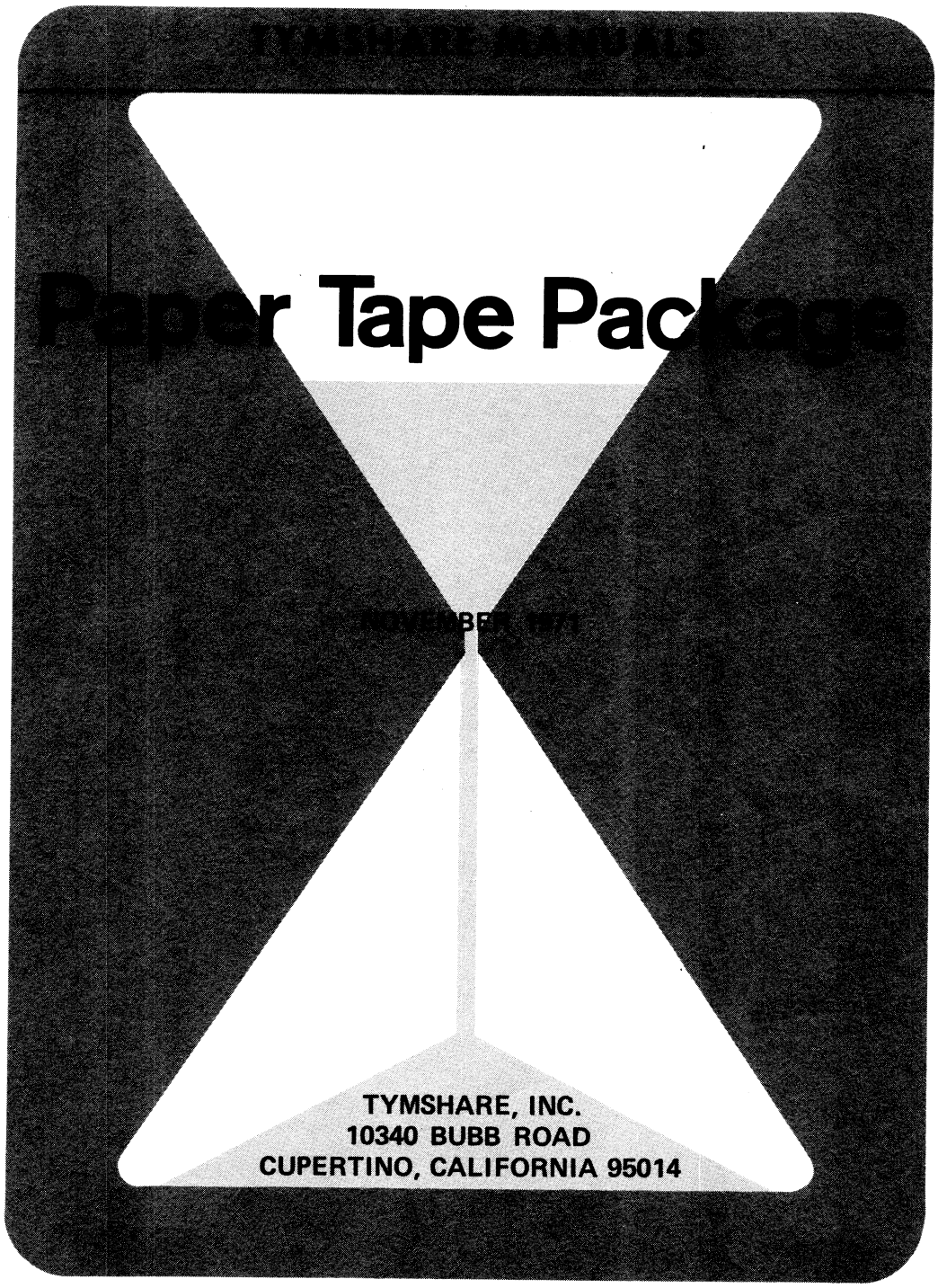


TYMSHARE®

CONTENTS

SECTION 1
INTRODUCTION

The Tymshare system has outstanding paper tape handling features, providing the capability to read and punch tape and to convert character codes between input and output.

Any information stored in either symbolic or binary representation in the computer can be punched on paper tape, and either binary or symbolic paper tape can be read and the data stored in the computer.

Symbolic data stored in the computer may readily be converted to a different code before it is punched on tape, and code read from a paper tape may be converted to different character representations before it is stored in the computer.

There are many features available which extend the convenience of reading, punching, and converting paper tape. Some of these are:

- An identifying title may be punched at the beginning of a paper tape.
- Checks are made for reading and punching accuracy while the tape is being read.
- A simple recovery can be made if the program is interrupted while reading tape.
- Information may be punched on and read from a tape in blocks of words so that a minimum of reading time is lost if the computer is disconnected or interrupted while reading paper tape.
- The user may specify any conversion of symbolic code he chooses when reading and punching tape.
- To increase conversion flexibility, alternate schemes are available for file-to-tape conversion and for tape-to-file conversion.

Section 2 of this manual describes paper tape reading and punching procedures for symbolic and binary files. Section 3 explains paper tape conversion: creating a conversion table and using the conversion programs. Complete sample problems are presented in Section 4, illustrating the various conversion methods possible. Appendix A explains how to interpret the punched paper tape. It is suggested that users who are unfamiliar with interpreting punched tape read Appendix A first. A table of various conversions, including internal and external ASCII[1] octal codes, is given in Appendix B.

In all examples in this manual, everything typed by the user is underlined. The following symbols are used to indicate user-typed characters:

Carriage Return:	↵
Line Feed:	↴
Alt Mode/Escape:	⊕
Control Characters:	Superscript c; thus, D^c denotes Control D.

1 - American Standard Code for Information Interchange. The group has recently changed its name to ANSI, American National Standard Institute.

SECTION 2
PAPER TAPE READING AND PUNCHING

The following examples and explanations of paper tape reading and punching serve as an introduction to paper tape handling.

The user has information stored on the computer that he wishes to punch on paper tape. The data is located on a file named RATES. After logging in, the procedure he follows for punching the tape is:

```
-TAPE↵
```
The user calls the program from the EXECUTIVE.

```
:PUNCH↵
FROM: RATES↵
```
The user wishes to punch a tape.
The file named RATES is to be punched.

```
TITLE: INTEREST RATES↵
```
The user requests that the words INTEREST RATES be punched at the beginning of the tape as identification.

```
CONTROL-D AT END? YES↵
```
The user requests that a Control D be punched at the end of the tape as the terminating character.[1]

```
TYPE A CARRIAGE RETURN.
THEN TURN ON PUNCH.
↵
```

The procedure for reading a paper tape is very similar. In the following example, the user reads a paper tape and stores its contents on a file named TEST. Before reading the tape, the user positions the tape on the paper tape reader within the leader section or after the title, if one exists.

```
-TAPE↵
```
The user calls the paper tape program from the EXECUTIVE.

```
:READ↵
TO: TEST↵
 NEW FILE↵
```
The user wishes to read a paper tape.
The data read from the tape is to be stored in the file named TEST.

```
EDITING? NO↵
```
The user wishes all characters on the tape to be interpreted literally.

```
TURN ON READER
```

```
READ COMPLETE
304 CHARACTERS WRITTEN
```
Reading terminates when the Control D at the end of the tape is encountered, or the user types a Control D at the terminal to terminate the process.

```
:QUIT↵
```

```
-
```

NOTE: In the examples above, the files are assumed to be symbolic.

1 – Users operating half-duplex terminals with EOT enabled must use a Control Shift N in every instance, instead of the Control D, as the terminating character for tape reading.

After the user calls the TAPE program, he may request information and perform tasks using any of the following commands:

Command	Description
CAPABILITIES	Gives a description of the program.
INSTRUCTIONS	Explains how to use the program.
CREDITS	Lists the author of the program.
VERSION	Prints the current version number of the program.
HELP *or* ?	Lists the commands available and a brief description of each.
RUN	Executes the program.
READ	Executes the program to read a paper tape.
PUNCH	Executes the program to punch a paper tape.
CONTINUE	Resumes reading or punching tape.
DO	Opens and accepts commands from a specified command file.
"	Indicates a comment and is terminated with a Carriage Return.
QUIT	Returns the user to the EXECUTIVE.

NOTE: All user commands and responses must be followed by a Carriage Return unless otherwise specified.

The RUN command can be used for either reading or punching a paper tape. It is included here to maintain compatibility with the previous version of TAPE, although it is easier to use either the READ or PUNCH command, as appropriate. The RUN command causes the program to begin requesting information.

The first request is:

INPUT FROM:

The user responds with a T if he wishes to read a paper tape or with a file name if he wishes to punch a file on paper tape.

The next request is:

OUTPUT TO:

If the user wishes to read a paper tape, he responds with the name of a file on which to store the data read. To punch a paper tape, the user responds with T.

The remaining requests are identical to the READ or PUNCH requests, discussed below.

SYMBOLIC DATA

The TAPE program may be used to read paper tape and store the information read in the computer in symbolic[1] form or to read information from a symbolic file (a storage area in the computer) and punch the contents on paper tape. Several files of data may be punched on paper tape at the user's request.

The program is called from the EXECUTIVE by typing:

-TAPE↵

1 - This is the trimmed ASCII representation of characters. A table of conversions may be found on page 19.

Paper Tape Reading

Over 1,200 feet of paper tape can be read during program execution. This is four full hours of tape reading at 10 characters per second, which is equivalent to 144,000 characters.

An option is available to edit the incoming characters, similar to the editing available in the EDITOR APPEND mode. The valid editing characters are Control A, Q, W, and V.[1]

When the system prints

EDITING?

the user responds with YES or NO followed by a Carriage Return. If the user responds YES, the incoming data is edited according to the control characters on the tape. If the user responds NO, the control characters are accepted literally. The only way to store a Control D or a Control V from tape onto a file is to precede either character with a Control V. Rubouts on paper tape are ignored and are not stored on the file.

A file written by TAPE does not undergo multiple blank compression; therefore, the file sizes and checksums[2] may differ if a previously created file is compared with that same file written in TAPE.

The paper tape reading process is:

```
-TAPE↵          The user calls TAPE in the EXECUTIVE.

:READ INVENTORY↵    The user types on one line the READ command and the
 NEW FILE↵          file on which the data from the tape is to be written.

EDITING? YES↵       The user wants certain control characters on
                    the tape to perform an editing function.
TURN ON READER      The user turns on the paper tape reader.

READ COMPLETE       The program encounters a Control D on the tape, or the user types
234 CHARACTERS WRITTEN  a Control D from the terminal. Users at half-duplex terminals with
                    EOT enabled must type a Control Shift N instead.
:QUIT↵

-
```

If the user is subject to frequent telephone disconnect problems, he may circumvent that problem by periodically turning off the paper tape reader and typing a Control D. TAPE then writes what it has read to the output file. After each Control D, the system appends the data read to the output file. The user should not manually adjust the tape in the reader. To proceed, the user types CONTINUE after he receives the system prompt (:) and turns on the reader again when requested. For example,

1 - Consult the *Tymshare EDITOR Manual* for details concerning the editing functions of these control characters.

2 - A checksum is a unique number associated with a particular file content. The checksum is calculated by adding the binary equivalent of all the words on the file.

```
-TAPE↵

:READ↵                    The user requests to read a tape and store the contents in the file SURVEY.
TO: SURVEY↵
 NEW FILE↵

EDITING? YES↵

TURN ON READER
                          The user turns off the reader and types a Control D.

READ COMPLETE
413 CHARACTERS WRITTEN

:CONTINUE↵                The command is given to read and append more data from the tape.

TURN ON READER
                          The user turns off the reader and types a Control D.

READ COMPLETE
602 CHARACTERS WRITTEN

:CONTINUE↵                The user gives the CONTINUE command.

TURN ON READER


READ COMPLETE             The program encounters the terminating Control D on the tape.
888 CHARACTERS WRITTEN

:QUIT↵

-EDITOR↵                  After the tape reading is completed, the file created by periodically
*READ SURVEY↵             turning off the reader contains the total number of characters read.
1903 CHARACTERS
*QUIT↵

-
```

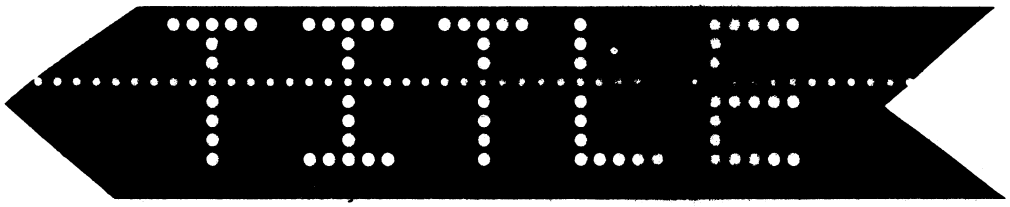
The contents of the tape are not printed on the terminal while the tape is read in full-duplex mode. If the user desires hard copy of the tape contents, he may simply copy the resultant file to the terminal after reading. Terminals operating in half-duplex mode do print the contents as the tape is being read.

Users who produce tapes on local should note that the Line Feed/Carriage Return combination on tape is written on a file as a Line Feed, whereas the Carriage Return/Line Feed combination is written as a Carriage Return on the file.

Paper Tape Punching

Several convenient features are available to the user when he wishes to punch a tape with the TAPE program.

A title of readable characters may be punched on the tape to identify it to the user. For example,



These characters are, of course, not to be read by the computer, but by the user to identify his tape. A title may contain as many as 120 characters.

If a single file is punched, the file is separated from the title by blank leader and is followed by blank trailer. There is a three-second delay after all trailers, during which time the user should turn off the paper tape punch to avoid extra characters being punched at the end of the trailer, namely a Carriage Return and a colon. If the user wishes to punch several files, he should use the program's multiple file option rather than punching one single file after another. Using the multiple file option causes the files to be separated by blank tape.

A feature is available for full-duplex terminals to suspend punching temporarily to allow loading a new roll of tape in the punch.

The program punches a Control D after the last file at the option of the user. The Control D is used to terminate tape reading. Users operating half-duplex terminals with EOT enabled must bypass this option. They must manually type the Control Shift N to indicate the end of tape reading.

The paper tape punching process is:

```
-TAPE↵
```
The user calls the program from the EXECUTIVE.

```
:PUNCH↵
FROM: SURVEY↵
```
The user gives the command to punch a tape. He may specify the file to be punched on the same line or on the next line.

```
TITLE: SURVEY DATA↵
```
The user wishes a title to be punched at the beginning of the tape.

```
CONTROL-D AT END? YES↵
```
The user requests that a Control D be punched at the end of the tape.

```
TYPE A CARRIAGE RETURN.
THEN TURN ON PUNCH.
↵
```

The characters appear on tape in 7-bit ASCII code with the eighth level always punched.[1]

1 - See Appendix A for an explanation of interpreting punched tape.

If the user wishes to punch more than one file, he types the names of the files to be punched, terminating each file name with a Line Feed and the last file name with a Carriage Return. For example,

```
-TAPE↵

:PUNCH↵          The user types the PUNCH command.
FROM: SURVEY↓    Each file name is followed by a Line Feed except the last, which is followed by
INVENTORY↓       a Carriage Return.
TEST↵

TITLE:↵          The user chooses to bypass the title option.

CONTROL-D AT END? YES↵    The user requests that a Control D be punched at the end of the last file.

TYPE A CARRIAGE RETURN.
THEN TURN ON PUNCH.
↵                The user types a Carriage Return and turns on the punch.
```

NOTE: The first file name may be on the same line as the PUNCH command; in this case, the line must be terminated with a Line Feed to indicate the multiple file option.

The user may halt punching at any time by typing an S. The system response is not immediately apparent, and several feet of tape are still punched. Acknowledgment of the suspended state is a trailer of Rubouts[1] (all positions punched) on the tape. The user has a three-second delay time in which to turn off the punch; otherwise, the system prompt (:) is punched on the tape. To proceed after the halt, the user types CONTINUE and turns the punch on when requested to do so. A leader of Rubouts is punched to distinguish the continue condition, and punching of the file(s) resumes normally.

Using a Command File With TAPE[2]

The user may call the TAPE program from a command file or call TAPE from the EXECUTIVE and specify the name of the command file with the DO command. For example,

```
-TAPE↵

:DO↵
FROM: COMFILE↵    The program requests the name of the command
                  file if it is not included in the DO command.
```

When the user uses the RUN command (instead of READ or PUNCH) in the command file, the INPUT FROM: and OUTPUT TO: replies may be entered from the terminal or from the command file. All other replies can only be taken from the command file. Of course, the final Carriage Return that begins punching is entered from the terminal, as well as the Control D that terminates reading a tape that does not have a Control D punched on it.

1 - Rubouts are ignored when the tape is read.

2 - Command files are described in the *Tymshare EXECUTIVE Reference Manual.*

To indicate that the INPUT FROM: and OUTPUT TO: replies are to be taken from the terminal rather than the command file, the user types T after the RUN command in the command file. Otherwise, he types COM after the RUN command to indicate the input/output file names are to be taken from the file.

Comments may be used in the command file by preceding the comment with a double quote mark (") and terminating the comment with a Carriage Return. This feature is particularly useful when special instructions should be printed on the terminal when TAPE is run from a command file.

Alt Mode/Escape control is available. Typing an Alt Mode/Escape closes the command file and returns control to the TAPE command level. During program execution, the program prints ABORTED after an Alt Mode/Escape. When punching a tape, a Rubout trailer is punched. When reading a tape, the Alt Mode/Escape is accepted literally and may garble a character on the file if it is typed before READ COMPLETE. The user should type a Control D, followed by an Alt Mode/Escape, instead. It is not possible to use the CONTINUE command if the Alt Mode/Escape is typed during the RUN, READ, or PUNCH command. At any other time, a CONTINUE is allowed after the Alt Mode/Escape, if it is logical to do so.

The examples shown below illustrate the procedure for punching the contents of AFILE on paper tape and two corresponding command files that perform the same function using the RUN command.

Running Program	**Command File** (with all commands taken from the file)	**Command File** (with INPUT FROM: and OUTPUT TO: replies taken from the terminal)
–TAPE↵	TAPE↵	TAPE↵
:RUN↵	RUN↵ COM↵ (Input/output file names are taken from this file.)	RUN↵ T↵ (Input/output file names are taken from the terminal.)
INPUT FROM: AFILE↵	AFILE↵	
OUTPUT TO: T↵	T↵	
TITLE: ACCOUNT FILE↵	ACCOUNT FILE↵	ACCOUNT FILE↵
CONTROL D AT END? YES↵	YES↵	YES↵
TYPE A CARRIAGE RETURN. THEN TURN ON PUNCH. ↵		
:QUIT↵	QUIT↵ COMMAND T↵	QUIT↵ COMMAND T↵
–	(Commands are to be taken from the terminal again.)	

When the command file consisting of the commands given on the right above is executed, it requests two entries from the terminal: the name of the input file (AFILE) and the output file (T). If the name of the command file is CMD, the execution is as follows:

```
-COMMAND CMD↵

INPUT FROM: AFILE↵

OUTPUT TO: T↵

TYPE A CARRIAGE RETURN.
THEN TURN ON PUNCH.
↵
```

The command file shown below produces the same results using the PUNCH command. The user calls TAPE first and then specifies the command file with the DO command.

```
-COPY COMFILE TO T↵        COMFILE is the name of the command file.

PUNCH
AFILE
ACCOUNT FILE
YES
QUIT
COMMAND T

-TAPE↵          The user calls the TAPE program.

:DO COMFILE↵    The user gives the DO command and indicates that the remaining
                responses are written on the command file specified.
TYPE A CARRIAGE RETURN.
THEN TURN ON PUNCH.
↵
```

The data is punched on paper tape and printed on the terminal.

The procedure is similar for using a command file to read a paper tape. For example, the command file named COMREAD reads a paper tape and stores the contents in FILEX.

```
-COPY COMREAD TO T↵

TAPE
READ
FILEX
↵           The user's response for the NEW FILE/OLD FILE message.
YES
QUIT
COMMAND T
```

```
-COMMAND COMREAD

TURN ON READER


READ COMPLETE
308 CHARACTERS WRITTEN

-
```

The command file returns the user to the EXECUTIVE, and the system accepts commands from the terminal again.

BINARY DATA

The BINTAPE program may be used to punch the contents of a binary file on paper tape and, at a later time, to read the resulting tape back into a file. The tape is punched in a special format to include extra features, such as checking for correct reading of tape and the ability to recover the data read before an interruption or disconnection. BINTAPE may be used with full-duplex terminals only.

After the user calls BINTAPE, he may request information and perform tasks using the following commands which are defined on page 4.

CAPABILITIES	VERSION
INSTRUCTIONS	HELP *or* ?
CREDITS	RUN
CHARGES	QUIT

The user may also use the RECOVER command to recover from an interruption or disconnection that occurred while reading tape. The recovery procedure is described on page 14.

NOTE: BINTAPE reads and punches BIN type files rather than GO files.[1]

Paper Tape Punching

A title of readable characters may be punched on the tape to identify it. This title, which is optional, is followed by "blocks" of information and checksums for each block.

The exact format of the punched tape is:

1. Title (if specified)
2. Blank leader
3. An hourglass symbol
4. A blank frame of tape
5. The size of the next block of data (the size is 128 words unless there are fewer words remaining in the file)

1 - See the *Tymshare EXECUTIVE Reference Manual* for a description of BIN and GO files.

6. A block of data
7. The checksum of the block just punched (items 3 through 7 are repeated until the end of the file is reached)
8. An hourglass symbol
9. Blank trailer

If the user wishes to punch several files, he should use the program's multiple file option, which causes the files to be separated by blank tape only. The command format is identical to that used in punching several symbolic files; that is, each file name is followed by a Line Feed, and the last file name is terminated with a Carriage Return.

The procedure used to punch a binary file is:

```
-BINTAPE↵          The user calls the program in the EXECUTIVE.

:RUN↵              The user gives the RUN command to begin execution.

INPUT FROM MODEL↵  The user types the name of the binary file to be punched.

OUTPUT TO T↵       T indicates paper tape punch.

TITLE: MODEL DATA↵ The user specifies a title for the tape.

TURN ON PUNCH.
THEN TYPE CARRIAGE RETURN.
↵
```

The user turns on the paper tape punch, types a Carriage Return, and the tape is punched.

Paper Tape Reading

When reading is specified, each block of data (128 words) read from the paper tape is stored on the output file. Accordingly, no more than 128 words are lost if there is an interruption or disconnection while the tape is being read.

There are checks made while reading the paper tape to ensure that no characters are lost or transmitted incorrectly during the transfer of data. A checksum is calculated for each block as it is read, and this checksum is compared to the checksum punched on the tape for that block of information. If the checksums match, the data is stored on the output file; otherwise, the data must be reread. The data read from the tape is not printed on the terminal while it is being read.

If there is an interruption or disconnection while the tape is being read, it is a simple matter to recover all blocks of information that have been read successfully and to continue reading beginning with the interrupted block.

Over 300 feet of paper tape can be read during one execution of BINTAPE. This is one full hour of tape reading.

To read a binary tape, the user positions the tape in the reader within the blank leader section after the title (if there is a title) or on the first hourglass. Then the procedure is:

```
-BINTAPE↵
```
The user calls the program in the EXECUTIVE.

```
:RUN↵
```
The user gives the RUN command to begin execution.

```
INPUT FROM T↵
```
The user specifies terminal input, indicating he wishes to read a tape.

```
OUTPUT TO HIGHWAY↵
 NEW FILE↵
```
The user types the name of the file to which the data on the tape is to be written.

```
TURN ON READER.
```
The user turns on the paper tape reader and the tape is read.

```
NUMBER OF WORDS WRITTEN IS 1665

:QUIT↵

-
```

NOTE: The tape read by BINTAPE must be in the same format as tape punched by BINTAPE.

As each block of information is read from the tape, a checksum is calculated for that block and compared to the checksum punched on the tape. If the checksums match, the data block is stored on the output file. If the checksums do not match, an error message is printed and the user must:

1. Turn off the reader.
2. Reposition the tape before the current block (within the leading hourglass symbol).
3. Turn on the paper tape reader.

If checksums for a block of data continually mismatch (this happens only if there was an error in punching the tape or if the tape is torn), the user can terminate reading by positioning the tape on the last hourglass.

To terminate tape reading early for any reason, the user may either:

1. Induce a checksum mismatch and reposition the tape on the hourglass, or
2. Turn off the paper tape reader while an hourglass is being read and reposition the tape on the last hourglass.

NOTE: BINTAPE does not print the tape contents on the terminal during reading. However, a Control Q or Control S is typed every three seconds during reading or while waiting for input. These characters do not print on the paper and affect only terminals which have the X-ON/X-OFF capability. Control Q turns on the tape reader, and Control S turns it off to regulate tape reading on these terminals. Terminals that do not have the X-ON/X-OFF feature are not affected by the Control Q and Control S.

If there is an interruption or disconnection while the tape is being read, the BINTAPE program may be called again. The user may then continue reading the tape at the interrupted block by using the RECOVER command. For example, the user wishes to read a paper tape into the file named HIGHWAY.

```
-BINTAPE↵

:RUN↵

INPUT FROM T↵

OUTPUT TO HIGHWAY↵          The previous contents of HIGHWAY are erased.
 OLD FILE↵

TURN ON READER.
```

The program reads two blocks, but the reading is interrupted in the third block. The user recovers as follows:

```
-BINTAPE↵          The user calls the program again.

:RECOVER↵          He gives the RECOVER command.

PREVIOUS OUTPUT FILE: HIGHWAY↵     The user types the name of the file which contains the
                                   data written prior to the interruption.
NEW OUTPUT FILE: HIGHWAY1↵         He gives a new name for the file on which the data on the
 NEW FILE↵                         paper tape is to be stored.

POSITION TAPE BEFORE BLOCK 3.      The user positions the tape in the hourglass symbol
THEN TURN ON READER.               before block 3 and turns on the reader, and the
                                   program reads the rest of the tape.

NUMBER OF WORDS WRITTEN IS 1665

PREVIOUS OUTPUT FILE MAY NOW BE DELETED.

:QUIT↵

-
```

HIGHWAY1 contains the entire contents of the tape; the user may delete HIGHWAY.

Using a Command File With BINTAPE[1]

When the BINTAPE program is called from a command file, the user has the option of entering the replies to INPUT FROM and OUTPUT TO from either the command file or from the terminal. This option is indicated by adding either COM or T after the RUN command in the command file.

The examples below demonstrate reading a paper tape into the file BINDATA and two corresponding command files which perform the same function.

Running Program	Command File (with all commands taken from the file)	Command File (with INPUT FROM and OUTPUT TO replies taken from the terminal)
-BINTAPE↵	BINTAPE↵	BINTAPE↵
:RUN↵	RUN↵ COM↵ (Input/output file names are taken from this file.)	RUN↵ T↵ (Input/output file names are taken from the terminal.)
INPUT FROM T↵	T↵	
OUTPUT TO BINDATA↵ NEW FILE↵	BINDATA↵ ↵	
TURN ON READER. NUMBER OF WORDS WRITTEN IS 306 :QUIT↵	QUIT↵ COMMAND T↵	QUIT↵ COMMAND T↵
-	(Commands are to be taken from the terminal again.)	

#IOBIN

The #IOBIN program, in the Tymshare User Program Library, allows the user to read or punch any binary tape. The program writes from one to three paper tape frames per number, depending on the user's specification. In addition, #IOBIN may be used to read or punch symbolic data and convert the internal or external code read to the corresponding external or internal code in a number system specified by the user.

For further information, the user may call #IOBIN in the EXECUTIVE and give the CAPABILITIES and INSTRUCTIONS commands, or ask his Tymshare representative.

1 - Command files are described in the *Tymshare EXECUTIVE Reference Manual.*

SECTION 3
PAPER TAPE CONVERSION

Any symbolic file may be read, converted to another code, and punched on tape in one simple process using the Tymshare system. Also, a paper tape may be read, the code converted, and the converted contents written on a file.

As an introduction to the programs that perform these tasks, simple examples of the most common use of the programs CONFILE and CONTAPE are presented below.

In the examples, the name of a conversion table is requested by the program. Assume that the conversion table has already been created and is named EXTAB. The format and construction of conversion tables are explained in detail on page 18.

NOTE: Any of the following standard commands can be used with CONFILE and CONTAPE: INSTRUCTIONS, CAPABILITIES, CREDITS, VERSION, HELP (or ?), RUN, and QUIT.

In addition, the SUPPRESS command is available for users operating from a half-duplex terminal. After calling CONFILE or CONTAPE, the user must first give the SUPPRESS command in response to the system prompt (:) to avoid X-ON/X-OFF during reading. Then, the procedure is:

```
-CONFILE↵        CONFILE is called in the EXECUTIVE.

*RUN↵            The user gives the RUN command to execute the program.

INPUT FROM MARTY↵    This is the file to be converted and punched.

OUTPUT TO T↵     T indicates the terminal (the paper tape punch).

NAME OF TABLE FILE: EXTAB↵    The user names the file containing the conversion table.

TITLE: THANKS↵   The word THANKS is punched on the tape for identification.

TURN ON PUNCH.
THEN TYPE CARRIAGE RETURN.
↵
```

The CONFILE program is usually used for file-to-tape conversion. That is, it is usually requested to convert characters read from an input file before punching them on tape.

CONTAPE, on the other hand, is usually used to convert codes read from a paper tape before writing them on an output file.

```
-CONTAPE↵          The user calls CONTAPE in the EXECUTIVE.

:RUN↵              The RUN command executes the program.
INPUT FROM T↵      T indicates the terminal (the paper tape reader).
OUTPUT TO TRIAL↵   The converted data is stored in the file TRIAL.
 NEW FILE↵
NAME OF TABLE FILE: CONTAB↵   CONTAB is the name of the conversion table file.
TURN ON READER.


NUMBER OF CHARACTERS WRITTEN IS 299
:QUIT↵
-
```

THE CONVERSION TABLE

The conversion table is used to redefine the symbolic representations of characters. This is the essential component needed to operate CONFILE and CONTAPE. In the conversion table, the user specifies the representation desired for each character present in the data.

ASCII Codes

To construct the table, the user must know the ASCII codes for each character. Each character has two codes associated with it. A character is represented on a file by an octal number known as the internal ASCII code. On paper tape, a character is represented by an external ASCII octal code.

A table of internal and external ASCII octal codes for all printing characters on the Model 33 Teletype is listed below.

Internal ASCII Octal Code (file)	Character	External ASCII Octal Code (paper tape)
0	blank	40
1	!	41
2	"	42
3	#	43
4	$	44
5	%	45
6	&	46
7	'	47
10	(	50
11	)	51
12	*	52
13	+	53
14	,	54
15	–	55
16	.	56
17	/	57
20	0	60
21	1	61
22	2	62
23	3	63
24	4	64
25	5	65
26	6	66
27	7	67
30	8	70
31	9	71
32	:	72
33	;	73
34	<	74
35	=	75
36	>	76
37	?	77
40	@	100
41	A	101
42	B	102
43	C	103
44	D	104
45	E	105

Internal ASCII Octal Code (file)	Character	External ASCII Octal Code (paper tape)
46	F	106
47	G	107
50	H	110
51	I	111
52	J	112
53	K	113
54	L	114
55	M	115
56	N	116
57	O	117
60	P	120
61	Q	121
62	R	122
63	S	123
64	T	124
65	U	125
66	V	126
67	W	127
70	X	130
71	Y	131
72	Z	132
73	[	133
74	\	134
75	]	135
76	↑	136
77	←	137
152	Line Feed	12
155	Carriage Return	15
137	Rubout	177

Codes for control characters can be obtained by adding 100 to the internal ASCII octal code and subtracting 100 from the external ASCII octal code for the specific alphabetic character. For example, the internal ASCII octal code for Control A is 141; the external code is 1.

The complete range of these codes is 0—377 octal, that is, 256 different representations. Codes not accounted for above are normally not generated from a Teletype Model 33 keyboard, but can be created within the system (such as with CONFILE and CONTAPE) and from newer types of terminals. A code of 400 may be used in the conversion table to ignore any specified character.

Paper tape punched on the Tymshare system (from a COPY command or with the TAPE program) always has the eighth channel punched.[1] Tape punched off line may or may not have the eighth channel punched, depending on the type of terminal being used. Tape punched off line on certain Teletype terminals has the eighth channel punched. Thus, the codes appear on tape as 200 plus the external ASCII octal code. For example, A is punched as 301, 9 is punched as 271, Control Q is punched as 221, a Rubout is punched as 377, etc.

The TAPE program ignores the eighth level; therefore, it can read paper tape punched in the standard external code, as well as tape punched on other systems where the eighth level is used for even or odd parity.

The conversion programs read tape exactly as it is punched, so the user must construct his conversion table accordingly. Tape is punched exactly as it is specified in the conversion table. The eighth level is not punched automatically.

The conversion table is a list of octal numbers. Each entry in the list corresponds with a particular position in the list. The positions range from 0 octal for the first entry to 377 octal for the 256th entry. The user should visualize the table as two columns in order to understand how the conversions are made. For example,

Conversion Table	(Position)
2	0
7	1
15	2
44	3
6	4
6	5
0	6
0	7
377	10
⋮	⋮
0	375
0	376
0	377

General Rules

The rules given below define the contents and structure of the conversion tables used with CONFILE or CONTAPE:

1. The table may be up to 256 entries long, since the positions correspond to the codes 0 to 377 octal (256 decimal or 400 octal entries).
2. If the table entered contains fewer than 256 entries, the CONFILE or CONTAPE program fills the rest of the table with zero entries.
3. If the table entered contains more than 256 entries, the CONFILE or CONTAPE program reads only the first 256 entries, prints a message informing the user of this, and continues.

1 – See Appendix A for an explanation of channels on paper tape.

4. Each entry must be an octal integer value (from 0 to 377) terminated by a comma, a space, or a Carriage Return. For example, any of the following is permitted:

   0,1,41,6

   or

   0 1 41 6

   or

   0
   1
   41
   6

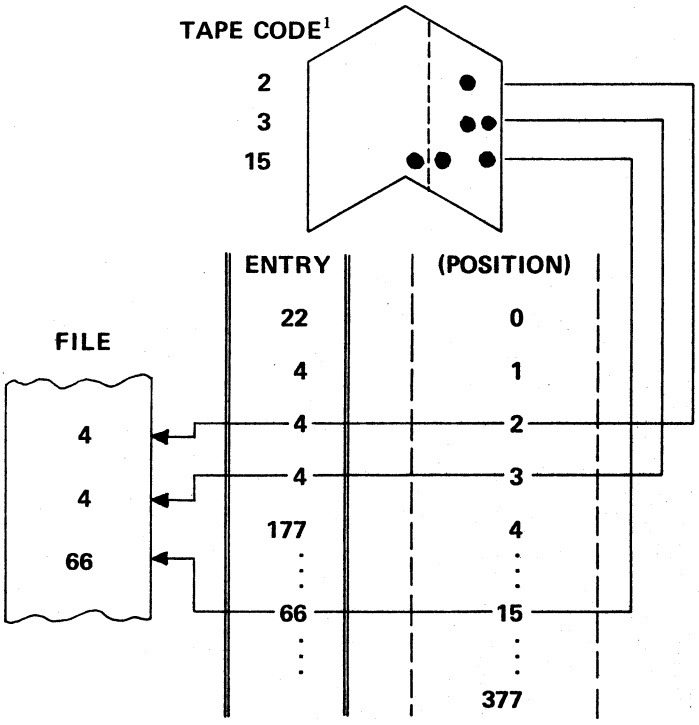
Each of these three conversion tables has an entry of 0 for position 0, 1 for position 1, 41 for position 2, and 6 for position 3.

Methods of Conversion

There are four methods of conversion possible in the Tymshare system: tape-to-file conversion (a tape is read, its contents converted according to the conversion table, and the converted information written on a file) using CONTAPE, tape-to-file conversion using CONFILE, file-to-tape conversion using CONFILE, and file-to-tape conversion using CONTAPE.

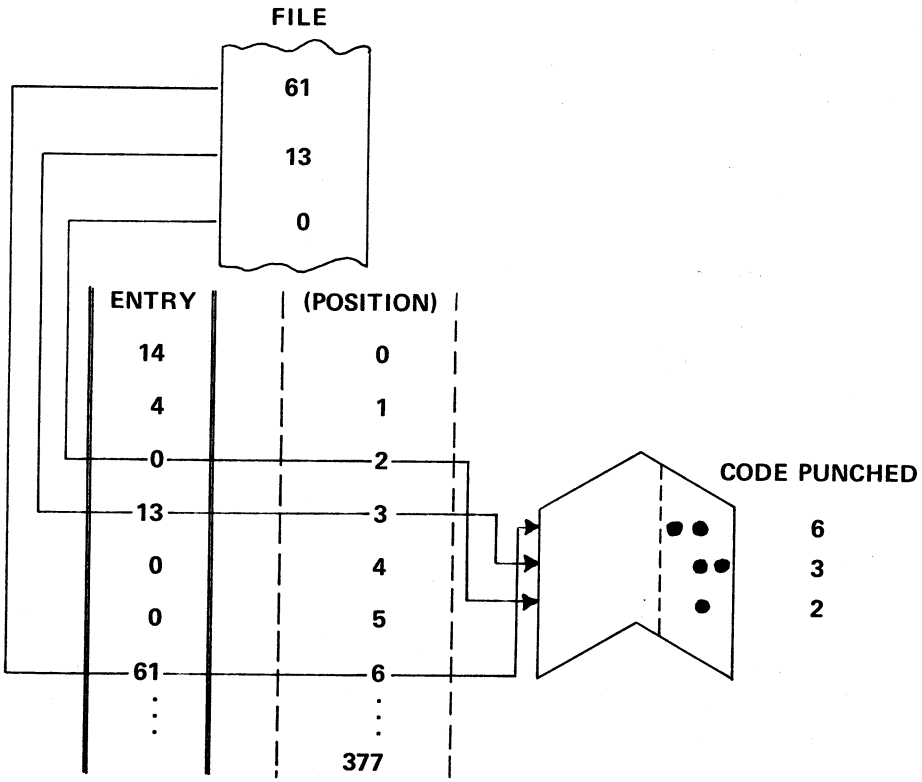
The table is read by the programs in two different ways, depending on the method of conversion selected by the user. The diagrams shown below illustrate how the table is read for all four methods of conversion.

CONTAPE Tape-to-File Conversion

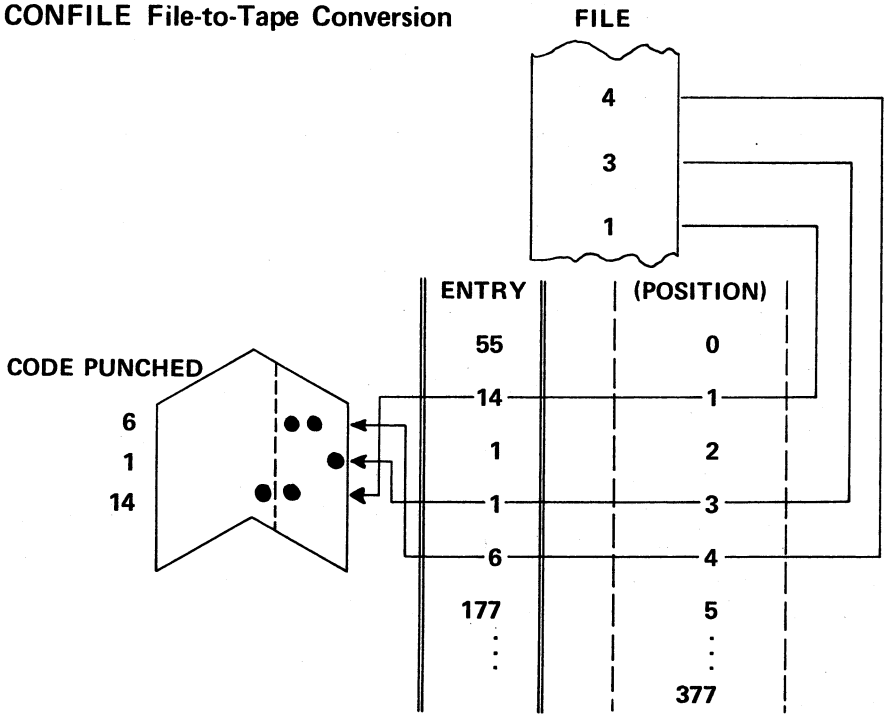


1 – See Appendix A for an explanation of reading codes punched on tape.

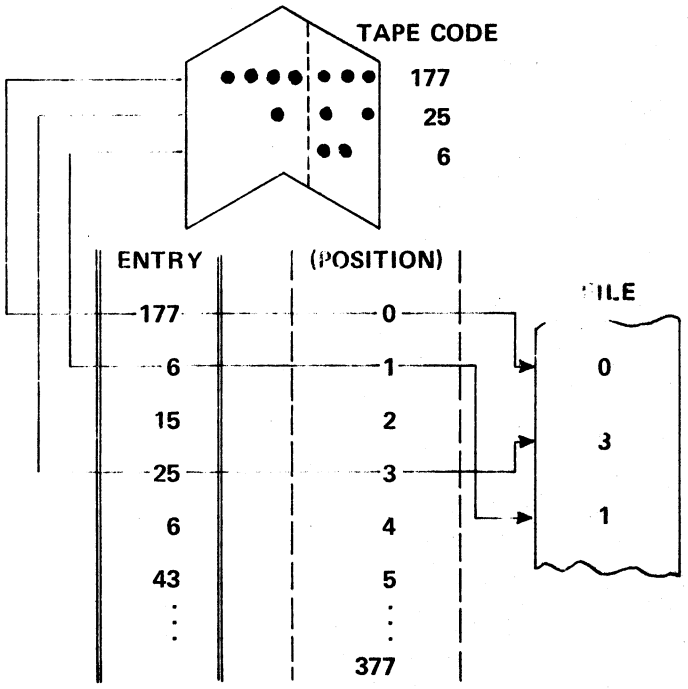
CONTAPE File-to-Tape Conversion



CONFILE File-to-Tape Conversion



CONFILE Tape-to-File Conversion



When programs are used for their primary function, that is, CONFILE for file-to-tape conversion or CONTAPE for tape-to-file conversion, the code read (either on the file or the tape) directs the program to the position equal to the code read. The entry at that position is the code that is punched or written. For example,

Code Read (on file or tape)	Conversion Table Entry	(Position)	Code Written or Punched
0	2	0	2
4	4	1	56
6	6	2	1
2	10	3	6
2	56	4	6
	0	5	
	1	6	

This is the actual table.

However, when the programs are used for their secondary capability, that is, CONFILE for tape-to-file conversion or CONTAPE for file-to-tape conversion, the code read (on file or tape) is sought as an entry in the table. The program starts reading the table at the top and locates the first occurrence of that code. The position of that code is the code that is punched or written. For example,

Code Read (on file or tape)	Conversion Table Entry	(Position)	Code Written or Punched
0	2	0	5
4	4	1	1
6	6	2	2
2	10	3	0
2	56	4	0
	0	5	
	1	6	

Creating a Conversion Table With the TABLEMAKER Program

The TABLEMAKER program is designed to help the user create a conversion table to be used with CONFILE or CONTAPE or to modify an existing conversion table.

In using TABLEMAKER, the user specifies the position number and its corresponding entry. All unspecified positions have an entry of zero.

The example on the following page demonstrates the use of TABLEMAKER in creating the conversion table XTABLE.

```
-TABLEMAKER↵

:RUN↵

DO YOU WANT TO REVISE AN EXISTING TABLE? NO↵

TYPE NAME OF TABLE FILE YOU WANT TO CREATE: XTABLE↵

BEGIN:
? 2,22↵        Position 2 is given an entry of 22.
? 4,24↵        Position 4 is given an entry of 24.
? 5-10,"↵      Positions 5, 6, 7, and 10 are given entries equal to their position numbers.
? 23,43↵       Position 23 is given an entry of 43.
? 24-26,117↵   Positions 24, 25, and 26 are given entries of 117.
? LIST↵        The user gives the LIST command, and the system prints all entries up to the last non-zero entry.
   0     <POSITION 0>
   0
  22
   0
  24
   5
   6
   7
  10     <POSITION 10>
   0
   0
   0
   0
   0
   0
   0
   0     <POSITION 20>
   0
   0
  43
 117
 117
 117
? QUIT↵        The conversion table is created when the user types QUIT.  Instead of typing QUIT, the user may make
               additional entries to the table by continuing as above.  In this case, positions 27–377 are given entries of

:QUIT↵

-
```

The rules for making table entries are listed below:

1. The user types the position (the range is 0 through 377 octal), a comma, the code to be entered for that position, and a Carriage Return.
2. A dash (—) may be used to indicate a range of positions. In the example
   ? 24—26,117↵
   the program places 117 in positions 24, 25, and 26.
3. A double quote mark may be used to indicate that the entry for the position or positions specified should be equal to the position number itself. In the example
   ? 5—10,"↵
   the program places 5 in position 5, 6 in position 6, etc.
4. All entries not specified by the user are set to zero.
5. The entries may be listed up to the last non-zero entry by typing LIST and a Carriage Return after the question mark.
6. The user ends the entries by typing QUIT (or Q) and a Carriage Return after the question mark. The conversion table is created when the user types QUIT.

The following examples show some of the errors that cause the message

ERROR. PLEASE RETYPE.

to be printed.

```
? 14.,47↵      Decimal points are not permitted.
ERROR. PLEASE RETYPE.
? 501,47↵      The position cannot exceed 377.
ERROR. PLEASE RETYPE.
? 14,501↵      The entry cannot exceed 400.
ERROR. PLEASE RETYPE.
? 14,48↵       Numbers must be in octal.
ERROR. PLEASE RETYPE.
? 14;47↵       The position and entry must be separated by a comma.
ERROR. PLEASE RETYPE.
? 14,47↵       The information is typed correctly.
?
```

The TABLEMAKER program may be used to revise an existing conversion table. In the example on the following page, the user revises the table XTABLE, created on the previous page, and writes the resulting conversion table on a file named NEWTABLE.

```
-TABLEMAKER↵

:RUN↵

DO YOU WANT TO REVISE AN EXISTING TABLE? YES↵

TYPE NAME OF TABLE FILE YOU WANT TO REVISE: XTABLE↵

TYPE NAME OF TABLE FILE YOU WANT TO CREATE: NEWTABLE↵

BEGIN:
? 15,35↵
? 11,"↵
? 7,117↵
? LIST↵
  0     <POSITION 0>
  0
 22
  0
 24
  5
  6
117
 10     <POSITION 10>
 11
  0
  0
  0
 35
  0
  0
  0     <POSITION 20>
  0
  0
 43
117
117
117
? QUIT↵

:QUIT↵

-
```

The user types three positions and their corresponding entries.

The new table contains all the entries from the old table, as well as new entries for positions 15, 11, and 7.

The user gives the QUIT command to terminate the table entry mode and create the conversion table NEWTABLE.

The user types QUIT again to return to the EXECUTIVE.

The user could have named the revised table XTABLE if there were no reason to save the original conversion table. The contents of the revised table would have been written over the old table.

A complete list of instructions may be obtained by typing

:INSTRUCTIONS↵

while in TABLEMAKER. The other standard commands available are: RUN, VERSION, CREDITS, CHARGES, CAPABILITIES, HELP (or ?), and QUIT.

CONFILE: ITS USE, OPTIONS, AND EXECUTION

CONFILE is designed primarily to convert code read from a file before punching it on tape. The advantages of using CONFILE in the file-to-tape conversion mode are:

- Several different file characters can be punched as the same tape code.
- Any number of different file characters can be ignored.

CONFILE's secondary capability is tape-to-file conversion. This option is convenient for users who want to read tape that has been punched by CONFILE using the same conversion table. The secondary option is also useful in finding errors because it performs an error check as the data is converted.

CONFILE File-to-Tape Conversion

The general procedure for punching tape using CONFILE is as follows:

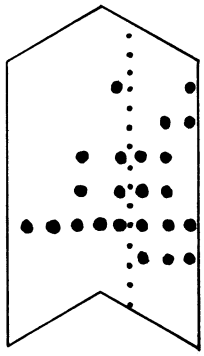
```
-CONFILE↵        The user calls the program in the EXECUTIVE.

:RUN↵            The user gives the RUN command to execute the program.

INPUT FROM DATA↵     This is the file to be converted and punched.

OUTPUT TO T↵         T indicates the terminal's paper tape punch.

NAME OF TABLE FILE: CONTABLE↵     This is the file containing the conversion table.

TITLE: TEST DATA↵    The user specifies a title to be punched on the tape for identification.

TURN ON PUNCH.
THEN TYPE CARRIAGE RETURN.
↵
```

The conversion table determines what code is punched on the tape for each code read in the input file. The input file is written in internal ASCII octal codes. The conversion table is a list of external ASCII octal codes to be punched for all internal codes read. The code read on the file specifies the position in the table that contains the code to be punched on the tape.

Ignoring Characters

An external code of 400 specifies that nothing is to be punched. An entry of 400 may be used to ignore any codes the user chooses. In the example below, a code of 2 is ignored when the user specifies a code of 400 for the external code.

Code Read From Input File	Conversion Table Entry	(Position)	Code Punched on Tape
0	11	0	11
3	56	1	3
5	400	2	56
1	3	3	56
6	7	4	377
2	56	5	7
4	377	6	
	⋮	⋮	



Converting the Multiple Blank Character

It is the convention in the Tymshare system to treat the octal code 135 as a multiple blank character. The multiple blank code is created when more than two blanks are typed. That is, 135 followed by 6 is interpreted as six blanks. The code 135 is treated this way regardless of what is in the conversion table. However, if the user has a special case and wishes to treat 135 as an ordinary character, he can specify this as an option when running CONFILE. For example,

```
-CONFILE↵

:RUN↵

INPUT FROM DATA↵

OUTPUT TO T↵

NAME OF TABLE FILE: CONTABLE↓        The user types a Line Feed after entering the
DO YOU WANT TO CONVERT THE MULTIPLE BLANK CHARACTER? YES↵   conversion table file to request the option.

TITLE: TEST TWO↵

TURN ON PUNCH.
THEN TYPE CARRIAGE RETURN.
↵
```

CONFILE Tape-to-File Conversion

Tape-to-file conversion is the secondary process of CONFILE. This process is a convenient way to perform error checking while reading and converting tape. It is also useful to the user who has punched tape with CONFILE and wants to read it back into the system using the same conversion table.

When CONFILE is used for tape-to-file conversion, the code read from the tape (external code) is sought in the conversion table (list of external codes), and the first position of that entry is then written on the output file.

The example of a tape punched using CONFILE, on page 30, is used below to demonstrate how CONFILE reads and converts a tape to a file using the same conversion table.

Code Read From Tape	Conversion Table Entry	(Position)	Code Written on Output File
11	11	0	0
3	56	1	3
56	400	2	1
56	3	3	1
377	7	4	4
7	56	5	
	377	6	
	⋮	⋮	

Note, however, that all the characters written on the file are not the same as those originally read from the file. There are several differences:

1. If a code appears more than once in the table, the position of its first occurrence is written on the file. That is, the character code written on the output file for tape code 56 is always 1 because that is the first occurrence of 56 in the conversion table.
2. Code 2 is not written on the output file because it caused no corresponding code to be punched during the file-to-tape conversion.
3. The Rubout code (377) is ignored when it is read and thus causes no corresponding character to be written on the file.

The user can treat the Rubout code like any other code and convert it to a file character code if he chooses. This is accomplished by following the table file name with a Line Feed rather than a Carriage Return. For example,

```
-CONFILE↵

:RUN↵
INPUT FROM T↵        T indicates the terminal's tape reader.
OUTPUT TO DATA2↵
 NEW FILE↵
NAME OF TABLE FILE: CONTABLE↓     The user types a Line Feed after the
                                  table file name to request the option.
DO YOU WANT TO CONVERT RUBOUTS? YES↵   The user may answer NO if he
                                       typed the Line Feed by mistake.
TURN ON READER.
```

If a code is read from the paper tape that does not appear in the conversion table, an error message is printed after the entire tape is read, and all converted codes up to the character in error are written on the output file.

CONTAPE: ITS USE, OPTIONS, AND EXECUTION

CONTAPE is primarily a tape-to-file conversion program. It is advantageous to use CONTAPE in the tape-to-file mode because:

- Several different tape codes can be stored as the same file character.
- Any number of different tape codes can be ignored.

CONTAPE's secondary capability is file-to-tape conversion. This capability is a convenience for users who want to punch tape using the same conversion table that was originally used to read a tape with CONTAPE. This secondary capability can also be used to perform error checking since the system prints an error message whenever the code read from a file is not found in the conversion table.

CONTAPE Tape-to-File Conversion

The procedure for reading tape using CONTAPE is:

```
-CONTAPE↵          The user calls the program in the EXECUTIVE.

:RUN↵              He gives the RUN command.

INPUT FROM T↵      T indicates the terminal's paper tape reader.

OUTPUT TO TEST3↵   The converted data is to be written on the file specified.
 NEW FILE↵

NAME OF TABLE FILE: TABLE3↵   The user types the name of the file
                              containing the conversion table.
TURN ON READER.


NUMBER OF CHARACTERS WRITTEN IS 254

:QUIT↵

-
```

Tape reading terminates if no more data is entered within six seconds.

The user should trim the tape evenly on one of the trailing Rubouts or turn off the reader when the Rubout trailer is encountered to be sure there are no unwanted characters on the end.

The code read from the tape specifies in which table position the code to be written on the file can be found.

NOTE: If the tape has the eighth level punched (tape generated on the Tymshare system), the user must be sure to type the entries in the proper table position, that is, 200 plus the external code.

The example below demonstrates the tape-to-file conversion process using CONTAPE.

Code Punched on Tape	Conversion Table Entry	(Position)	Code Written on Output File
205	0	0	2
205	⋮	⋮	1
202	0	200	2
201	1	201	0
202	2	202	56
200	56	203	56
203	4	204	0
203	400	205	1
207	*(CONTAPE sets the*		
201	*remaining entries to*		
377	*zero.)*		

A table entry of 400 specifies that the tape code corresponding to that position is to be ignored; that is, nothing is written on the file when that tape code is read. Rubouts (377 and 177) on the tape are automatically ignored; however, the user may specify that Rubouts be treated as ordinary code. The method of requesting this option in CONTAPE is identical to the method used in CONFILE: the user types a Line Feed after entering the name of the conversion table file. (See the example on page 31.)

CONTAPE File-to-Tape Conversion

When CONTAPE is used for file-to-tape conversion, the code read from the input file is sought in the conversion table; the first position in which it is found is the code that is punched on the tape. If a code that is read from the file does not appear in the conversion table, the system prints an error message and the program terminates without punching any tape.

The example below demonstrates how the file created above is punched on tape using the same conversion table.

Code Written on File	Conversion Table Entry	(Position)	Code Punched on Tape
2	0	0	202
1	⋮	⋮	201
2	0	200	202
0	1	201	0
56	2	202	203
56	56	203	203
0	4	204	0
1	400	205	201

Note that characters that are ignored when converting the tape code to the output file code are lost and are not written on the tape (this includes Rubout codes).

When CONTAPE is used for file-to-tape conversion, the user may specify that the multiple blank character be treated as an ordinary code by typing a Line Feed rather than a Carriage Return after the table file name. The method of requesting this option is the same as is used with CONFILE in file-to-tape conversion. (See the example on page 30.)

An ASCII Conversion Table for CONTAPE

Available in the Tymshare User Program Library is a conversion table, #ASCII, which consists of internal octal codes to perform the normal conversion from external ASCII to internal ASCII representation. The table covers all eight levels of tape input; therefore, whether or not the eighth level is punched on the tape, the usual representation results. In other words, #ASCII performs the same function as the TAPE program. Users who read tape at 30 characters per second may prefer to use CONTAPE with the #ASCII conversion table; it is more accurate than the TAPE program, providing the terminal has the X-ON/X-OFF capability.

The #ASCII table is advantageous when reading paper tape that is almost ASCII. When the user desires to make a few modifications to the usual conversion from external to internal ASCII octal code, he can use #ASCII as the starting conversion table and use TABLEMAKER to revise it. The user enters the few entries that differ from the usual conversion, and the rest of the table contains the standard ASCII conversion.

For example, the user wishes to read a tape and modify the conversion of the characters 6 and L when it is written on a file. He wishes to convert 6's to 9's and L's to Q's.

Tape Code (with eighth level punched)		File Code Desired	
Character	Octal Code	Character	Octal Code
6	266	9	31
L	314	Q	61

The procedure to construct the conversion table to accomplish this is as follows:

```
-TABLEMAKER↵

:RUN↵

DO YOU WANT TO REVISE AN EXISTING TABLE? YES↵

TYPE NAME OF TABLE FILE YOU WANT TO REVISE: #ASCII↵

TYPE NAME OF TABLE FILE YOU WANT TO CREATE: ASC7↵

BEGIN:
? 266,31↵
? 314,61↵
? QUIT↵

:QUIT↵
```

The user must type a different name here for his revised ASCII conversion table.

Position 266 has an entry of 31.

USING A COMMAND FILE WITH CONFILE OR CONTAPE[1]

When CONFILE or CONTAPE is called from a command file, the INPUT FROM and OUTPUT TO replies may be entered from the command file or from the terminal. Other replies must be taken from the command file. After typing the program name and the RUN command, the user types COM in the command file if the INPUT FROM and OUTPUT TO replies are contained in the command file; he types T if they are to be entered at the terminal.

In the command file listed below, the user calls the CONFILE program and specifies that the INPUT FROM and OUTPUT TO replies are contained in the command file.

```
-COPY COMCONFILE TO T↵

CONFILE
RUN
COM
T
FILE1
↵          The response to NEW FILE/OLD FILE message.
TABLE1
QUIT
COMMAND T

-
```

1 - Command files are described in the *Tymshare EXECUTIVE Reference Manual.*

To execute the command file above, named COMCONFILE, the procedure is:

```
-COMMAND COMCONFILE↵

 NEW FILE
TURN ON READER.

NUMBER OF CHARACTERS WRITTEN IS 349

-
```

The program prints NEW FILE or OLD FILE, as appropriate, for the OUTPUT file specified. No user response is expected. (It is contained in the command file.)

SECTION 4
SAMPLE PROBLEMS

CONVERSIONS USING CONFILE

The user has a file named CASE1 which contains the following data:

```
-COPY CASE1 TO T↵
80684233349711665396614
```

He wishes to make these changes when the file is punched on paper tape:

8 changed to 4
4 changed to 8
6 changed to 3
3 changed to 6

First, the user creates the conversion table, specifying how all codes read should be converted when the file is punched.

```
-TABLEMAKER↵

:RUN↵

DO YOU WANT TO REVISE AN EXISTING TABLE? NO↵

TYPE NAME OF TABLE FILE YOU WANT TO CREATE: CASE1TAB↵

BEGIN:
? 20,60↵      Position 20 has an entry of 60.
? 21,61↵
? 22,62↵
? 23,66↵
? 24,70↵      Position 24 has an entry of 70.
? 25,65↵
? 26,63↵
? 27,67↵
? 30,64↵
? 31,71↵
? 155,15↵     Position 155 has an entry of 15.
? QUIT↵       This QUIT creates the conversion table. All unspecified positions have entries of 0.

:QUIT↵        The user gives the QUIT command to return to the EXECUTIVE.
```

This is the conversion table that the user created:

Entry	(Position)
0	0
0	1
⋮	⋮
60	20
61	21
62	22
66	23
70	24
65	25
63	26
67	27
64	30
71	31
0	32
⋮	⋮
15	155
⋮	⋮
0	377

Next, the user calls CONFILE to perform the conversions and punch the converted data on tape.

```
-CONFILE↵

:RUN↵
INPUT FROM CASE1↵        This is the name of the file to be converted.
OUTPUT TO T↵             T indicates the terminal's paper tape punch.
NAME OF TABLE FILE: CASE1TAB↵    The user specifies the conversion table.
TITLE:↵                  The user chooses to bypass the title option.
TURN ON PUNCH.           The user turns on the punch and types a Carriage Return.
THEN TYPE CARRIAGE RETURN.
↵
403482666897113356933l8  The results of the conversion are printed on the terminal.
:QUIT ↵
-
```

At a later time, the user uses CONFILE's secondary capability, tape-to-file conversion, to read the tape created previously with CONFILE and convert its contents using the original conversion table, writing the results on a file called RECONCASE1.

```
-CONFILE↵

:RUN↵
INPUT FROM T↵          T indicates the terminal's paper tape reader.
OUTPUT TO RECONCASE1↵  The converted data is to be written on the file RECONCASE1.
 NEW FILE↵
NAME OF TABLE FILE: CASE1TAB↵   This is the same table that was used in
                                the original file-to-tape conversion.
TURN ON READER.


NUMBER OF CHARACTERS WRITTEN IS 24

:QUIT↵
-COPY RECONCASE1 TO T↵   This file contains the results of the tape-to-file conversion.
80684233349711665396614
```

CONVERSIONS USING CONTAPE

The user wishes to read a paper tape and convert its contents. For purposes of this example, the tape is read first by the TAPE program so that the original data can be shown.

```
-TAPE↵

:READ↵
TO: DATA2↵
 NEW FILE↵
EDITING? NO↵
TURN ON READER

READ COMPLETE
16 CHARACTERS WRITTEN

:QUIT↵
-COPY DATA2 TO T↵

AFECDBLRVMMXZBA
```

The user wishes to make these changes:

A changed to L
L changed to A

First, the user creates the conversion table, specifying how all codes read on the paper tape should be converted when the file is written.

```
-TABLEMAKER↵

:RUN↵

DO YOU WANT TO REVISE AN EXISTING TABLE? NO↵

TYPE NAME OF TABLE FILE YOU WANT TO CREATE: DATA2TAB↵

BEGIN:
? 301,54↵        Position 301 has an entry of 54.
? 302,42↵
? 303,43↵
? 304,44↵
? 305,45↵
? 306,46↵
? 314,41↵        Position 314 has an entry of 41.
? 315,55↵
? 322,62↵
? 326,66↵
? 330,70↵
? 332,72↵
? 215,155↵
? QUIT↵          The user gives the QUIT command to create the
                 table. All unspecified positions have entries of 0.

:QUIT↵           The user gives the QUIT command to return to the EXECUTIVE.
```

NOTE: The user created the paper tape at a Teletype terminal on local; therefore, all codes have the eighth channel punched.

This is the conversion table that the user created above:

Entry	(Position)
0	0
0	1
:	:
155	215
0	216
:	:
54	301
42	302
43	303
44	304
45	305
46	306
:	:
41	314
55	315
:	:
62	322
:	:
66	326
0	327
70	330
0	331
72	332
:	:
0	377

Next, the user calls CONTAPE to perform the conversions and write the results on a file named CONDATA2.

```
-CONTAPE↵

:RUN↵
INPUT FROM T↵          T indicates the terminal's paper tape reader.
OUTPUT TO CONDATA2↵    The converted contents are written on a file named CONDATA2.
 NEW FILE↵
NAME OF TABLE FILE: DATA2TAB↵
TURN ON READER.

NUMBER OF CHARACTERS WRITTEN IS 16    Reading terminates when no more
                                      data is entered for six seconds.
:QUIT↵
-COPY CONDATA2 TO T↵   This file contains the results of the conversion.
LFECDBARVMMXZBL
```

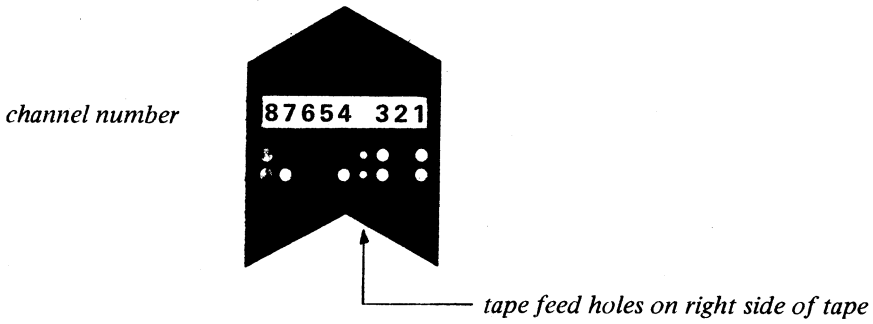
At a later time, the user uses CONTAPE's secondary capability, file-to-tape conversion, to read the file created previously with CONTAPE and convert its contents using the original conversion table.

```
-CONTAPE↵

:RUN↵
INPUT FROM CONDATA2↵   This is the file to be converted.
OUTPUT TO T↵           T indicates the terminal's punch.
NAME OF TABLE FILE: DATA2TAB↵   This is the same table used in the
                                original tape-to-file conversion.
TITLE:↵                The user does not wish a title to be punched on the tape.
TURN ON PUNCH.
THEN TYPE CARRIAGE RETURN.      The user turns on the punch, types a Carriage Return, and
↵                               the tape is punched and printed on the terminal, showing
AFECDBLRVMMXZBA                 the results of the conversion.
:QUIT↵                 The user gives the QUIT command to return to the EXECUTIVE.
-
```

APPENDIX A
HOW TO INTERPRET PUNCHED TAPE

The code that is punched on paper tape is a binary number. The binary number must be converted to an octal number. The octal number is the external code for that character on the tape.

To interpret the number, the user should look at the tape in this position:



A punch represents a 1; no punch represents a 0. The first number punched above is 10000101 binary.

This can be converted to octal by grouping in threes the digits from right to left. Each group becomes an octal digit. The octal digit is the sum of the values of the numbers punched in each group. For example,

10	000	101	
21	421	421	*value*
2	0	5	*octal digit*

The code punched is 205. In the Tymshare system, the eighth level is always punched; therefore, the external ASCII octal code is found by subtracting 200 from the octal code punched on the tape. In this example, the external code is 5. This corresponds to the character Control E.

The second number punched on the tape above is:

11	001	101	
21	421	421	*value*
3	1	5	*octal digit*

The external code is 115 (315 minus 200) for the character M.

APPENDIX B
TABLE OF CODE CONVERSIONS

Internal Decimal	Internal Octal	Character	ASCII[1,2] Keyboard	FLEX, N/C, EIA	External Octal	External Decimal	Binary[3]
0	0	blank		0	40	32	0100.000
1	1	!	1^s		41	33	0100.001
2	2	”	2^s		42	34	0100.010
3	3	#	3^s	t	43	35	0100.011
4	4	$	4^s		44	36	0100.100
5	5	%	5^s	v	45	37	0100.101
6	6	&	6^s	w	46	38	0100.110
7	7	’	7^s		47	39	0100.111
8	10	(	8^s		50	40	0101.000
9	11	)	9^s	z	51	41	0101.001
10	12	*	$:^s$		52	42	0101.010
11	13	+	$;^s$		53	43	0101.011
12	14	,			54	44	0101.100
13	15	–			55	45	0101.101
14	16	.			56	46	0101.110
15	17	/			57	47	0101.111
16	20	0			60	48	0110.000
17	21	1		/	61	49	0110.001
18	22	2		s	62	50	0110.010
19	23	3			63	51	0110.011
20	24	4		u	64	52	0110.100
21	25	5			65	53	0110.101
22	26	6			66	54	0110.110
23	27	7		x	67	55	0110.111
24	30	8		y	70	56	0111.000
25	31	9			71	57	0111.001
26	32	:			72	58	0111.010
27	33	;		,	73	59	0111.011
28	34	<	$,^s$		74	60	0111.100
29	35	=	$-^s$		75	61	0111.101
30	36	>	$.^s$	TAB	76	62	0111.110
31	37	?	$/^s$		77	63	0111.111

1 - Superscript s stands for a shift character.

2 - Standard Teletype keyboard configuration.

3 - This is the code actually punched on the tape. The decimal point represents the tape feed hole on the paper tape. A 1 represents a punched hole.

Internal Decimal	Internal Octal	Character	ASCII Keyboard	FLEX, N/C, EIA	External Octal	External Decimal	Binary
32	40	@	P^s	–	100	64	1000.000
33	41	A			101	65	1000.001
34	42	B			102	66	1000.010
35	43	C		l	103	67	1000.011
36	44	D			104	68	1000.100
37	45	E		n	105	69	1000.101
38	46	F		o	106	70	1000.110
39	47	G			107	71	1000.111
40	50	H			110	72	1001.000
41	51	I		r	111	73	1001.001
42	52	J			112	74	1001.010
43	53	K			113	75	1001.011
44	54	L			114	76	1001.100
45	55	M			115	77	1001.101
46	56	N			116	78	1001.110
47	57	O			117	79	1001.111
48	60	P			120	80	1010.000
49	61	Q		j	121	81	1010.001
50	62	R		k	122	82	1010.010
51	63	S			123	83	1010.011
52	64	T		m	124	84	1010.100
53	65	U			125	85	1010.101
54	66	V			126	86	1010.110
55	67	W		p	127	87	1010.111
56	70	X		q	130	88	1011.000
57	71	Y			131	89	1011.001
58	72	Z			132	90	1011.010
59	73	[	K^s		133	91	1011.011
60	74	\	L^s		134	92	1011.100
61	75	]	M^s		135	93	1011.101
62	76	↑	N^s		136	94	1011.110
63	77	←	O^s		137	95	1011.111

Internal Decimal	Internal Octal	Character	Special Internal	FLEX, N/C, EIA	External Octal	External Decimal	Binary
64	100	`			140	96	1100.000
65	101	a		a	141	97	1100.001
66	102	b	↴ only[1]	b	142	98	1100.010
67	103	c			143	99	1100.011
68	104	d		d	144	100	1100.100
69	105	e	↵ only[1]		145	101	1100.101
70	106	f			146	102	1100.110
71	107	g		g	147	103	1100.111
72	110	h		h	150	104	1101.000
73	111	i			151	105	1101.001
74	112	j			152	106	1101.010
75	113	k		.	153	107	1101.011
76	114	l			154	108	1101.100
77	115	m			155	109	1101.101
78	116	n			156	110	1101.110
79	117	o			157	111	1101.111
80	120	p		+	160	112	1110.000
81	121	q			161	113	1110.001
82	122	r			162	114	1110.010
83	123	s		c	163	115	1110.011
84	124	t			164	116	1110.100
85	125	u		e	165	117	1110.101
86	126	v		f	166	118	1110.110
87	127	w			167	119	1110.111
88	130	x			170	120	1111.000
89	131	y		i	171	121	1111.001
90	132	z			172	122	1111.010
91	133	{			173	123	1111.011
92	134	|	†		174	124	1111.100
93	135	}	††		175	125	1111.101
94	136	~			176	126	1111.110
95	137	RUBOUT		DEL	177	127	1111.111
				EOB	200	128	10000.000

†Line Printer Page.
††Multiple Space Compression or Alt Mode.

1 – This is true when the ONLC command is not used.

Internal Decimal	Internal Octal	Character[1]	ASCII Keyboard	FLEX, N/C, EIA	External Octal	External Decimal	Binary
96	140	P^{cs}	NULL		0	0	0000.000
97	141	A^{c}		1	1	1	0000.001
98	142	B^{c}		2	2	2	0000.010
99	143	C^{c}			3	3	0000.011
100	144	D^{c}		4	4	4	0000.100
101	145	E^{c}			5	5	0000.101
102	146	F^{c}			6	6	0000.110
103	147	G^{c}		7	7	7	0000.111
104	150	H^{c}		8	10	8	0001.000
105	151	I^{c}			11	9	0001.001
106	152	J^{c}	↴		12	10	0001.010
107	153	K^{c}		REW STOP	13	11	0001.011
108	154	L^{c}			14	12	0001.100
109	155	M^{c}	↵		15	13	0001.101
110	156	N^{c}			16	14	0001.110
111	157	O^{c}			17	15	0001.111
112	160	P^{c}		SP	20	16	0010.000
113	161	Q^{c}			21	17	0010.001
114	162	R^{c}			22	18	0010.010
115	163	S^{c}		3	23	19	0010.011
116	164	T^{c}			24	20	0010.100
117	165	U^{c}		5	25	21	0010.101
118	166	V^{c}		6	26	22	0010.110
119	167	W^{c}			27	23	0010.111
120	170	X^{c}			30	24	0011.000
121	171	Y^{c}		9	31	25	0011.001
122	172	Z^{c}			32	26	0011.010
123	173	K^{cs}	⊕		33	27	0011.011
124	174	L^{cs}			34	28	0011.100
125	175	M^{cs}			35	29	0011.101
126	176	N^{cs}			36	30	0011.110
127	177	O^{cs}			37	31	0011.111

1 - Superscript c stands for a control character; superscript cs stands for a control shift character.

USER EVALUATION

Tymshare would like to improve the quality and usefulness of its publications. However, to achieve this goal, we need your help and critical evaluations. Will you please provide us with such constructive information by filling out this questionnaire and mailing it back to us?

1. (a) Is this manual a useful document? ☐ Yes ☐ No

   (b) If your answer is Yes, what features make it useful.

   (c) If your answer is No, what features prevent it from being a useful document.

2. (a) Is the text clear and readily understandable? ☐ Yes ☐ No

   (b) If your answer is No, please cite the sections, subsections, or paragraphs that are unclear or difficult to understand.

3. (a) Are you pleased with the organization of this manual? ☐ Yes ☐ No

   (b) Should the organization be changed? ☐ Yes ☐ No

   (c) What changes do you suggest?

4. (a) Are the example problems helpful and easy to understand? ☐ Yes ☐ No

   (b) Should more examples be added when this manual is revised? ☐ Yes ☐ No

   (c) What kind of programs would you like to see added?

5. (a) Should anything be deleted from this manual when it is revised? ☐ Yes ☐ No

   (b) If your answer is Yes, give us your suggestions.

PAPER TAPE PACKAGE

6. List any further suggestions you have for the improvement of this manual.

OPTIONAL INFORMATION:

Name _____ Company _____

Street Address _____

City _____ State _____ Zip _____

Occupation _____

THANK YOU FOR YOUR ASSISTANCE.

To mail this questionnaire, simply cut from manual, fold Part A back, fold Part B back, and staple where indicated. **Part A**

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

First Class
Permit No.
300
Los Altos,
California

POSTAGE WILL BE PAID BY

TYMSHARE, INC.
525 UNIVERSITY AVENUE, SUITE 220
PALO ALTO, CALIFORNIA 94301

ATTN: Documentation Group

Part B

STAPLE HERE