCC is field testing the '7.01' monitor. The standard monitor is TOPS-10. If TOPS-10 is reinstated, the log on message will be changed. The finite state machine will have to be modified to recognize the different message. By replacing recognition of '7.01' by 'Pleas', this can be accomplished.

Procedure GETTENFILE opens a file on the DEC-10 which is to be down-loaded to a mini-computer. In order to decide whether the file exists, TENCOM runs the editor SOS on the file. If the message returned contains the string 'Creating', then the file does not exist, but if it contains the string 'Editing', then the file does exist. The machine shown in Fig. 3 recognizes either 'Crea' or 'Edit'.

S0 is the initial state, S4 and S8 are the final states. 'X' represents any character other than one which will move the machine to a next higher valued state. If the final state reached is S4, then the file is non-existent. If the final state reached is S8, then the file exists.

The procedure RECEIVELINE reads lines one at a time as sent from the Dec-10. When the down-loaded file is completed, upon request for another line, the message

%No such line(s)

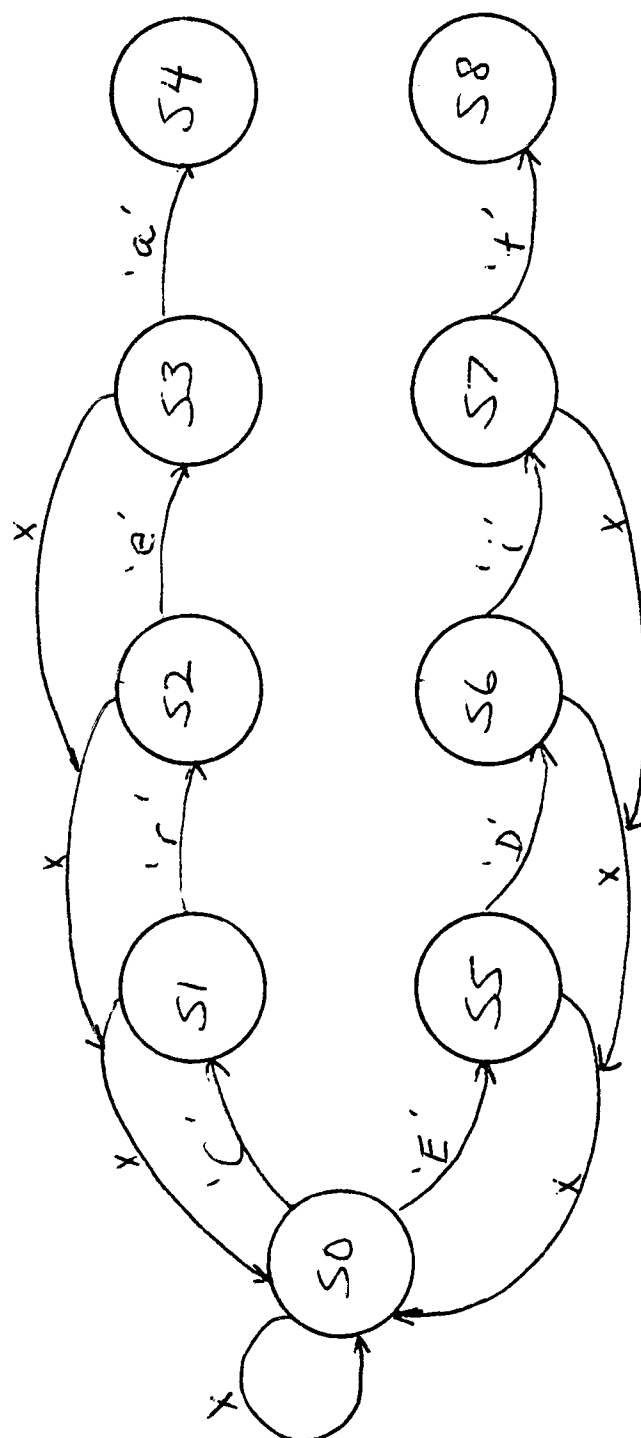


FIGURE 3. STRING RECOGNIZER

is sent. Fig. 4 shows a machine that finds the string '%No s'. This machine is run on each line to test for end of file.

S0 is the initial state, S5 is the final state. 'X' represents any character other than one which will move the machine to the next higher valued state. When S5 is reached, end of file is indicated.

In each of the above string recognizers, the individual elements of the strings are very likely to change with the introduction of a different monitor on the DEC-10. Rather than use the actual characters in the strings, CONSTs are used to allow easy modification. When the message strings are changed, the CONSTs may be changed, with no other modification to the program.

BUFFERS

Procedures which must recognize strings sent from the DEC-10 contain input buffers of appropriate size to store the various strings. Procedure Transmit uses an output buffer, LINE, which has room for 80 characters. Procedure RECEIVE uses an input buffer, BUFFER, which has room for 100 characters. These may be altered by changing the values of the CONSTs INBUFFSIZE and OUTBUFFSIZE at the beginning of the program and recompiling the program.



FIGURE 4. STRING RECOGNIZER

IV. PROBLEMS AND SOLUTIONS

While working on this project, several 'stumbling blocks' came up. In order to help those who work on similar projects in the future, a list of many of these problems and an explanation of their solutions is contained here. These are listed in approximate chronological order of occurrence.

1. PHYSICAL CONNECTION

At first the attempt to use the plug-in board in the mini-lab for an interface was made. One computer could send to the other, but not vice-versa. Upon checking the RS-232 connectors to the plug-in board, it was found that two-way communication was not possible because of non-standard pin connections. A new cable with pin connections (described in an earlier chapter) for two-way communication was assembled.

2. BIOS DIFFICULTIES

When using the Imsai in trying to communicate, there was still a problem (even after using the new interface cable). The Imsai was not able to either send or receive. Upon examining the code in BIOS (Basic I/O System) used on the Imsai, the reason became apparent. The routines for remote input and output jumped directly to the console controller. By rewriting the code to handle the list port

on the Imsai, two-way communication on the machine was established.

3. INPUT BUFFERS

The machines used in this project have different input buffer sizes. This caused the programs to work differently on the different machines. This problem was handled in the programs SEND and RECEIVE by adding a routine which initialized the contents of these buffers.

4. INTERACTIVE FILE SPECIFICATION

Since the remote input and output ports, REMIN: and REMOUT:, are considered by the Pascal system to be files, a file pointer must be declared and used when opening, closing, and accessing these files. There are several choices for file types in UCSD Pascal, including TEXT, INTERACTIVE, and untyped or FILE.

It was assumed that since the definition of file type TEXT is a file of character, the input and output port file pointers should be declared as type TEXT. But, when accessing a file of type TEXT, a look-ahead for the next character is always done. This caused problems with character recognition and synchronization. This type should be used when accessing actual disk files.

Since the files for input and output ports act more like the keyboard and console files, INTERACTIVE file type was chosen for use with them. This prevents the problems mentioned in the above paragraph.

5. SYNCHRONIZATION

Because there is no ability for polling the REMIN: port, special care must be taken to be sure that a reading program always has a read instruction waiting when the writing program executes the write. In SEND and RECEIVE the solutions to these problems were discussed in an earlier section. With the TENCOM program, synchronization was more difficult.

Because there is no way to control character by character output from the DEC-10, all messages from the DEC-10 must be read into an input buffer without processing until all input characters have been received. Another problem arises here. If a read from the input port is issued by TENCOM and there is no output from the DEC-10, TENCOM will wait until a character comes on the line. This will cause problems, because the DEC-10 will not respond without input, except in cases like OPSEK messages. TENCOM must issue a number of reads which is less than or equal to the number of characters output from the DEC-10. This is accomplished by either counting the characters in a message, or by recognizing the last character in the message.

6. RECOGNIZING PROPER ASCII CHARACTERS

When reading characters from the DEC-10, a puzzling phenomenon occurred. After reading a character into the variable CH, the following code

```
WRITELN('THIS CHARACTER IS ',CH);  
IF CH = '*' THEN WRITELN('STAR')  
ELSE WRITELN('NOT A STAR');
```

would produce the output

```
THIS CHARACTER IS *  
NOT A STAR
```

The write routine would recognize the characters as they were read, but comparisons would not work at all. This is because the upper bit in most characters was being set by the DEC-10. This bit is used in comparisons but not in writing. The solution to this was to test the ORD of each character as it was read. If its ORD was above 127 (decimal, 177 octal), then the upper bit was removed by subtracting 128 from the ORD of the character.

```
IF ORD(CH) > 127 THEN CH = CHR(ORD(CH) - 128);
```

This placed all characters in the range of 0-127 (decimal) so that both writes and comparisons treated characters the same way.

7. SENDING A CARRIAGE RETURN TO THE DEC-10

When a carriage return is sent from a machine using UCSD Pascal, a line feed is appended to it. When a carriage return is read by the DEC-10, a line feed is appended to it. This would cause CR,LF,LF to be placed in the DEC-10 input buffer when a carriage return is sent from a mini-computer. TENCOM's receive routine uses SOS to cause line by line transmission of files. The character used to get a new line is line feed. When entering SOS the characters

SO filename <cr>

are sent to the DEC-10. When the DEC-10 appends the extra line feed to the carriage return at the end of the string, it causes the first line of the file to be lost because TENCOM was not ready to read it yet. Also TENCOM's transmit routine had a similar problem with carriage return. After sending all characters in a line, a carriage return was sent. The DEC-10 appended an extra line feed causing a blank line between each line of a file.

The solution to both of these problems was the same. Deceiving the mini-computer into not knowing it had sent a carriage return was the answer. Whenever a carriage return was needed, the character whose ORD is 128 greater than the ORD of carriage return was sent (decimal 141). The mini-computer would not append the extra line feed but action caused by a carriage return at the DEC-10 would be the same.

V. CONCLUSION

1. MODIFICATIONS

These programs were written with the intent of easy modification. While programs SEND and RECEIVE will probably require little or no modification, TENCOM is at the mercy of the monitor used on the DEC-10. Most (if not all) required modifications can be made within the CONST declarations at the beginning of the program.

2. SUGGESTED FURTHER RESEARCH

In writing these programs, one of the greatest problems which arose was that of small input buffer size coupled with the inability to poll the input port. A modification to the Pascal I/O system which includes a buffer of the type used in keyboard and console I/O would make a Pascal communication program easier to work with and also faster. The ability to poll the input port would greatly enhance synchronization.

LIST OF REFERENCES

LIST OF REFERENCES

1. Bowles, Kenneth L., UCSD (Mini-Micro Computer) Pascal Version II.0, UCSD Institute for Information Systems, La Jolla, Ca., 1979.
2. Jensen, Kathleen, and Wirth, Niklaus, Pascal User Manual and Report, Springer-Verlag, New York, 1974.
3. Kohavi, Zvi, Switching and Finite Automata Theory, Mcgraw-Hill Book Company, New York, 1978.
4. Sundarem, Mukesh, UNIX to DEC-10: A Simple Teleprocessing Link, University of Tennessee, Knoxville, Tenn., 1979.

VITA

Charles McDowell was born in Cleveland, Tennessee in 1950. He graduated from Cleveland High School in 1968. After attending Cleveland State Community College, the author entered The University of Tennessee and received a Bachelor of Science degree from the Institute of Agriculture, Knoxville Campus, in June, 1974.

After working in local business and teaching high school in Sevier County, Tennessee, he returned to graduate school at The University of Tennessee in 1978. He received the Master of Science degree with a major in Computer Science in December 1980.