

 *OVERLAND DATA*

Installation & Tape Utilities

User Manual



Installation and User's Manual

PN 10401Q

© 1983, 1985, 1986, 1987, 1988, 1990 & 1991
by Overland Data, Inc.
ALL RIGHTS RESERVED

Table of Contents

If You Don't Read This Manual	i
Where to Look for Information.....	iii
Overview.....	iii
Installation and Testing.....	iii
Subsystem Test Tools.....	iii
Easy Operation	iii
Data Interchange.....	iv
Hard Disk Backup	iv
Tape Programming Utilities.....	iv
UNIX/XENIX 286 or 386	iv
Custom Configuration.....	iv
Warranty Information.....	iv
About 9-Track Tape.....	1
Overland Data 9-Track Products.....	1
Flashbak: High Speed Backup & Restore	1
Novell Tar: 286 Novell Network Backup.....	1
MS-DOS 9-Track Programming Toolkit	1
UNIX and XENIX Support	2
TapeView: Enhanced Data Interchange	2
Details About 9-Track Tape.....	3
Physical Characteristics of 9-Track Tape	3
Tape Format and Density	3
Data Bytes.....	4
Tape Capacity.....	4
Tape Layout.....	5
Unlabeled Tapes	5
Labeled Tapes.....	5
Tape Data	7
Data Blocks.....	7
Choosing an Appropriate Block Size.....	7
Records.....	8
Data Fields.....	8
Tape Formats.....	8
Format Specifications.....	9
Signed Integers	9
Unsigned Integers	9
Microsoft Floating Point.....	9
IEEE Floating Point.....	10
Packed BCD.....	11
Unpacked BCD	11
Comp3.....	11
Zoned Decimal.....	12
Installation and Configuration	13
Controller Model TX-8	14

Standard Features.....	14
Inventory Checklist.....	14
Related Information.....	14
System Requirements.....	14
Hardware Installation.....	15
Remove the System Cover.....	15
Select an Open Card Slot.....	15
Check the DIP Switch Settings.....	16
Install the TX-8.....	17
Install the Cable.....	17
Reinstall the System Cover.....	17
Connect the Cable to the Tape Drive.....	18
Grounding.....	19
Installing the Software.....	19
TX-8 Configuration.....	19
The CONFIG Program.....	20
Base Port.....	20
Drive Type.....	21
J2-50 Speed/Density Select Line.....	21
Interrupt Level.....	22
Tape Drive Address.....	22
Test the Complete System.....	23
Controller Model TXi-16.....	24
Standard Features.....	24
Inventory Checklist.....	24
Related Reading.....	24
Hardware Installation.....	25
Remove the System Cover.....	25
Select an Open Card Slot.....	25
Install the TXi-16.....	26
Install the Cable.....	26
Reinstall the System Cover.....	26
Connect the Cable to the Tape Drive.....	27
Grounding.....	27
Installing the Software.....	28
TXi-16 Configuration.....	28
Configuration Procedure.....	28
TXi-16 Configuration.....	29
Setting Tape Drive Type and the IO Address.....	30
Test the Complete System.....	32
TXISPD.....	33
SETCARD.....	34
Controller Model XL/2.....	35
Standard Features.....	35
Inventory Checklist.....	35
Related Reading.....	35
System Requirements.....	35
Hardware Installation.....	36
Remove the System Cover.....	36

Install the XL/2.....	36
Reinstall the System Cover.....	36
Connect the Cable to the XL/2.....	36
Connect the Cable to the Tape Drive.....	36
Grounding.....	37
XL/2 Configuration Overview.....	37
Address.....	38
Interrupt Level.....	38
Drive Address.....	38
Tape Density/Tape Speed Select.....	38
Streaming or Start/Stop.....	38
Configuring the PS/2 for the XL/2.....	39
Start Reference Diskette.....	39
Copy XL/2 Configuration File to Reference Diskette.....	39
Activate XL/2 Configuration File.....	39
Installing the Software.....	40
Test the Complete System.....	40
9-Track Tape Subsystem Tools.....	41
XTEST.....	41
Features.....	41
Use.....	41
WHAT.....	43
Use.....	43
Limitations.....	43
Tape Applications.....	45
DEPOT4.....	46
Setup.....	46
Using DEPOT4.....	47
Starting DEPOT4.....	47
The DEPOT4 Screen.....	47
Menus.....	47
Dialog Boxes.....	48
Help.....	48
The Main Menu.....	48
Programs and Data Windows.....	49
DEPOT.....	52
Features.....	52
Storing Data on Tape.....	52
Recovering Data From Tape.....	53
Use.....	53
Main Menu.....	54
Translation.....	54
Writing a New Tape (Option 1).....	54
Appending to an Existing Tape (Option 2).....	55
Reading a Tape (Option 3).....	56
Multiple Volume Tape Writes.....	58
Error Handling.....	58

Limitations.....	59
DEPOT2.....	60
Features.....	60
Reading from Tape.....	60
Record Size and Blocking Factor.....	61
Use.....	61
Writing to Tape.....	62
Record Size and Blocking Factor.....	63
Multiple File Operations.....	64
Options.....	65
Exit Codes.....	65
Multiple Volume Tape Writes.....	66
Error Handling.....	66
Command Line Examples.....	67
Example Command File.....	68
DOS Batch File Examples.....	69
Limitations.....	70
Bugs.....	70
DEPOT3.....	71
Use.....	71
Creating Script Files.....	71
Example Scripts.....	72
Command Reference.....	80
Error Handling.....	82
FDUMP.....	84
Features.....	84
Use.....	84
Examples.....	89
Limitations.....	91
LABEL.....	92
Use.....	92
IBM Standard Labeled Tape.....	93
Write Tape.....	98
End of File Processing.....	98
ANSI Standard Labeled Tape.....	98
TAPEUTL.....	101
TAPEUTL.....	101
Processing Unlabeled Tapes.....	102
Copy from Disk to Tape.....	103
Fixed-Length Data Transfer.....	105
Variable-Length Data Transfer.....	106
String Data Transfer.....	108
Copy from Tape to Disk.....	109
Variable-Length Data Transfer.....	112
IBM Variable-Length Record Transfer.....	113
IBM VARIABLE-LENGTH RECORD LAYOUTS.....	113
String Data Transfer.....	114
Reposition Tape or Display Records.....	114
Copy Multiple Files from Tape to Disk.....	115

Summarize Tape Data	115
Processing Labeled Tapes.....	115
About Tape Positioning During Labeled Tape Operations	116
Copy from Disk to Tape.....	117
Fixed-Length Data Transfer	119
Variable-Length Data Transfer.....	121
String Data Transfer.....	123
Copy from Tape to Disk.....	124
Fixed-Length Data Transfer	127
Variable-Length Data Transfer.....	129
IBM Variable-Length Record Transfer.....	130
String Data Transfer.....	131
List Files on the Tape.....	132
Initialize Volume Label.....	133
Positioning the Tape and Displaying the Records.....	134
How the Repositioning Operations Work.....	135
Spacing Forward/Backward by Files	135
Spacing Forward/Backward by Blocks.....	136
Space to Logical End of Tape.....	137
Rewind Tape.....	138
Rewind and Unload Tape.....	138
Write Tapemark	138
Displaying Tape Records.....	138
CHANGING THE DISPLAY BETWEEN ASCII & EBCDIC	140
TAPE RECORD DISPLAY SCREENS	140
Changing the Tape System Parameters.....	141
Set low tape speed (default).....	142
Set high tape speed	142
Set Short Interrecord gap for write	142
Set long Interrecord gap for write	142
Set error retry count.....	143
Set tape address.....	143
Set 1600 bits/inch density.....	143
Set 3200 bits/inch density.....	143
Set maximum buffer size for read	143
The DOS Filter Function.....	144
Initializing the DOS Filter.....	144
How and When the Filter Functions.....	144
Filter Translation Statements	145
FILTER TRANSLATION STATEMENT TYPES	147
Special Note on ASCII vs. EBCDIC Character Strings.....	148
Translation Statement Examples.....	148
Special Note on the Order of Translations.....	149
Automatic Operations of TAPEUTL.....	150
The Automatic Mode Status Marker.....	150
Recording Your TAPEUTL Session	150
Providing for User Input in Automatic Mode.....	151
Ending Automatic Execution Recording.....	151
Running TAPEUTL from a Command File.....	151

ASCII Batch Command File.....	152
Tape Data Extraction.....	154
How Programs in This Package Work Together.....	154
System Requirements.....	155
Program Descriptions.....	156
EXTRACTT.EXE.....	156
RECONT.EXE.....	156
TDE.BAT.....	156
RECFMT.EXE.....	156
INSUTIL.EXE.....	156
EXTRACTT.EXE Program.....	157
General Features.....	157
Screen Description—INITIAL.....	158
Screen Entries.....	158
Labeled Tape Notes:.....	159
Existing Control Statement Files:.....	159
Screen Description—MAIN.....	159
Screen Divisions.....	159
Upper Screen.....	159
Field Marking.....	160
Control Statement File Sequence:.....	161
Screen Description—FILL.....	161
Screen Entries.....	161
Screen Description—INSERT.....	162
Screen Entries.....	162
Screen Description—MOVE.....	163
Input Location Start.....	163
Screen Description—SELECT.....	163
Select/Reject.....	164
Compare.....	164
Field.....	164
Compare Data.....	164
Beginning Position in Input Record.....	165
Numeric Field Length.....	165
Numeric Field Translation.....	165
Screen Description—NUMERIC.....	165
Input Type of Field.....	165
Input Beginning Position.....	165
Input Field Length.....	165
Input Translation.....	165
Output Type of Field.....	166
Output Beginning Position.....	166
Output Field Length.....	166
Output Translation.....	166
Name.....	167
Screen Description—FINAL.....	167
Output Data File.....	167
Record Type.....	167
Record Length.....	167

Skip.....	167
Process.....	167
Speed.....	167
Address.....	167
ASCII to EBCDIC Table.....	168
EBCDIC to ASCII Table.....	168
Sample.....	168
Screen Description—HELP.....	168
RECONT.EXE Program.....	169
Error Statements.....	170
Error Messages Listed by Number.....	170
Input Field Errors.....	170
Output Field Errors.....	170
Field Conversion Errors.....	171
RECONT.EXE.....	171
TDE.BAT.....	171
RECFMT.EXE Program.....	171
DOS Device Names for Use With RECFMT:.....	172
INSUTIL.EXE.....	173
Tape Backup.....	175
TAR.....	175
Features.....	175
Options.....	175
Warnings.....	178
Examples.....	178
Archive Format.....	180
Limitations.....	180

If You Don't Read This Manual

Not reading the manual can present hazards to your equipment and to the data you are putting on tape.

At a minimum:

- Review the introduction
- Skim over the sections that affect your application to make certain that you take any precautions needed.

Where to Look for Information

(When All Else Fails)

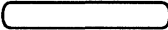
Many people don't read manuals; at least, not all at once. When you are looking for specific information, check the table of contents. It is likely this will lead you to that information.

Note: We try to keep our user documentation up to date. However, last-minute changes may supercede this manual. Read any addenda included with the manual, and read the `readme.doc` file on the release disk before using the utilities.

Overview

Conventions

Throughout this manual, we use the following conventions:

- input** Something you must type from the keyboard is shown bold in this constant width typeface.
- ENTER** This typeface that shows the name of the key on your keyboard in an oval, such as **ENTER** or **CTRL**.
  indicates the spacebar.
- CTRL)+E** When you need to press two keys together, we show a both keys, joined by a plus.
- prompt The program's prompts, and information displayed on-screen is shown using this typeface.

Installation and Testing

- For information on installing, configuring and testing:
 - TX-8, see page 14.
 - TXi-16, see page 24.
 - XL/2, see page 35.

Subsystem Test Tools

- **xTEST** tests the integrated TX-8, XL/2 or TXi-16 tape subsystem; see page 41.
- **WHAT** shows each source module's version in the tape programs; see page 43.

Easy Operation

- For menu driven access to the tape utility programs, see **DEPOT4** on page 46 .

Data Interchange

- For simple, menu driven applications; **DEPOT**, page 52.
- For a "batch" file application; **DEPOT2**, page 60.
- For record-level data transformations; **DEPOT3**, page 71.
- To examine a "mystery tape," see **FDUMP** on page 84.
- To write IBM or ANSI labeled tapes; **LABEL**, page 92, or **TAPEUTL**, page 101.
- For visually oriented tape interchange; **TAPEVIEW** (optional) in the (separate) **TAPEVIEW** manual.

Hard Disk Backup

- To backup a hard disk to tape, see **TAR**, page 175.
- For enhanced backup from hard disk to tape, see **FLASHBAK** (optional), in the (separate) Tape Options manual.
- For network backup (network backup is only available with the enhanced software of the TXi-16, TX-8 and the XL/2 boards or as an option), see **NOVELL TAR**, in the (separate) Tape Options manual.

Tape Programming Utilities

All of the Tape Programming Utilities are included in the (separate) Tape Options manual.

- To write custom tape programs in TurboPascal, Microsoft Quick Basic, IBM Professional Fortran, Microsoft Fortran and Microsoft C see the **MS-DOS DEVICE DRIVER**.
- To write "low level" custom tape programs in Microsoft C see **ALLTAPE**.
- **TCOPY** provides the ability to copy from tape to tape (needs two drives, two controllers and two DOS device drivers).
- **TDISK** lets a tape drive function somewhat like a disk drive.

UNIX/XENIX 286 or 386

- Refer to the (separate & optional) UNIX Tape Package manual.

Custom Configuration

- For Controller Model TX-8, see Appendix A, page A-5
- For Controller Model XL/2, see Appendix A, page A-8
- To reconfigure the software, see Config in Chapter 2.

Warranty Information

- For warranty information, see Appendix B, page B-1.

The following notice applies to all Overland Data tape drive controller cards; the TX-8, TXi-16, and XL/2. All have been tested and meet the standards for Class A computing equipment.

WARNING—This equipment generates, uses and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of the FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

About 9-Track Tape

Nine-track tape has a special status for mainframes and mini-computers. The drive interface and recording technique are standard, so data interchange with 9-track tape is easy, and an alternative to direct links between PCs, mainframes and mini-computers. To move data from “large” computers to PCs, write a 9-track tape and then read it on your personal computer. Moving data to a mainframe or mini-computer is just as easy.

Overland Data 9-Track Products

We offer subsystems (software, controller card and 9-track tape drive) that add 9-track tape to your PC. We offer the TX-8 for PC XT, AT, and 286/386-class machines; the TXi-16 for PC AT 286/386/486 and compatible Compaq EISA 486 (16 bit slot) machines; and the XL/2 for PS/2 Micro Channel Architecture machines. Chapter 2 describes installation and configuration.

Flashbak: High Speed Backup & Restore

Flashbak is an interactive, window-oriented tape backup and restore utility, allowing you to: examine your disk directory structure and file names, select files that match a set of file names for backup or restore, tag modified files for backup, or select only the files in a subtree. Flashbak does not support network backup—for network backup, see Novell Tar, below.

Novell Tar: 286 Novell Network Backup

Novell Tar is an upgraded version of the Tar utility included in this package which allows you to backup and restore 286 Novell networks (it does not support 386 Novell). The DEPOT4 program included in this package will access and work with Novell Tar, providing the same user friendly interface.

MS-DOS 9-Track Programming Toolkit

This is a tool set giving access to the the programming interface for your tape controller. These include an installable “character” device driver allowing you to use DOS calls to read, write and position the tape drive; an object module linked to your application; and a “block” device driver providing limited access to tape drives for programs which are not “tape-aware.”

UNIX and XENIX Support

We offer an installable device driver for your SCO XENIX 286 or 386, Microport System V/AT, Microport System V/386 or Interactive UNIX 386 system. This lets you use the standard utilities, dd, tar, and cpio for interchange and backup. See the UNIX/XENIX Reference Manual for information. The TX-8 and TXi-16 support UNIX, while XENIX operates only on the TX-8.

TapeView: Enhanced Data Interchange

We offer a powerful tape interchange utility, TapeView—a window-oriented program that lets you visually define records and fields and specify field-level translations. TapeView writes QuickBASIC programs to perform requested data transfer and translations. QuickBASIC-compatible libraries allow you to write custom programs or modify TapeView-produced programs.

Details About 9-Track Tape

Since 9-track drives use a standard format, software relies on consistent tape format and physical attributes.

Physical Characteristics of 9-Track Tape

Tape reel diameters are usually 10½, 8½ or 7 inches. A 10½ inch reel holds 2400 or 3600 feet of tape, depending on tape thickness. The 2400-foot tape is most common. A 7 inch reel holds 600 feet. Some tape drives can only use the smaller reel size (lengths are for new tapes; used tapes may be shorter for several reasons).

Each reel has two reflective strips on the back of the tape. The Beginning of Tape (BOT) marker is about 15 feet from the start of the tape. Tape drives “see” this marker and place the head just past it when loading or rewinding. The End of Tape (EOT) marker is about 15 feet from the tape’s end. The EOT marker warns software that the tape end is near. Drives can record past this, but must stop fast to keep the tape from coming off the reel.

Occasionally BOT or EOT markers are damaged. Note the position of the marker on the tape carefully before replacing it. Data is recorded in “blocks” separated by a small amount of blank tape called an inter-record gap (IRG). A file is one or more blocks separated from the next file by a special block called a filemark (or filemark). Two filemarks signal the end of data (called logical EOT)—usually before physical EOT. No data should be recorded past logical EOT. To avoid tape coming off the supply reel, logical EOT should appear soon after the EOT marker.

9-track drives have a “safety,” to prevent writing over data on tape. Usually, a tape is “write-protected” if it lacks a write ring in the back of the reel (the ring allows writing). A reel without the ring keeps the drive from writing. Some drives ignore the write ring and use a file protect switch on the drive front panel instead.

Tape Format and Density

Density is measured in bytes-per-inch (bpi), the bytes of data on an inch of tape. This is the data density, ignoring IRGs. The format describes the recording technique used by the drive. There are three standard formats:

- NRZI (non-return to zero inverted):
The oldest and least common format; density is 800 bpi.
 - PE (phase-encoded):
Recording density is 1 600 bpi or 3 200 bpi. 1 600 bpi is most common for data interchange. 3 200 bpi is used for backup (because twice as much data fits on a tape reel) and is not as commonly used for interchange.
 - GCR (group-coded recording):
GCR records at 6 250 bpi. GCR is a common interchange format, especially between mainframes. With almost four times 1 600 bpi's data capacity, it is a good backup choice.
- Many drives support several densities. These are described as single-, double-, tri- or quad-density.

Data Bytes

9-track drives write nine parallel data tracks for each data byte; one for each of the eight bits and one for parity. The parity track is used to detect read and write errors (and to correct read errors).

Tape Capacity

Tape capacity depends on tape length, block size, IRG size, and density. The formula for computing tape capacity is as follows:

$$\text{TAPECAPACITY} = \frac{\text{TAPELENGTH}}{\frac{\text{BLOCKSIZE}}{\text{DENSITY}} + \text{IRG}} \times \text{BLOCKSIZE}$$

TAPECAPACITY is tape capacity measured in bytes
 TAPELENGTH is the tape length measured in inches
 BLOCKSIZE is the block size measured in bytes
 DENSITY is the recording density measured in bytes per inch
 IRG is the length of the inter-record gap in inches.

This table lists tape capacity for various block sizes for 1 600 and 3 200 bpi, assuming a 0.6 inch IRG and a 2 400-foot reel of tape.

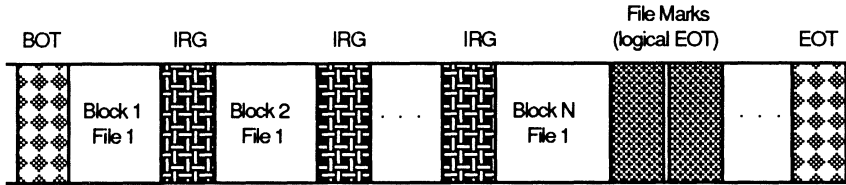
Density (bpi)	Block Size (bytes)					
	512	1024	3072	5120	10240	16384
1600	16.0	23.7	35.1	38.8	42.1	43.5
3200	19.4	32.0	56.7	67.0	77.6	82.4

Tape Layout

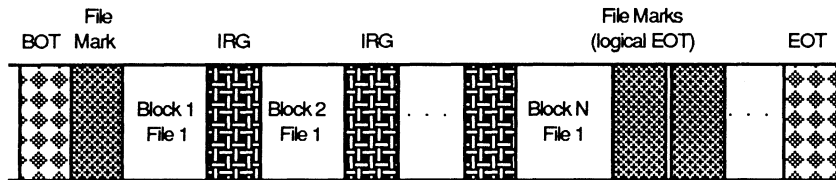
The combination of filemarks and data blocks is the tape layout.

Unlabeled Tapes

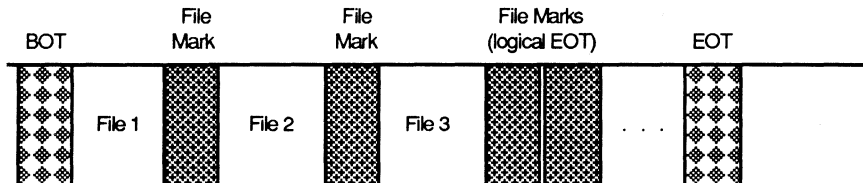
The generic layout is “unlabeled tape.” For example, a tape with a single file. The file contains blocks separated by IRGs and ended by two filemarks. This diagram shows such a file, of N blocks.



Some tapes are written with an initial file mark, as follows:

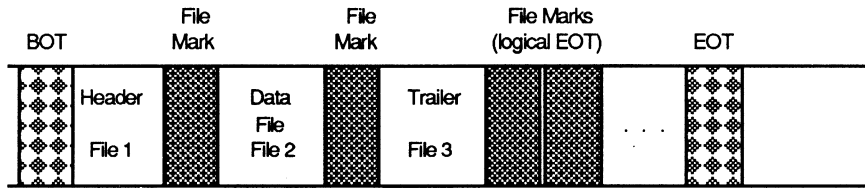


Tapes with two or more files use the same format (one filemark between files; two filemarks after the last file showing the logical EOT). The blocks are separated by IRGs. This diagram shows the layout for three files (we just show files and filemarks).



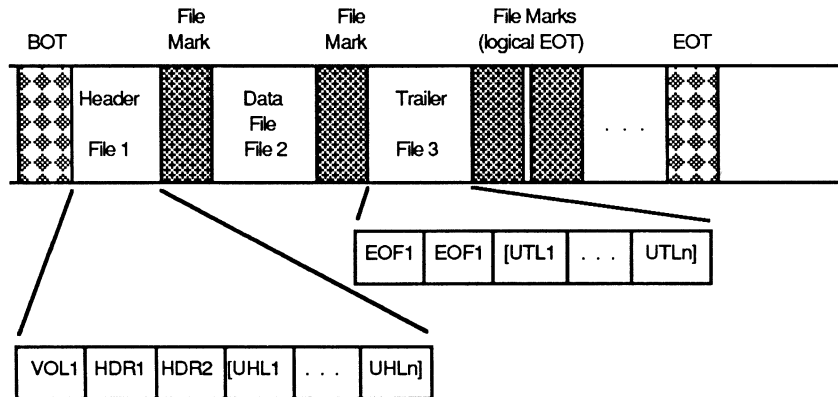
Labeled Tapes

In general, labeled tapes have at least three files—a header file describing the data, a data file and a trailer file. Other data files may follow with the same format: header, data and trailer. This diagram shows a labeled tape with a single data file.



There are two standards, IBM and ANSI. IBM specifies labels (header and trailer) hold only EBCDIC characters. ANSI specifies labels use ASCII characters only. Labels are 80-byte blocks and both standards define similar information in the headers and trailers. For the IBM standard, refer to IBM Publication OS/VS Tape Labels , GC26-3795-3, available from the IBM branch office for your area. For the ANSI standard, refer to American National Standard Magnetic Tape Labels and File Structure for Information Interchange X3.27-1978. Copies may be ordered from: American National Standards Institute, 1430 Broadway, New York, NY 10018

For both, the label header is a header file of at least three 80-byte blocks. Further blocks are optional user-defined labels.



These blocks are: volume label (VOL1), header 1 label (HDR1) and header 2 label (HDR2). Optional user labels (UHL1—UHL8) follow (not supported by **LABEL**). The file ends with a filemark. The header and trailer labels use identical formats, except that there is no VOL1 trailer. End of file trailer 1 (EOF1) follows the data file and filemark. It has the same format as the HDR1 label. The end of file trailer 2 (EOF2) follows EOF1. Any user trailers, called UTL1 through UTL8, follow EOF2. Two filemarks end the last trailer.

Tape Data

Here is an analogy between tape data and a filing cabinet. Each drawer is a reel of tape. It contains one or more files. A file in the drawer is marked with a tab. Such tabs are similar to the tape's filemarks. The data between the tabs may be divided into manila folders, each of which is like a block of data on tape. Each folder can have many pieces of paper in it, and this is similar to having several logical records in a block on tape.

Data Blocks

The unit of data on tape is a data block. Sometimes, people use "record" when referring to a block. In this document, we use the term block for the data on tape, surrounded by the IRGs. We use the term record for the logical grouping of the data. In many cases the record size is the same as the block size; that is, a block contains a single record. More often, blocks contain multiple records (20, 100, 500, etc.). The number of records per block is the blocking factor and the block size is an even multiple of the record size. Typically, a program writes data to tape in a series of fixed-sized blocks, ending the data transfer with two filemarks.

The IBM standard for variable-sized blocks requires a 4-byte header (in zoned decimal), showing the block size, before the data. The block length includes the 4-byte header. If the data length is 2,000 bytes, the block length header contains the value "2004".

Choosing an Appropriate Block Size

There is not a "correct" block size, but a range of sizes appropriate for an application. If the data is in fixed-size records, the block size will be a multiple of the record size. If we have a database of 10,000 records and each record is 230 bytes long, we can pick a block size that is an even multiple of 230 (such as 230, 4600, etc.). Remember: block size affects tape capacity and tape drive data rate (the larger the block size, the higher the tape capacity and the higher the tape drive data rate). Here, 4600 is a "medium-sized" block, offering acceptable tape capacity and tape drive data rate and high compatibility within an unknown configuration.

Records

A record is a logical data grouping. There are fixed length and variable length records. Fixed length records are convenient for writing to tape (accommodating blocking characteristics). Often, data is fixed-size because it is stored that way in a database. If the data is not fixed-size (e.g., a text file), we can place each line of text in a record (say, 100 bytes) and “blank-pad” the record to size.

IBM-format, variable-size records use a 4-byte header (in zoned decimal) before the data, showing the record length (including header). This is used with the variable-size blocks. Such a block has a 4-byte block size indicator, a 4-byte record size indicator, and the first record’s data. The next record appears with a 4-byte record size indicator followed by that record’s data.

Data Fields

We can break up a data record into individual fields. For fixed-size records, each record has the same number and sizes of fields. Different fields, however, have various sizes and formats. If the entire record is character data, each field will consist of ASCII or EBCDIC values. It’s possible for the fields to be a number format, such as Binary Coded Decimal (BCD), binary 2-byte integer, binary 4-byte integer, or one of several floating point formats. The next section discusses standard formats found in 9-track tape data.

Tape Formats

This section shows you the standard formats that you’re likely to run across while reading and writing 9-track tapes.

Format	Size (bytes)	Format	Size (bytes)
Integer	1, 2 or 4	Packed BCD	open
Unsigned Integer	1, 2 or 4	Unpacked BCD	open
Microsoft Floating Point	4 or 8	Comp3	open
IEEE Floating Point	4 or 8		

Format Specifications

Signed Integers

Signed integers all use the leftmost bit as a sign bit (1 for negative, 0 for positive).

- One Byte Signed
One byte integer values range from -128 (0x80) to +127 (0x7F). They are 8 bits wide.
- Two Byte Signed
Two byte integer values range from -32,768 (0x8000) to +32,767 (0x7FFF). The format is 16 bits wide.
- Four Byte Signed
Four byte integers range from -2,147,483,648 (0x80000000) to +2,147,483,647 (0x7FFFFFFF). They are 32 bits wide.

Unsigned Integers

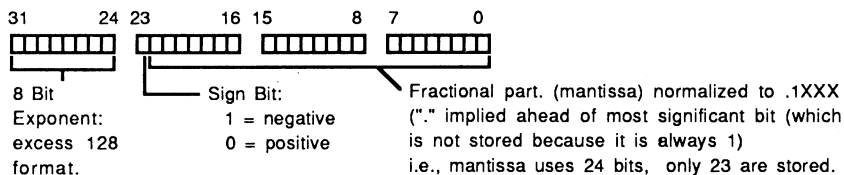
Unsigned integers do not use a sign bit.

- One Byte Unsigned
One byte unsigned integer values range from 0 (0x00) to 255 (0xFF). The format is 8 bits wide.
- Two Byte Unsigned
Two byte unsigned integer values range from 0 (0x0000) to 65,535 (0xFFFF). The format is 16 bits wide.
- Four Byte Unsigned
Four byte unsigned integers range from 0 (0x00000000) to 4,294,967,295 (0xFFFFFFFF). The format is 32 bits wide.

Microsoft Floating Point

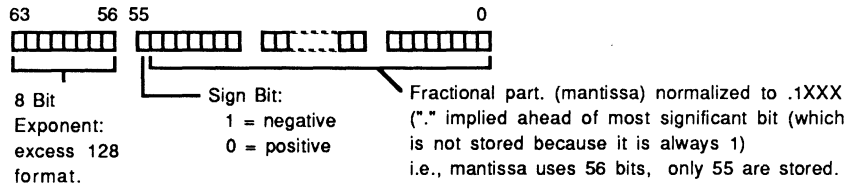
Microsoft floating point numbers' range is 1.7E+38 to 2.9E-39.

- Four Byte Floating Point
The bias of the exponent is 128 and the mantissa is normalized to between .1 and .2. A zero is represented by an exponent of all zeros (the other bytes don't matter). The format is:



- Eight Byte Floating Point

These have the same range as the 4-byte numbers, but 8-byte format's accuracy is 15 decimal places. The format is:

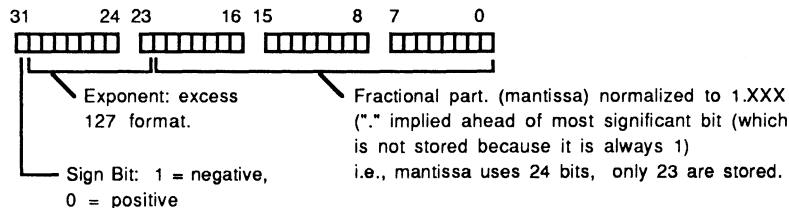


Note that the only difference between the 32 bit and the 64 bit format is the number of bits allocated to the mantissa.

IEEE Floating Point

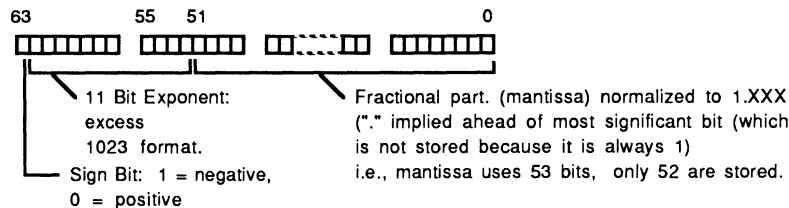
- Four Byte Floating Point

Four byte IEEE floating point numbers range from about $3.4E-38$ to $3.4E+38$. The format of this 32 bit number has three fields: 1 sign bit, an 8 bit biased exponent, and a 23 bit mantissa. The bias of the exponent is 127 and the mantissa is normalized to between 1 and less than 2. The format is:



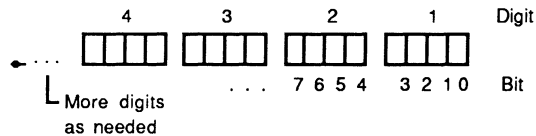
- Eight Byte Floating Point

Eight byte IEEE floating point numbers range from about $1.7E-307$ to $1.7E+308$. The format of the 64 bit number has three fields: 1 sign bit, an 11 bit biased exponent, and a 52 bit mantissa. The bias of the exponent is 1023. It looks like:



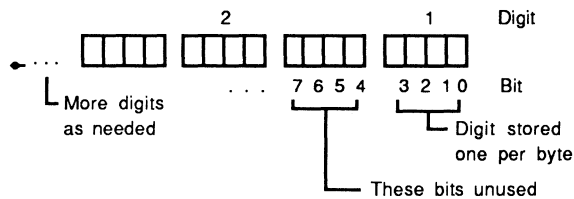
Packed BCD

For packed BCD (Binary Coded Decimal) each digit is stored in a 4 bit nibble (half a byte). Binary numbers 0000 to 1001 represent the digits 0 to 9. Digits are “packed,” 2 per byte, for a number of any length. BCD usually represents an even number of digits, but the most significant digit can be discarded. BCD does not have a decimal point, but can imply one. The format is:



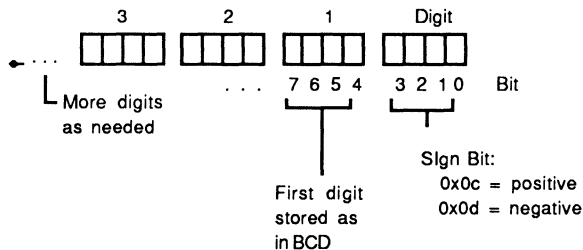
Unpacked BCD

BCD may also be unpacked, with only one digit per byte. This is not as commonly as packed BCD (it wastes storage). Number representations are the same as packed BCD, but the high four bits in each byte are not used. The format is:



Comp3

Comp3 (also known as packed decimal), has the same format as packed BCD, except that Comp3 uses the least significant nibble to represent a sign bit. The format of a Comp3 number is:



Zoned Decimal

Zoned decimal stores values for each digit in EBCDIC, one digit per byte. The least significant digit is modified on negative numbers by mapping the EBCDIC codes as shown below.

Digit	EBCDIC	Becomes
0	0xF0	0xD0
1	0xF1	0xD1
2	0xF2	0xD2
3	0xF3	0xD3
4	0xF4	0xD4
5	0xF5	0xD5
6	0xF6	0xD6
7	0xF7	0xD7
8	0xF8	0xD8
9	0xF9	0xD9

For example, -25 would be coded as 0xF2D5.

Installation and Configuration

This chapter describes installation and configuration procedures for Overland Data's 9-track controllers: TX-8, XL/2 and TXi-16. All models use programmed I/O instead of DMA for data transfer. The TX-8 and XL/2 both have 64kB on-board cache buffers, the TXi-16 has a 1MB cache buffer. The TXi-16 improves performance by using 16-bit I/O. The DOS software uses loadable drivers (set during installation) making it device independent.

The following notice applies to all Overland Data tape drive controller cards; the TX-8, TXi-16 and XL/2 . All have been tested and meet the standards set for Class A computing equipment.

WARNING — This equipment generates, uses and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of the FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Controller Model TX-8

This describes the TX-8 and its installation procedures. Read it and make sure you have everything in the checklist, the PC system documentation, that your card is correctly configured, and that you understand the procedures before installation.

Standard Features

The standard TX-8 features are:

- 64kB cache buffer.
- Switch selectable I/O address (8 contiguous ports required).
- Modes: GCR, PE and NRZI.
- Densities: 800, 1600, 3200, and 6250 bytes per inch (bpi).
- Compatible with most Cipher/Pertec formatted tape drives.
- Burst transfer rate: 900kB/s. Sustained rate: 632kB/s. System throughput depends on the PC, program, tape condition and hard disk attributes).

Inventory Checklist

Check that your TX-8 package includes the following:

- TX-8 circuit board and bracket.
- One round, shielded cable.
- One "P"-clip for cable grounding at the drive.
- 5 1/4 inch or 3 1/2 inch disks (the number of disks varies) .

Related Information

Some familiarity with DOS and computer hardware is assumed. Operating system commands and I/O port designations assume a single-user, single tasking version of MS-DOS.

System Requirements

Any IBM PC, XT, AT or fully compatible machine (i.e., COMPAQ portable and 386, and the Tandy PC line) under MS-DOS.

The supplied software requires a minimum of 512kB of RAM.

No DMA channel is required for operation.

Be sure the MS-DOS system option `VERIFY` mode is off when using the DOS device driver.

The DOS Device Driver (Options Manual) is NOT required to use the DOS programs provided with the controller. Installed, the driver uses memory; do not install it unless you plan to access the tape system using your own software.

Hardware Installation

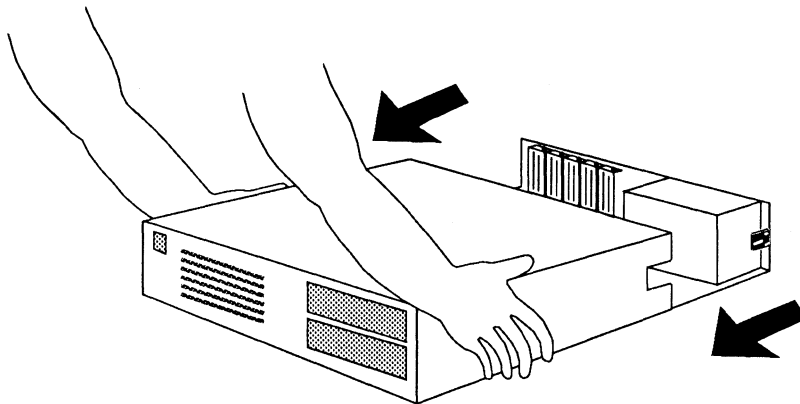
Remove the System Cover

Turn off and unplug the computer and all peripherals.

Move your keyboard and all external options away from the work area. Place the PC in a position so you can work at its back.

Remove all cover mounting screws (counterclockwise). Be sure to save all screws for the cover replacement.

Move to the front of the PC. Slide the cover to the front. When the cover will go no further, tilt it up, remove it, and set it aside.



Removing the System Cover

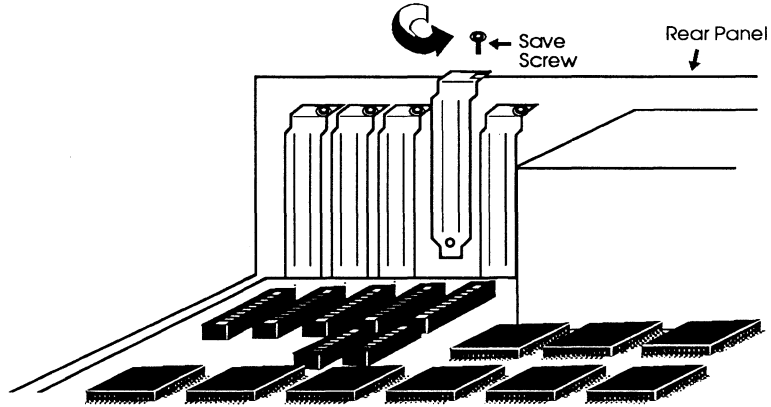
Select an Open Card Slot

Look at your computer's inside left rear. The PC has five slots and the XT and AT have eight. Do not install the adapter in the short slots in the XT. In an AT, use a 16-bit slot if possible.

If you are installing the TX-8 in an 8 bit slot (PC, XT or AT) you must configure the controller as follows:

- If you plan to use hardware interrupts, you must physically change the IRQ jumper from 10 to 3, 4, 5, 6 or 7.
- The 16-bit (small) connector does not have to be in a motherboard connector for the TX-8 to work.
- If you install the TX-8 in an 8-bit slot, it is important that the 16-bit connector does not touch (and short out on) anything!
- The extra connector is for extended AT interrupts only.

Remove the screw that holds the expansion slot cover in place. Turn counterclockwise to loosen the screw. Be sure you save the screw; it will be used to install the product later.

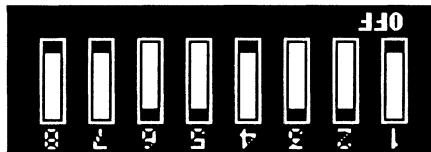


Removing the empty slot cover

Check the DIP Switch Settings

Hold the TX-8 with the component side facing you, gold card-edge connector down. The DIP switch is the upper right corner. Check that the settings of the eight switches are as shown.

Switch	Setting
1	ON
2	OFF
3	OFF
4	ON
5	OFF
6	OFF
7	ON
8	ON



The appearance of the red dots on the rocker switches indicates that that side of the switch is inactivated. In the example above, switches 1,4,7 and 8 are ON, so the red dots are UP on the OFF side of the switches.

If you know of an I/O address conflict between the TX-8 and another adapter in your system, see the table below to select a new I/O base address, and reset the DIP switches before continuing with the installation. You can change the I/O base port on a TX-8 controller without running `CONFIG` if the new base port you select appears in the table following.

DIP Switch	1	2	3	4	5	6	7	8
Address Line	A10	A9	A8	A7	A6	A5	A4	A3
360 (default)	ON	OFF	OFF	ON	OFF	OFF	ON	ON
300	ON	OFF	OFF	ON	ON	ON	ON	ON
310	ON	OFF	OFF	ON	ON	ON	OFF	ON
320	ON	OFF	OFF	ON	ON	OFF	ON	ON
330	ON	OFF	OFF	ON	ON	OFF	OFF	ON
340	ON	OFF	OFF	ON	OFF	ON	ON	ON
350	ON	OFF	OFF	ON	OFF	ON	OFF	ON

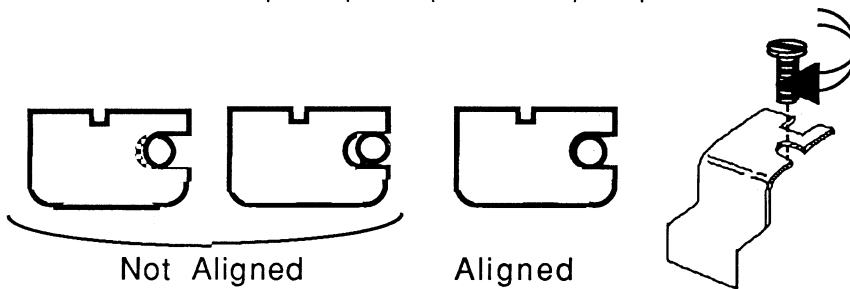
Set the DIP switch, and I/O base port is reset. So long as the address selected is in this table, no reconfiguration is required. If the address is not in this range, you must run **CONFIG** on each utility you plan to use.

Unlisted base port addresses require you to set the correct DIP switches and run **CONFIG**.

Note the new address for reference during configuration.

Install the TX-8

Press the TX-8 into the expansion slot. Put the non-connector end into the card guide. Align the slot in the bracket with the hole in the rear computer panel (see below). Replace the screw.



Aligning the TX-8 retaining bracket

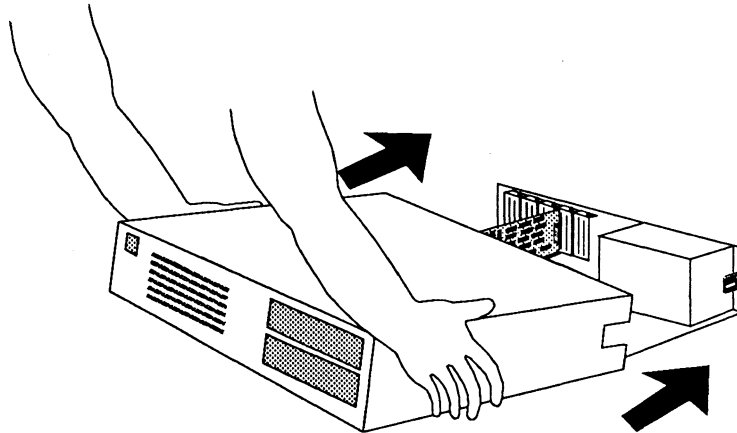
Install the Cable

Plug the single, "D"-type connector on the TX-8s connector at the back of the PC. Secure the connector by tightening both screws.

Reinstall the System Cover

Replace the cover. Slide it toward the rear of the computer.

When the cover is all the way to the rear, align the cover screws and threaded tabs. Tighten screws (clockwise) with a screwdriver.



Replacing the system cover

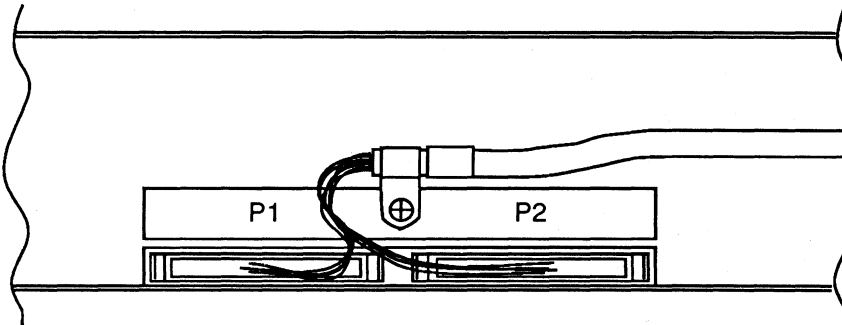
Connect the Cable to the Tape Drive

Follow these steps to properly connect the cable to the tape drive:

1. Locate the drive connectors: On the rear of the drive locate two card edge connectors that match the split end of the cable.
2. Identify drive connectors P1/J1 and P2/J2: These connectors are normally labeled, on some drives they have different names. The lower number connector corresponds to P1 and the higher one to P2. On drives with two sets of connectors, use either set (refer to your tape drive manual for more detail).
3. Identify the pin 1 end: each connector has pins, numbered 1 through 50. On some drives the chassis identifies the pin 1 end of the connector; on others you must check the connector.
Close inspection reveals a number silk screened on the board or formed in copper. Finding a 1 or a 2 shows the pin 1 end.
4. Connect the cable: the cable splits at the drive end. Connectors are labeled J1 and J2. The pin 1 end of each is marked "1". Press J1 into the P1 connector. Press J2 into the P2 connector.

Grounding

The drive must be grounded to meet FCC requirements. Place the “P”-clip around the cable where the silver mesh is exposed. Now use a Phillips screwdriver to screw the P-clip to the drive, just above the P1 and P2 connectors. This grounds the cable and keeps the cable from accidentally pulling out of the drive.



Cable and “P” clip installed on an OD3210 tape drive

Installing the Software

Installing the software is easy. Make sure that all your cables and peripherals are reconnected. Now turn on your PC and tape drive.

Make a “working” copy of the disks: Turn on your PC and tape subsystem. Follow the procedure in the DOS manual for copying a disk. Use the `diskcopy` command.

If you look at the files on disk, you will not see many. The files are compressed—**do not copy the files from the disk to your hard disk!** Instead, insert disk #1. Move to the drive that the disk is in. Now type `INSTALL`, and the install program will automatically decompress the files and install them on your hard disk. There will be prompts, asking you what directory to use, or asking you to insert a disk.

When this is complete, run **Config** and then test the system.

TX-8 Configuration

Changes are made to the TX-8 configuration using a supplied program, **CONFIG**. **CONFIG** makes changes (patches) to the `ODI.RLB` file, including the configuration parameters you supply. The changes remain until they are changed again. In addition, two of the parameters (base I/O port and interrupt level) require changing settings on the TX-8.

The following parameters are configurable for the TX-8:

- The I/O port used as the controllers base address.
- Drive Type: Start/Stop or Streaming.
- The J2-50 Speed/Density select line.
- The Interrupt level used by the controller.
- The Tape Drive address.

Before proceeding, refer to your notes on address selection. If you did not change the switch settings, the address is '360.' If you changed the settings, determine the value.

The CONFIG Program

The config program must be run for the ODI .RLB file.

To invoke **CONFIG**, type: `c:\> config` from the command prompt followed by **(RETURN)**. A message similar to the one below will appear on the screen:

```

                UTILITY CONFIGURATION PROGRAM
    Vers 1.8  8/23/85 (c) 1985 OVERLAND DATA, INC.
    Enter filename<cr> to configure or <cr> to quit.
    filename -->
    
```

Type **ODI.RLB (ENTER)**. **CONFIG** will display:

```
Configure Tape Controller
```

```
Enter new value<cr> to set new value, or <cr> to leave value unchanged.
```

CONFIG searches the target files for the configuration data and displays the current values:

```
Item to change: Current Value:
```

Base Port

The TX-8 requires eight contiguous I/O ports. This configuration parameter specifies the first (lowest) address in the selected address range. With an address of 360 (hex) (factory default), the controller will occupy I/O addresses 360 (hex) to 367 (hex). You can change the I/O base port on a TX-8 controller without using **CONFIG** if the new base port you select is listed below.

DIP Switch	1	2	3	4	5	6	7	8
Address Line	A10	A9	A8	A7	A6	A5	A4	A3
360 (default)	ON	OFF	OFF	ON	OFF	OFF	ON	ON
300	ON	OFF	OFF	ON	ON	ON	ON	ON
310	ON	OFF	OFF	ON	ON	ON	OFF	ON
320	ON	OFF	OFF	ON	ON	OFF	ON	ON
330	ON	OFF	OFF	ON	ON	OFF	OFF	ON
340	ON	OFF	OFF	ON	OFF	ON	ON	ON
350	ON	OFF	OFF	ON	OFF	ON	OFF	ON

Set the DIP switch (in the board's upper right corner) as shown, and the I/O base port reconfiguration is complete. Normally, the software running with the controller would also have been reconfigured, but the supplied utility programs (as well as **mt0** and **ALLTAPE**) include an automatic configuration routine which looks for the TX-8 controller at each listed address in turn, and configures accordingly if the board is found.

The auto-configure feature is activated by selecting a base I/O address of 0 in **CONFIG** (this is the factory default).

Unlisted base port addresses require you to determine the DIP switch settings, change the switches, and run **CONFIG**.

```
ODI Controller Address (0 for PS2): = 0000 (base 16)
New Value:
```

Drive Type

Older model Start/Stop drives require programs to wait until the Formatter Busy signal goes false before issuing new commands. By default, the utility package does not do this. If you have a Start/Stop drive, and **xTEST** returns block size errors, then you may need to change this configuration parameter.

It is hard to predict every configuration incompatibility symptom. We suggest you run **xTEST** using the standard configuration. If **xTEST** reports problems, check the board installation, cable connection and hardware configuration. If these are correct, then check the configuration parameters for possible conflicts.

```
START/STOP DRIVE(1), NORMAL(0) = 0 (base 10)
New Value:
```

J2-50 Speed/Density Select Line

This typically controls speed selection in a dual-speed drive. Our software uses this to dynamically control speed (seeks use higher speed). If your drive uses this as a density select line (older drives may), the program won't work (you may see data compare errors). To fix the conflict, check the drive manual, and set density selection to local control. If this is not possible, use **CONFIG** to set this line to LOW (value 1) or HIGH (value 2).

```
J2-50, Normal (0), LOW(1), HIGH(2) = 0 (base 10)
New Value:
```


Interrupt Level

A source of interrupts is needed for the TX-8. The table below lists available interrupts in a PC-AT. The jumper can be set for any one of the IRQ levels.

The TX-8 default is IRQ 0 (the PC's timer tick); this interrupt doesn't use a jumper. The board is shipped jumpered for IRQ 10. This doesn't cause a conflict. When IRQ 0 is selected, any jumper on the header pins is ignored. To reconfigure from IRQ 0 to IRQ 10, you don't need to remove the controller to move the jumper.

IRQ Level	TX-8 Needs Jumper	Possible System Conflict
0 (Timer Tick)	NO (default)	Multitasking DOS or TSR
3	Yes	COM 2
4	Yes	COM 1
5	Yes	Parallel Port 2
6	Yes	Floppy Disk Controller
7	Yes	Parallel Port 1
9	Yes	Software redirects to IRQ2
10	Yes (default position)	
11	Yes	
12	Yes	
14	Yes	Hard Disk Controller
15	Yes	

Note: In PC-XTs, interrupt levels 9 through 15 are not available.

To reconfigure the interrupt, remove the board and locate header pins marked 3 to 15 (for boards marked with A to M, see "TX-8 Interrupts" in Appendix A) on the lower edge of the board. Move the shunt jumper to the pair of pins corresponding to the new interrupt level, reinstall the board and reconfigure the software.

```
Interrupt Level (0=timer tick):          = 0 (base 10)
New Value:
```

Tape Drive Address

A 9-track tape drive is accessed with a unit address from 0 to 7. The address can be determined and changed (consult your drive manual); most systems with a single drive address it as unit 0. The tape drive address is configured using `CONFIG` (described below). Addresses 0 through 7 are supported.

If you made no changes to the values, `CONFIG` will display:

No change was made to the configuration.

Do you wish to change any of these values? (y,n):

If you make a change, **CONFIG** displays the new values and then asks if you want to make further changes:

Do you wish to change any of these values? (y,n):

Continue making changes, or make changes to more files:

Do you wish to configure another file? (y,n):

If you answer Yes, **CONFIG** repeats the configuration process. A No reply will return you to DOS.

Drive address (0-7): = 0 (base 10)
New Value:

Test the Complete System

The following steps outline the procedure to test your personal computer tape subsystem. This procedure will write to tape (writing over any data already on the tape!), so be sure the tape you choose has nothing important on it.

1. Load a tape onto the drive: Use a scratch tape with a write enable ring. See your drive manual for the procedure. When the tape is loaded, place the drive online.
2. Execute **xTEST** to perform the following three tests:
 - Status Test. This makes sure that the drive is ONLINE, not FILE PROTECTED, and that cables are properly connected.
 - On-board Cache Test. This test assures the integrity of the TX-8 cache memory.
 - Tape Read/Write Test. This tests the subsystem's ability to write data, write filemarks, read data, read filemarks, and rewind. On completion of these tasks, **xTEST** will display:

All tests successful.

in the Status Window.

For any other **xTEST** outcome, check that the TX-8 is installed and configured correctly, that the cables are properly connected to the drive, that the drive is ONLINE and not FILE PROTECTED.

After successful completion of **xTEST** you are ready to proceed with any of the tape applications.

Controller Model TXi-16

This section describes the TXi-16 controller and its installation and configuration procedures. Before installing the card in your computer, read the entire TXi-16 section. Be sure you have all parts in the checklist and that you have the recommended system documentation.

Standard Features

The standard TXi-16 features are:

- 1MB cache buffer.
- Configuration data is stored in NOVRAM (non-volatile RAM).
- Supports GCR, PE, and NRZI densities.
- On-board microprocessor.
- 16-bit architecture.
- Densities: 800, 1600, 3200, and 6250 bytes per inch (bpi)
- Compatible with most Cipher/Perfec formatted tape drives
- Burst rate: 2 MB/s. Sustained transfer rate: 1.3 MB/s. Throughput depends on the computer, the program, the condition of the tape, and attributes of the hard disk.

Inventory Checklist

Check that your TXi-16 package includes the following:

- TXi-16 circuit board and bracket
- One round, shielded cable
- TXi-16 configuration tool (plastic rod)
- One "P"-clip for cable grounding
- Distribution software on either 5 1/4-inch or 3 1/2-inch disks.

Related Reading

This manual assumes some familiarity with MS-DOS and IBM PC/XT/AT hardware. References to operating system commands and I/O port designations in this manual assume operation under a single-user, single tasking version of MS-DOS.

The TXi-16 works with all IBM ATs and 100% compatibles, (i.e., the COMPAQ 286/386/486, the EVEREX "STEP" series).

The supplied software requires a minimum of 512 kB of RAM.

No DMA channel is required for operation.

Hardware Installation

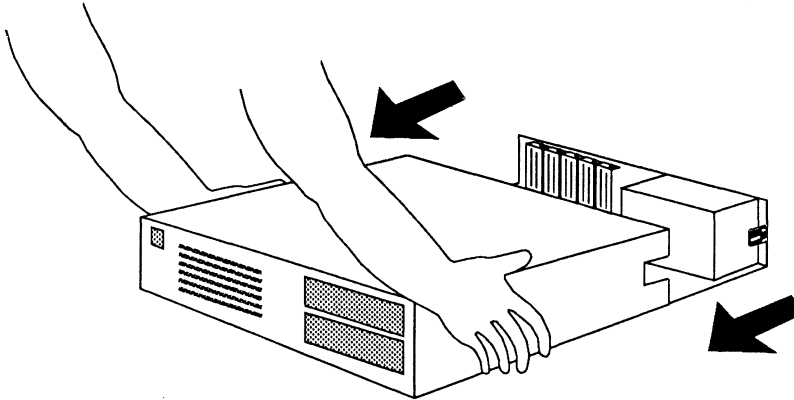
Remove the System Cover

Turn off and unplug the computer and all peripherals (note cable connections so that you can replace them later).

Move your keyboard and peripherals away from the work area. Place the PC in a position so you can work at its rear.

Remove all cover mounting screws (counterclockwise). Be sure to save all screws for the cover replacement.

Move to the front of the PC. Slide the cover to the front. When the cover will go no further, tilt it up, remove it, and set it aside.



Removing the system cover

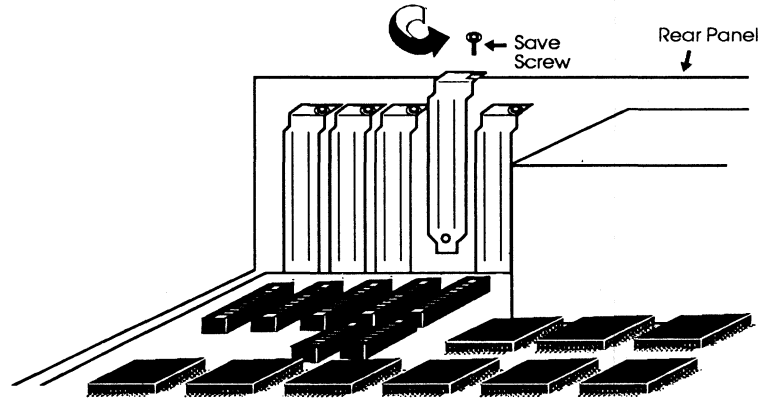
Select an Open Card Slot

Look at the inside left rear of your computer. The TXi-16 must be inserted into a 16-bit slot (one long and one short connector).

**The TXi-16 must be inserted into a 16-bit slot
(one long and one short connector).**

The board will not operate in an 8-bit slot.

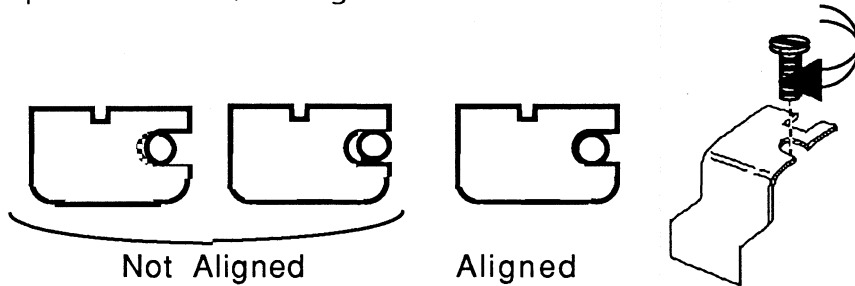
Remove the screw that holds the expansion slot cover in place. Turn counterclockwise to loosen the screw. Be sure you save the screw; it will be used to install the product later.



Removing the empty slot cover

Install the TXi-16

Hold the top of the TXi-16 and press it into the slot. The non-connector end should be in the vertical card guide as the TXi-16 is put into the computer. Align the board's retaining bracket slot with the hole in your computer's rear panel (below). Replace the screw, and tighten clockwise. Installation is done.



Aligning the TXi-16 retaining bracket

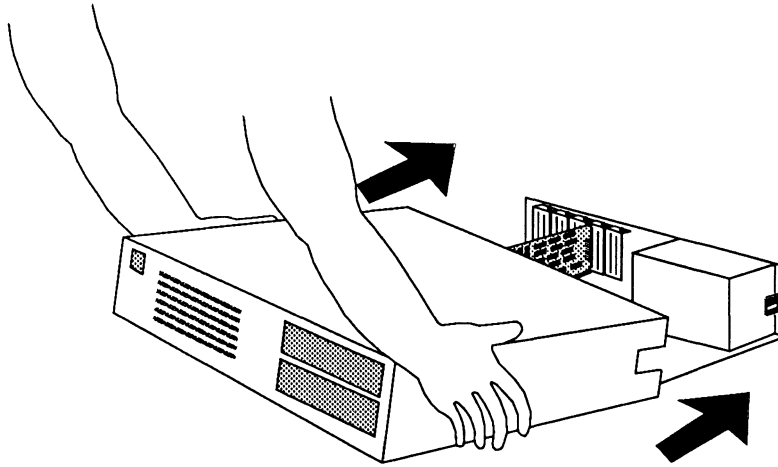
Install the Cable

Plug the single, "D"-type connector onto the TXi-16's connector at the back of the computer. Secure by tightening both connector screws.

Reinstall the System Cover

Replace the cover by putting it in a position as shown. Carefully slide it toward the rear of the computer.

When the cover is all the way to the rear, align the cover screws with the threaded tabs. Use a screwdriver and turn the screws clockwise to tighten them.



Replacing the system cover

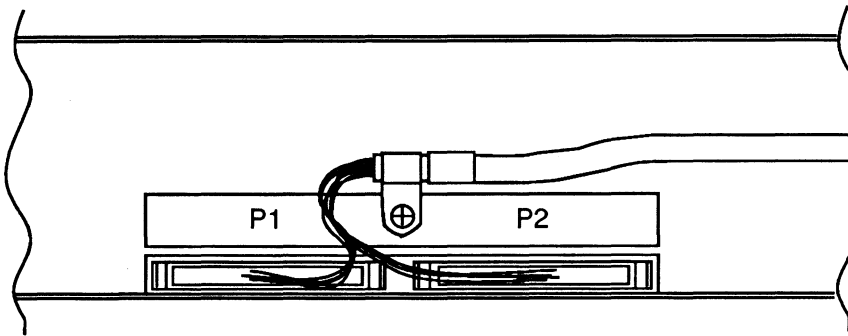
Connect the Cable to the Tape Drive

Follow these steps to properly connect the cable to the drive:

1. Locate the drive connectors: Check the rear of the tape drive and locate two card edge connectors that match those on the split end of the TX-8 cable.
2. Identify drive connectors P1/J1 and P2/J2: These connectors are normally labeled, on some drives they have different names. The lower number connector corresponds to P1 and the higher one to P2. On drives with two sets of connectors, use either set (refer to your drive manual for more detail).
3. Identify the pin 1 end: each connector has pins, numbered 1 to 50. On some drives the chassis identifies the pin 1 end of the connector; on others you must check the connector. Close inspection reveals a number silk screened on the board or formed in copper. Finding a 1 or a 2 shows the pin 1 end.
4. Identify and connect the cables: the cable splits at the drive end. The connectors are labeled J1 and J2. The pin 1 end of each is marked with a 1. Press J1 into the P1 connector. Press J2 into the P2 connector.

Grounding

The drive must be grounded to meet FCC requirements. Place the "P"-clip around the cable where the silver mesh is exposed. Now use a Phillips screwdriver to screw the P-clip to the drive, just above the P1 and P2 connectors. This grounds the cable and keeps the cable from pulling loose from the drive.



Cable and "P" clip installed on an OD3210 tape drive

Installing the Software

Installing the software is simple. Make sure that all your cables and peripherals are reconnected. Now turn on your PC and drive.

Make a "working" copy of the disks: Turn on your PC and tape subsystem. Follow the procedure in the DOS manual for copying a disk. Use the `diskcopy` or the `copy` command.

If you look at the files on disk, you will not see very many. Most files are compressed, so **do not copy the files from the disk to your hard disk!** Instead, insert Disk #1 and type: `INSTALL`, and the install program will automatically decompress the files and install them on your hard disk. There will be prompts, asking you what directory to use, or asking you to insert another disk.

When installation is complete, run the `TXISETUP` program from your hard disk (see below) and then test the system.

TXi-16 Configuration

Note: If there is an I/O address conflict between the TXi-16 and another card in your system, see `TXISETUP` (this chapter); program the card for a new I/O address, and patch the device drivers with it. Finish installing the card and software. Then complete the `TXISETUP` configuration. Once the card and software are reconfigured, return to this section to test it.

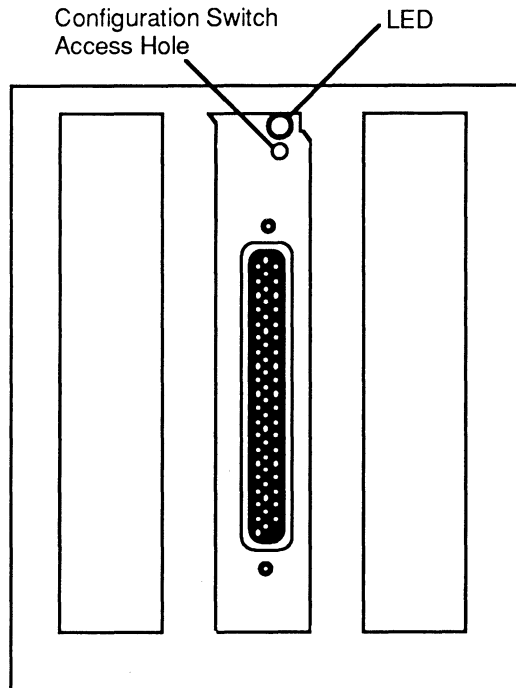
Configuration Procedure

`TXISETUP` configures the NOVRAM for proper I/O address and drive type. Run `CONFIG` on the `ODI.RLB`. Before starting configuration, be sure these tasks have been performed:

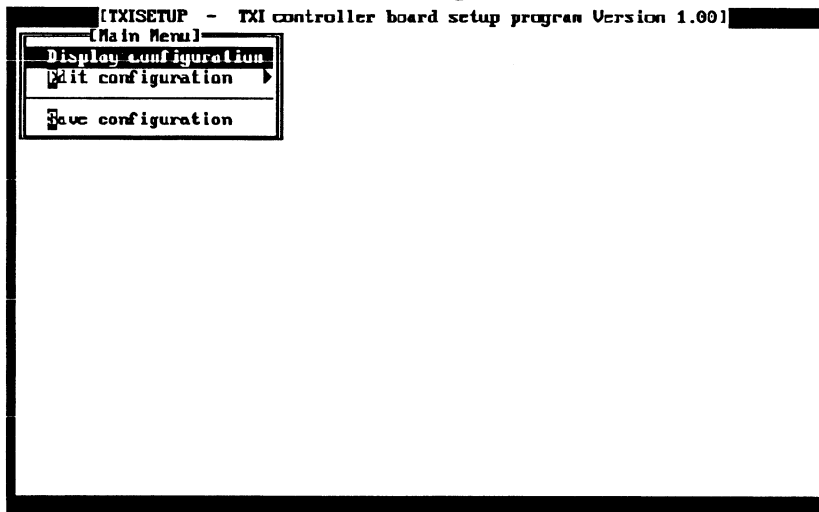
1. The Overland Data software must be installed in a directory on your hard disk.
2. The `ODI=` environment variable must contain a path to the above directory. To verify that the ODI environment variable exists, type: **SET (RETURN)**.
This lists the environment variables. If the ODI variable does not exist, the tape utility programs can only be used from the current directory. If an ODI environment variable does not exist, you must also run **TXISETUP** from this directory. To set the ODI environment variable, enter:
set ODI=C:\odiutils
(or whatever directory name you use)
This should be added to your DOS path.
3. A TXi-16 board is physically installed in the system.
4. The configuration tool, or small screwdriver is available.

TXi-16 Configuration

1. Turn the PC's power OFF and move the PC system unit to a position where you have access to the front and rear panels.
2. Locate the TXi-16 bracket on the rear of the computer.



3. Locate the configuration switch access hole just below the LED on the TXi-16 bracket.
4. Use the configuration tool or small set screwdriver to depress the switch just behind the access hole. You should feel the pressure of the spring in the switch as you insert the tool into the access hole. Keep the switch depressed.
5. Turn the power on while keeping the switch depressed. After about 2 seconds the LED on the TXi-16 bracket will glow both green and red simultaneously (and may appear orange).. This means the board recognizes the switch closure and is in configuration mode. Release pressure on the switch and remove the configuration tool.
6. Get into the ODI directory, and run **TXISETUP**. You should see a menu like the one following:



If an error window is displayed, be sure you are in the ODI directory and that the file ODI.RLB is present in the directory. If ODI.RLB is not in the directory, you must re-install the software.

Setting Tape Drive Type and the IO Address

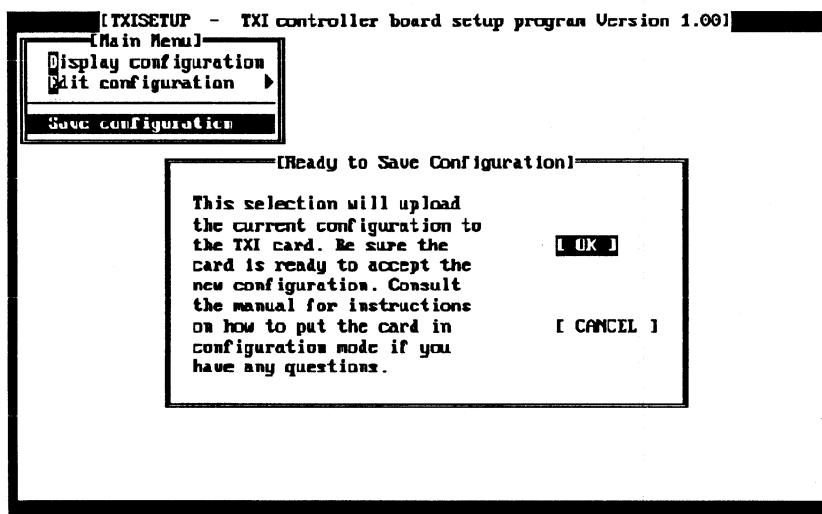
You can display the current configuration settings by selecting Display Configuration and change the configuration settings by selecting Edit Configuration.

To change board I/O address, select Edit Configuration, then select Change Board I/O Address. Select an address from the displayed list, using the arrow keys to highlight your choice and pressing **ENTER**. The TXi-16 requires eight contiguous

I/O ports for operation. This configuration parameter specifies the first (lowest) address in the selected address range. Thus, with an address of 340 (hex), the controller will occupy I/O addresses 340 (hex) to 343 (hex).

I/O Address	Notes
340 hex	DEFAULT SETTING
310 hex	
310 hex	
320 hex	
330 hex	
350 hex	
360 hex	
3E0 hex	
100 hex	
110 hex	
220 hex	
250 hex	
260 hex	
280 hex	
290 hex	
2A0 hex	

The TXi-16 base port is set with **TXISETUP**. Once the base port is set, the NOVRAM on the card will remember it until **TXISETUP** is used to change it. The *Your Choice* selection allows you to specify an I/O address if the listed addresses are not available. To change the tape drive type, choose *Select a New Drive Type*. A list of tape drives will appear. It is important to choose the drive you are using, since the TXi-16 uses this information to optimize performance and speed. Use the arrow keys to highlight your choice and press **(ENTER)**. This information is used by the TXi-16 to handle the interface differences some tape drives exhibit. If your drive is not on the list, choose *PE Streamer* or *Start/Stop Drive* depending on which type of drive you are using.



Selecting Save Configuration prompts you for confirmation; you may return to the main menu at this point by selecting CANCEL. To save the configuration, select OK.

8. Once OK has been selected, the prompt:
SAVING CONFIGURATION...
appears while **TXISETUP** is communicating with the TXi-16. When the configuration process is complete, **TXISETUP** will exit and a DOS prompt will appear.
9. Check that the LED on the rear panel shows green. This means the board has accepted the configuration and is ready.
10. Get into the ODI directory, and test the new system configuration with **XTEST**.
11. Turn off the power and arrange the PC for normal use.
12. Turn on the power and allow the system to boot up.
13. Since this board stores configuration data in non-volatile memory, removal of the PC unit cover is not necessary.

Test the Complete System

The following steps outline the procedure to test your personal computer tape subsystem.

1. If you have not done so, backup the software disks: Apply power to the computer and tape subsystem. Before testing the subsystem, use the procedure outlined in the MS-DOS manual for copying a disk. Use the DOS `DISKCOPY` or `COPY`

command to make backup copies of the supplied software disks.

2. Load a tape onto the drive: Load a scratch, write-enabled tape onto the tape drive. See your tape drive Operations Manual for the correct loading procedure.

This procedure will write to tape, so make sure the tape you choose has nothing important on it.

3. Execute the **xTEST** program to perform the following tests:
 - Status test. This test makes sure that the tape drive is ONLINE, not FILE PROTECTED, and that the cables are properly connected.
 - TXi cache is checked at power up. No cache test is performed during **xTEST**.
 - Tape read/write test. This tests the subsystems ability to write data, write file marks, read data, read file marks, and rewind. Upon successful completion of these tasks, **xTEST** will display the message:

All tests successful.
in the status window.

For any other outcome of **xTEST**, make sure the TXi-16 is properly installed and configured, that the cables are correctly connected to the tape drive, and that the tape drive is ONLINE and not FILE PROTECTED. After successful completion of **xTEST** you are ready to use the supplied applications.

After successful completion of **xTEST** you are ready to proceed with any of the applications in Chapters 3, 4, 5 or 6.

TXISPD

The TXi-16 (via drive selection in TXISETUP) automatically selects the highest speed available for your drive. In the event that you need to alter that speed, use the TXISPD program. Type:

TXISPD **(ENTER)**.

The program will show:

Currently drive speed selection is (the current valid selection).

1. Allow speed to be selected under program control.
2. Always use high speed.
3. Always use low speed.
4. Do not make any changes.

Your choice ==>

You must make a selection (use selection 4 if you simply want to escape without making changes) once you enter **TXISPD**.

SETCARD

Regardless of which Overland Data controller card you have, you use SETCARD to tell the software which driver to use. The default selection is the driver for the TX-8 and XL/2, so for the TXi-16 you MUST run this program.

Type: SETCARD **(ENTER)**

SETCARD will appear, telling you where the ODI.RLB file was found, what controller is presently selected and what drivers are available in your ODI.RLB file:

Found "ODI.RLB" in directory:

C:\ODI

SETCARD MENU:

Drivers available are:

1 "TX8_XL/2" << PRESENTLY SELECTED >>
2 "TXi"

type a number between 1 and 2 (or ESC to exit). -->

If you find that you do not need to change the present setting, press **(ESC)** now.

If you do need to select a new setting, press the number for the correct driver. SETCARD will ask you to confirm the selection. Press Y (yes) and **(ENTER)** to confirm, or N (no) and **(ENTER)** to abort the change.

Controller Model XL/2

This section describes the XL/2 controller installation and configuration procedures. Before installing the card in your PC, read the entire XL/2 section. Be sure that you have all parts in the checklist, that you have the recommended system documentation, and that you understand the procedure.

Standard Features

The standard XL/2 features are:

- Automatic configuration.
- Full hardware support of PS/2 Programmable Option Select.
- Supports GCR, PE, and NRZI densities.
- Densities: 800, 1600, 3200, and 6250 bytes per inch (bpi).
- Compatible with most Cipher/Pertec formatted tape drives.
- Burst data transfer rate: 900kB/s. Sustained data transfer rate: 632kB/s. System throughput will depend on the computer, the program, the condition of the tape, and attributes of the hard disk.

Inventory Checklist

Check that your XL/2 package includes the following:

- XL/2 circuit board assembly
- One round, shielded cable
- One "P"-clip for cable grounding
- Distribution software on a 3½-inch diskette

Related Reading

This manual assumes some familiarity with MS-DOS and the IBM PS/2 hardware. All references to operating system commands and I/O port designations in this manual assume operation under a single-user, single tasking version of the DOS operating system (OS/2 is not supported).

System Requirements

The XL/2 works with any IBM PS/2 with the Micro Channel Architecture expansion bus. (this includes Models 50, 60, 70, 75, 80 and future models with the Micro Channel Architecture bus.)

The supplied software requires a minimum of 512kB of RAM.

No DMA is required.

A copy of the IBM Reference disk is required to configure the XL/2 when it is installed.

Hardware Installation

Remove the System Cover

Locate the section in the IBM Quick Reference booklet that explains how to remove the system unit cover.

1. Turn off and unplug the computer and all peripherals (note the location of all cables and cords for reconnection later).
3. Check that the cover lock is unlocked.
4. Loosen the cover screws. If they are too tight, use a coin.
5. Remove the cover by sliding it forward approximately 2 inches and lifting it.

Install the XL/2

Locate the section in the IBM Quick Reference booklet that explains how to install an adapter.

1. Locate an open expansion slot.
2. Remove the expansion slot cover. To do this, loosen the screw (you can use a coin) and remove the expansion slot cover.
3. Install the adapter by pressing it into the expansion slot until the adapter clicks into place. Then tighten the screw.

Reinstall the System Cover

Locate the section in the IBM Quick Reference booklet that explains how to install system unit cover.

1. Install the cover by placing it over the system unit approximately 2 inches from the rear and sliding it on.
2. Tighten the cover screws.
3. Connect all cables and cords to the rear of the system unit. Then plug all computer power cords into electrical outlets.

Connect the Cable to the XL/2

Plug the single “D”-type connector into the XL/2’s connector at the back of the computer. Tighten both connector screws.

Connect the Cable to the Tape Drive

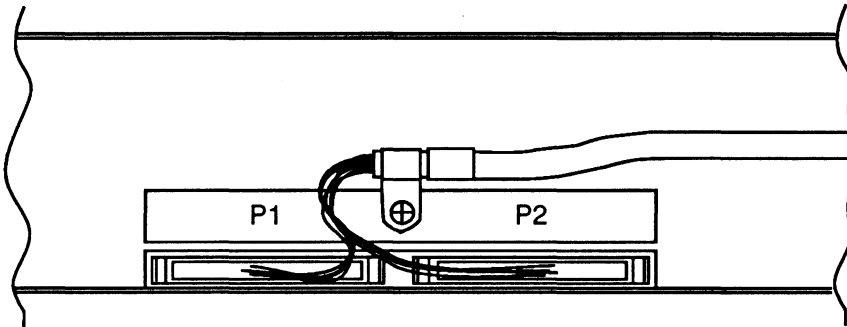
Follow these steps to properly connect the cable to the drive:

1. Locate the drive connectors. Check the drive to find two card edge connectors matching the cable connectors.
2. Find the drive connectors, P1 and P2 (some drives use other names, with the lower number being P1 and the higher P2). For drives with two sets of connectors, use either set.
3. Identify the pin 1 end. Each connector has pins, numbered 1 to 50. On some drives the chassis identifies the pin 1 end of

- the connector. On others you must examine the connector for a number (1 or 2) on the board, indicating the pin 1 end.
4. Connect the drive end of the cable (it splits in two at the drive end). These connectors are J1 and J2. The pin 1 end of each connector shows a 1. Press J1 to the P1 connector. Attach J2 to the P2 connector. Pin 1 of each cable connector attaches to pin 1 of each drive connector.

Grounding

The drive must be grounded to meet FCC requirements. Place the "P"-clip around the cable where the silver mesh is exposed. Use a Phillips screwdriver to screw the "P"-clamp to the drive, above the P1 and P2 connectors. This grounds the cable and keeps the cable from accidentally pulling loose from the drive.



Cable and "P" clip installed on an OD3210 tape drive

XL/2 Configuration Overview

Configuration of the XL/2 Tape Controller is easy. There are no switches to set or jumpers to move. With the card installed in the computer, the software on the IBM Reference Diskette, in concert with the Automatic Configuration File from Overland Data (file @7E76.ADF on the UTILITIES disk #2), handles the process.

This section describes the configurable settings. Usually you let the Configuration Program automatically select the appropriate settings. In some cases you may need to change them to solve a configuration conflict or to better match your environment.

The configurable XL/2 options and their default settings are:

Option	Default Setting
I/O Address	360 hexadecimal
Interrupt Level	10 decimal
Drive Address	0
Tape Density/Tape Speed	Normal
Start/Stop or Streaming	Streaming

Address

The XL/2 uses 8 contiguous I/O addresses and can be addressed to any base address on even boundaries of 8 (0000 to FFF8 hex). Much of the address space below 400 hex is used by IBM. The default address is 360 hex. If this conflicts with another device (shown by * next to the selection), make another selection.

Interrupt Level

The IBM PS/2 has 11 interrupt levels available on the Micro Channel bus. This selection allows you to select any one of these, or to disable the hardware interrupts. The Overland Data software chains into the DOS system clock interrupt (IRQ 8) if the hardware interrupt is disabled.

The default hardware interrupt level is 10 (decimal). If this conflicts with another device, select another interrupt.

Drive Address

This selection allows you to specify which drive address the XL/2 will select when the unit is first powered on. Normally, you should specify drive address 0.

Tape Density/Tape Speed Select

Pin number 50 on the J2 connector is used on some drives to select either one of two speeds or one of two tape densities. Refer to the tape drive documentation for the interpretation of this signal by your tape drive. The default setting sets this signal to Normal. The XL/2 only supports speed selection—not densities.

Streaming or Start/Stop

Selecting streaming or start/stop mode alters the commands sequence issued by the software to the drive. Streaming is the normal choice (for streaming tape drives). Some older drives have trouble with commands issued before the formatter-busy (IFBSY) line indicates that the formatter is not busy. This

problem sometimes causes the tape drive to quit functioning until it is reset. In these rare cases, try the start/stop setting. If you select start/stop on a streaming drive, you may experience degraded performance. This will manifest itself as an increase in tape repositioning.

Configuring the PS/2 for the XL/2

When a new adapter card is installed, the IBM PS/2 detects the change in its hardware configuration. This results in an error message and a request for the Reference diskette. As the computer goes through its configuration sequence it uses files on the Reference Diskette that tell it how to configure the card. If you haven't made a copy of the Reference diskette, do so.

The steps to configure the PS/2 for the XL/2 are:

- Start the working copy of the Reference diskette.
- Copy the XL/2 configuration file to the Reference disk.
- Activate the XL/2 configuration file.

Start Reference Diskette

Locate the section in the IBM Quick Reference booklet that explains how to start the Reference diskette.

1. Insert the Reference diskette into drive A. Make sure the diskette clicks into place.
2. If the PC is off, turn it on. If on, press **(CTRL)+(ALT)+(DEL)**, then release them. This will reboot the system into DOS.
3. Wait for the Reference disk sign-on message. It will ask if you want to run auto-configure. Answer no (press **N**).

Copy XL/2 Configuration File to Reference Diskette

1. Select Copy an option diskette from the menu. Wait for the prompt for the option diskette to be placed in drive A.
2. Place the XL/2 UTILITIES disk in drive A and press **(RETURN)**. Wait for the prompt to put Reference diskette in drive A.
3. Place the Reference diskette in drive A and press **(RETURN)**. Wait until the menu appears.

Activate XL/2 Configuration File

1. Select Set Configuration from the Main Menu.
2. Select Run Automatic Configuration from the Set Configuration Menu. Remove the Reference Diskette.

3. Press **CTRL**, **ALT** and **DEL** at the same time, then release them. This will reboot the system into DOS.

Installing the Software

Installation is simple. Make sure that all your cables and peripherals are reconnected. Now turn on your PC and drive.

Make a “working” copy of the disks: Turn on your PC and tape subsystem. Follow the procedure in the DOS manual for copying a disk. Use the `diskcopy` command.

If you look at the files on disk, you will not see many. The files are compressed, so **do not copy the files from the disk to your hard disk!** Instead, insert Disk #1 and type `INSTALL`. The install program will decompress the files and put them on your hard disk, prompting you as needed.

When installation is done, test the system.

Test the Complete System

These steps outline the subsystem test procedure.

1. Before testing the subsystem, make a backup of the disks—follow the procedure in the MS-DOS manual for copying a disk. Use the DOS `DISKCOPY` command.
2. Load a tape onto the drive: Load a scratch, write-enabled tape onto the tape drive. See your tape drive Operations Manual for the correct loading procedure.
This procedure will write to tape, so make sure the tape you choose has nothing important on it.
3. Execute `xTEST` to perform the following three tests:
 - Status test. This checks that the tape drive is online, not file protected, and that the cables are connected.
 - On-board cache memory test. This test assures the integrity of the XL/2 cache memory.
 - Tape read/write test. This tests the subsystems ability to write data, write file marks, read data, read file marks, and rewind. Upon successful completion of these tasks, `xTEST` will display the message:
`All tests successful.`
in the status window.

For any `xTEST` result, check that the XL/2 is properly installed and configured, the cables are correctly connected, that the drive is online and not file protected. After successful completion of `xTEST` you are ready to use the supplied applications.

9-Track Tape Subsystem Tools

XTEST tests the integrated TX-8, XL/2 or TXi-16 tape subsystem.

WHAT shows each source module's version in the tape programs.

XTEST

XTEST provides a rapid test of the integrated tape subsystem with Overland Data's controllers. After installing the controller, you should run `xtest` before attempting to execute any other tape utility. Passing the diagnostics indicates that your system is correctly installed and the hardware is working properly.

Features

XTEST tests the following controller and tape functions:

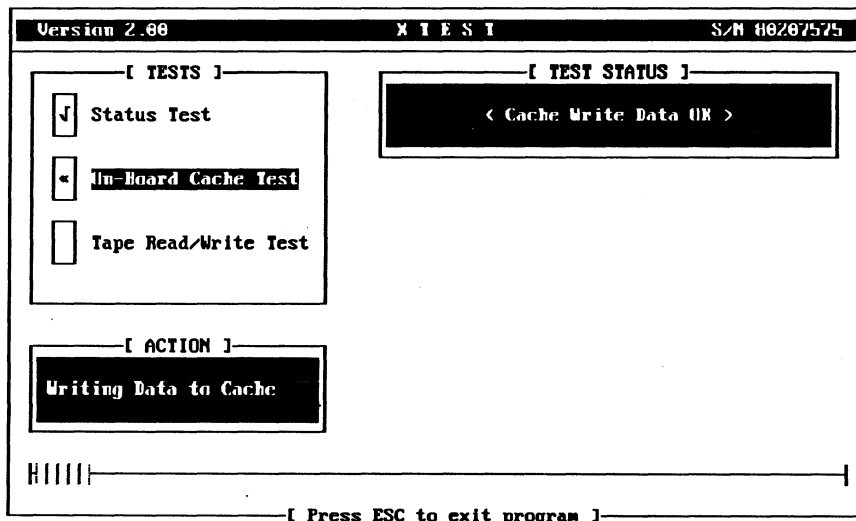
- Status Test
- Onboard Cache Test (TX-8 and XL/2 only)
- Tape Read/Write Test

Use

Find a scratch tape, insert a write enable ring in the back of the tape reel, and load it onto the tape drive. Put the drive ONLINE.

Note: This procedure will write to tape. Make sure that the tape you choose has nothing important written on it.

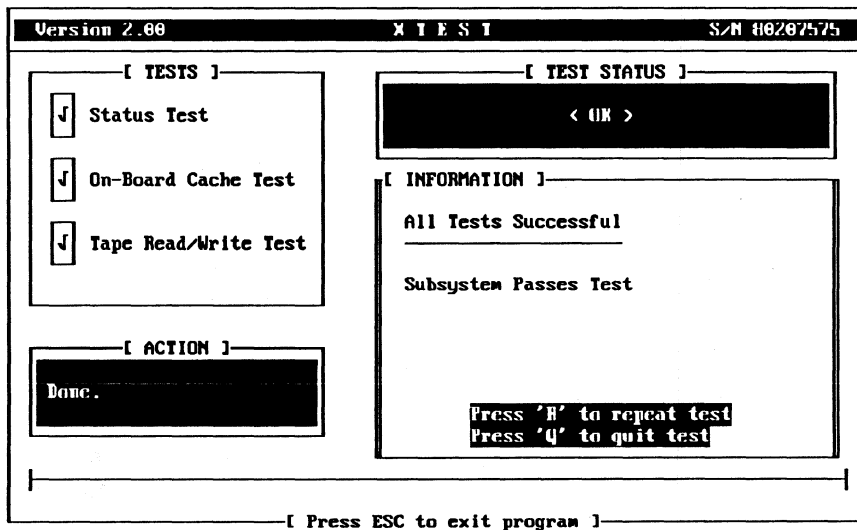
Boot DOS and run **XTEST**. **XTEST** displays the following :



XTEST performs the three tests listed in the Tests Window. As it completes each test, it places a check mark (√) in the appropriate box. The Test Status Window displays the status messages of each test. If an error occurs, the Test Status Window displays the exact error and the Information Window gives you trouble shooting guidelines. Be sure to read this information carefully before calling Overland Data. Sometimes problems with the controller or subsystem are simply incorrect cable installation or improper seating of the board.

The Action Window displays the action (for example, reading data, writing data, rewinding tape) of each test.

Upon successful completion of its tests, **XTEST** displays the following screen:



WHAT

Use **WHAT** to determine what source code modules were used to create your Overland tape software utility programs. This information is useful to Overland Data's software support in identifying and resolving any software problems.

Use

To use **WHAT**, type **WHAT** at the DOS command line followed by the full file name of each program that you want source version information for. **WHAT** recognizes the DOS wildcard characters * (matches any file name part) and ? (matches any single character).

For example, to display source version information for utility programs **XTEST** and **CONFIG**, type:

```
C:\> what xtest.exe config.exe
```

```
XTEST.EXE:
```

```
xtest.c          1.8 - 6/11/88
```

```
alltape.c        1.8 - 6/10/88
```

```
CONFIG.EXE:
```

```
config.c         1.8 - 8/23/88
```

Note that we specified the complete file name for each program (that is, **XTEST.EXE**, not just **XTEST**).

The version number and date indicate which software modules you have. We can determine if you have the newest release.

To display source version information for all of the Overland utility programs, type:

```
C:\> what *.exe
```

Since * matches any file name part, *.exe will match all programs with extension exe.

Limitations

WHAT only works with software from Overland Data Inc.

Tape Applications

The Tape Applications are a set of tools designed to optimize your PC's 9-track tape subsystem. This chapter describes the utilities for testing, data interchange and configuration. Chapter 6 describes the programming interface. **TXISETUP** and **CONFIG** let you configure parameters (dependent on the controller) and make these parameters consistent for the Tape Utilities. They are in Chapter 2, Installation and Configuration.

Note: We keep our manuals up-to-date, but last-minute releases may modify information in this manual. Refer to any addenda included with the manual, *and* read the `readme.doc` file on the release disk *before* using the utilities.

For data interchange, Overland Data provides:

- **DEPOT** (Data Exchange Program with Optional Translation) lets you to read or write tapes; and allows conversion between ASCII and EBCDIC. You may also append data to existing tapes and read from an arbitrary tape location.
- **DEPOT2** allows command line execution (for convenient batching) and automatic insertion of Carriage Return Line Feed (CRLF) delimiters after each record read.
- **DEPOT3** is the most flexible data conversion utility for reading tapes. By using a script file, you can define different translation criteria at the field (byte) level for each record.

FDUMP lets you quickly examine data blocks from a 9-track tape.

LABEL is a step-by-step menu driven utility, allowing you to create IBM or ANSI Standard labeled tapes.

TAPEUTL provides normal handling of labeled and unlabeled tapes.

TDE is designed to extract data from tape files and transfer it to a PC file. The data is extracted file-by-file from tapes with fixed length records, with various types of conversions being possible.

TAR is a backup application. It also provides data interchange capabilities with XENIX and UNIX systems.

DEPOT4

DEPOT4 ties together some of our most popular programs: **FLASHBAK**, (a separate option); **DEPOT2**, see page 60; **FDUMP**, see page 84; **LABEL**, see page 92; **TAR**, see page 175, and **NOVELL TAR** (an optional Novell backup/restore version of **TAR**). This lets you operate any of these utilities from a single, easy-to-use program. From **DEPOT4**, with the standard software package, you can:

- Read labeled or unlabeled tapes
- Write labeled or unlabeled tapes
- View the contents of any tape on your display
- Back up or restore your DOS filesystem
- Read or write UNIX / XENIX **TAR** tapes

You don't need to be a 9-track expert to use **DEPOT4**. The program provides on-screen explanation and help as long as you are within the program.

In **DEPOT4**, you can back up or escape by pressing **(ESC)**.

Before you can use **DEPOT4**, you must complete installation of the Overland Data utilities. See the instructions in Chapter 2.

Setup

DEPOT4 uses the DOS environment to locate the directory containing the Overland Data utility programs. This allows you to run **DEPOT4** from any sub-directory in your disk system.

For **DEPOT4** to function, when you put your Overland Data utility programs on your hard disk, you must install them in a sub-directory of their own, such as `C:\ODIUTILS`.

With the utilities installed (using the directory example above), enter the DOS command:

```
set ODI=C:\odiutils
(or whatever directory name you use)
```

This variable allows **DEPOT4** to locate the other Overland Data utilities—without it, **DEPOT4** will not work. To avoid entering this each time you boot up, add **set ODI=C:\odiutils** to the DOS path in your `autoexec.bat` file. You can add the pathname for the Overland Data utilities to your DOS path in your `autoexec.bat` file, ensuring that this environment variable is set when you boot up your PC.. **DEPOT4** is ready to go to work.

Using DEPOT4

Using **DEPOT4** is easy. Each step of the way, **DEPOT4** explains what you are doing and how to set up for the next step. If you make an error or change your mind, simply **(TAB)** to the data to change it or press **(ESC)** to back up until you can start again or escape **DEPOT4**.

Starting DEPOT4

If you placed the name of the directory containing the Overland Data utilities in your DOS PATH, then simply type:

depot4 (ENTER)

at the DOS prompt. If you have not modified your path, use the full path name (using our example from above):

C:\ODIUTILS\DEPOT4 to execute the program.

The DEPOT4 Screen

When you start **DEPOT4** you see two areas, labeled **Main Menu** and **Full Time Help**. The **Main Menu** consists of a list of selections, one of which is highlighted. We will refer to the highlighted cursor as the **Select Bar**. The **Full Time Help** window is always active, and contains a brief description of the item that the **Select Bar** is currently on.

Menus

DEPOT4 is a menu based program. Learning to “navigate” and select menu items is the first easy step in mastering **DEPOT4**.

You need to learn to:

- Move between menu items.

- Select menu items or options.

The arrow keys (**↓** and **↑**) are used to move from one menu item to another. The **↓** moves the **Select Bar** down one entry. The **↑** moves the **Select Bar** up one entry. To move the first item in any given menu, press **(HOME)**. To move instantly to the last item in a menu list, press the **(END)**. An alternative to using the arrow keys to move between menu items is **(TAB)**. Use **(TAB)** to move the **Select Bar** down one entry and **(SHIFT)+(TAB)** to move up one entry.

Hint: the last item in many menus is **Execute**. To begin the tape operation, press **(END)** to get to **Execute** immediately.

To select menu items press **(ENTER)** or (space bar).

Any individual item in a menu can also be selected by pressing the key corresponding to the highlighted character in the item. You can always change your mind while in the **DEPOT4** program. Pressing **(ESC)** will exit from any menu or dialog box. From the main menu, pressing **(ESC)** is the same as selecting the **Quit** option, and you will be asked if you wish to exit the program. Selecting **YES** will leave the program and bring you back to DOS. Selecting **NO** brings you back to the main menu.

Dialog Boxes

Dialog boxes pop up in response to certain menu selections, allowing you to enter filenames or numbers. You may edit the filename or number entry area with the following keys:

(ENTER)	Completes entry and moves to the OK dialog button.
(BACKSPACE)	Deletes the character left of the cursor.
(DELETE)	Character under the cursor is deleted.
(HOME)	The cursor is moved to the start of the entry.
(END)	The cursor is moved to the end of the entry.
(←)	Cursor moves left one character, up to the entry's start.
(→)	Cursor moves right one character, up to the entry's end.
(CTRL)+(END)	Delete from the cursor to the end of the entry.
(CTRL)+(HOME)	Delete from the start of the entry to the cursor.

Simple filenames refer to files in the directory in which you started **DEPOT4**. Full pathnames can be entered, which may refer to any drive and sub-directory.

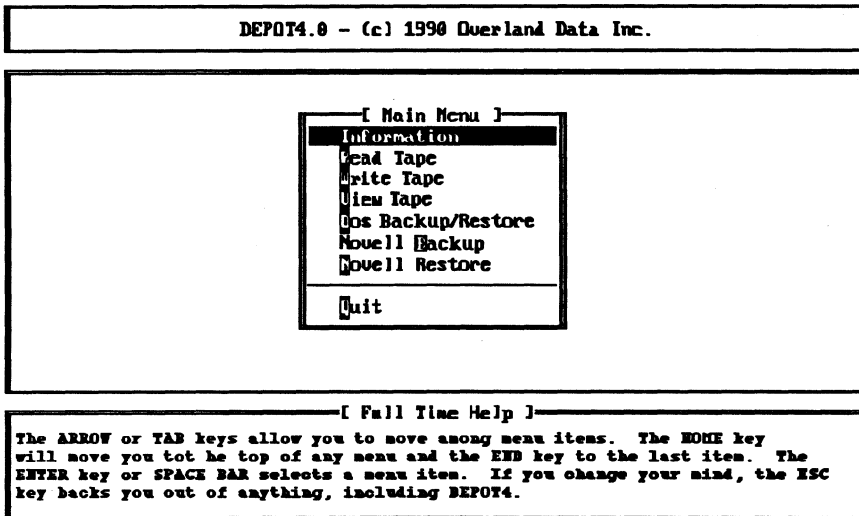
Some menu options require that you enter numbers. The number dialog box lets you enter numeric values. Negative numbers are not meaningful and are ignored (treated as zero).

Help

The bottom of the screen always contains a help message relevant to the menu item that is currently highlighted.

The Main Menu

DEPOT4 allows reading and writing data and backup tapes, as well as viewing a tape contents of on your video display.



This screen is the heart of **DEPOT4**.

Using the arrows moves the Select Bar, to a given item. If that item offers secondary choices, another menu appears which works in the same manner. If an item leads to a program or a data window, no menu appears. Press **(ENTER)** to select the item. For example, selecting Information (press **(ENTER)** to do so—try it now, press **(ESC)** to exit when you're done looking at the screen) shows you an overview of the **DEPOT4** information.

Programs and Data Windows

Remember each utility program is a stand alone and can be used alone. Detailed explanations for each standard utility are as follows: **DEPOT2**, page 60; **FDUMP**, page 84; **LABEL**, page 92; and **TAR**, page 175.

Change or select an item by highlighting it and pressing **(ENTER)** (this will not run the program until you select **Execute**).

If a selection toggles between two choices, **(ENTER)** changes that item. If it requires you to enter data, it opens a dialog box.

Continue to change items as needed, pressing **(↓)** to move to the next item, until you highlight **Execute**. Press **(ENTER)** to execute the program as you have directed, press **(↓)** to move to the first item to let you to change it; press **(ESC)** to escape.

DEPOT4 will return you to the **DEPOT4** program after you exit from any of the Overland Data utility programs.

Chapter 4—9-Track Tape Applications

Selecting Read Tape gets you the Read Unlabeled Tape window. Enter the filename (the item automatically highlighted when you enter the screen), and press \downarrow to move to the next item. The small DEPOT2 Command Line window shows the DEPOT2 command line that will be selected.

DEPOT4 Revision: 1.2 - (c) 1990 Overland Data Inc.			
[Read Unlabeled Tape]			
Filename	"DATAFILE"		
Translate	NO	Read To EOF	YES
Verbose	YES	Blocks To Read	0
Strip7	NO	Records To Read	0
Record Size	0		
Block Factor	0		
Skip Files	0		
Skip Blocks	0		
EXECUTE			
[Depot2 Command Line]			
depot2 /u /M /n DATAFILE			
[Full Time Help]			
Use this function to enter the name of the file you wish to create on the disk when reading from tape. Multiple filenames or DOS wildcard characters (* or ?) will not be recognized. You may enter a complete path name if you wish e.g.:			
D:\database\maillist.txt			

Selecting Write Tape brings up a menu where you select labeled or unlabeled tape.

DEPOT4.0 - (c) 1990 Overland Data Inc.	
[Main Menu]	
Information	
Read Tape	
Write	[Write Tape]
View	Write Labeled Tape
Ops	Write Unlabeled Tape
Move	
Novell Restore	
Quit	
[Full Time Help]	
Use this function to write tapes with IBM or ANSI tape labels. A tape label consists of a short data file written to the tape before the user data is written, which contains data that identifies the following data file.	

Selecting labeled tape runs Overland Data 's **LABEL** program, which offers you a data window where you may set the options. Selecting unlabeled tape allows you to choose the appropriate options for **DEPOT2**. Selecting Execute will execute **DEPOT2** using the options you selected. The small Depot 2 Command Line window shows the command line which will be executed.

Selecting View Tape will run **FDUMP**.

DOS Backup/Restore will offer you the choice of Standard Backup/Restore or **FLASHBAK**. Run **FLASHBAK** if you have purchased that (optional) program, otherwise it will inform you that it cannot find **FLASHBAK**. Or run Standard Backup/Restore (**TAR**).

Selecting Novell Backup will run **NOVELL TAR**, allowing you to use the **TAR** backup program for Novell, if you have the Novell version of **TAR**. If not, it will inform you that this selection requires the purchase of that option. The small **TAR** Command Line window shows the command line which will be executed.

Selecting Novell Restore will run **TAR**, allowing you to restore your backed up Novell system if you have the Novell version of **TAR**. If not, it will inform you that this selection requires the purchase of that option. The small **TAR** Command Line window shows the command line which will be executed.

DEPOT

DEPOT (Data Exchange Program with Optional Translation) lets you transfer data between system disk and 9-track half-inch tape. You can transfer data to or from tape in its binary form, or request translation between ASCII (American Standard Code for Information Interchange) and EBCDIC (Extended Binary Coded Decimal Interchange Code).

Features

DEPOT allows the following:

- Data interchange from tape to disk.
- Data interchange from disk to tape.
- Conversion from ASCII to EBCDIC.
- Conversion from EBCDIC to ASCII.
- Positioning to arbitrary location prior to data read.

You can append data from disk files to existing data tapes, or you can record data at the beginning of a new tape.

You can recover data recorded on tape by specifying file number, data block within the file, and the amount of data to read.

Character translation takes place (if required) in system memory before **DEPOT** writes the data to tape or disk.

Storing Data on Tape

The data to be stored on tape must exist in a disk file. **DEPOT** transfers the entire disk file to tape without interruption. The disk file is not modified.

DEPOT stores data on tape as sequential files of data blocks. Each data block is a binary image of the data in the specified disk file. **DEPOT** separates each file from the preceding file by one filemark, and terminates the last file with double filemarks to indicate the logical end of tape. **DEPOT** does not maintain a tape directory and does not insert any identification bytes.

You must specify the size of the data blocks to be written to tape. The tape block size can be any number between 1 and 65280 bytes.

In general, larger (2kB to 16kB) block sizes allow more data to be put on one tape. For example, a 2400-foot tape with 80-byte data blocks will yield approximately 3MB of storage. The

same tape written with 8192 byte (8kB) data blocks yields 40MB of storage.

You can append a file to a tape already containing data. **DEPOT** locates the two filemarks which indicate the logical end of tape, erases the last filemark, and writes the new file. **DEPOT** terminates the appended file with two filemarks and rewinds.

If the last sector of the disk file does not completely fill the last tape data block, **DEPOT** truncates the data block (it writes a short data block to tape).

Recovering Data From Tape

DEPOT recovers data by locating the specified file and the specified first block in the file and reading the data. Because there is no directory of the data on the tape, **DEPOT** locates the file by counting the number of filemarks from the beginning of the tape. It locates the data by counting the number of blocks (delineated by tape gaps) from the filemark. The first block of any file is number zero.

DEPOT reads the data from the tape regardless of the size of the recorded block (up to the maximum).

DEPOT will not operate if block size is larger than 65280 bytes.

DEPOT creates a new disk file to accept the data from tape. It records the data on the disk by using the operating disk file management system. If the disk file already exists, **DEPOT** asks for permission to overwrite it. If an error occurs during a tape operation or a disk operation, **DEPOT** will tell you about the problem and usually suggest something you may be able to do to correct it. A table later in this section describes your possible responses to an error condition.

Use

Find a blank tape, insert a write enable ring in the back of the tape reel, and load it onto the tape drive. This will write to the tape—be sure the tape you use has nothing important on it. When the tape is loaded, put the drive online.

Load **DEPOT** by typing:

```
C:\> depot
```

followed by **(ENTER)**. **DEPOT** displays a logon message, followed by a copyright notice, serial number and version. Then **DEPOT** sets

up the data buffer and checks the tape drive status. The drive must have a tape loaded and be on line.

Main Menu

If all is ok, **DEPOT** will display the menu:

```
1   Write a new tape
2   Append to an existing tape
3   Read a tape
4   Exit to DOS
    Your choice? -->
```

Enter the number corresponding to the operation you need.

Translation

If translation is needed, type **y** and **(ENTER)** at the prompt:

Is data conversion necessary?

Any other response will skip data conversion. If requested, **DEPOT** then displays the data conversion menu:

```
Enter:
1   -   If ASCII to EBCDIC
2   -   If EBCDIC to ASCII
      (1 or 2) -->
```

Writing a New Tape (Option 1)

This option assumes the tape mounted on the drive has no valid data recorded on it. **DEPOT** writes new data at the beginning of tape, overwriting any previously recorded data. When the file transfer is complete, **DEPOT** writes two filemarks, rewinds the tape and returns to the main menu.

You must enter the name of the disk file in response to:

Name of file to write to tape?

Enter the name in the form:

[d:]filename **(ENTER)**

where [d:] is an optional drive specification, and filename is the complete disk path name (including any extension).

To write file `C:\mail\list1`, type the path name as follows:

Name of file to write to tape? **C:\mail\list1**

DEPOT searches the file system for that filename. If it finds it, it opens it for reading. If it doesn't find the file, prints the message:

Can't open 'filename'.

Check your file name or path.

When you press a key, you will return to the main menu.

You must also indicate the size of the data blocks to write to tape. prompts:

Size of data block to write to tape (1-65280) -->
where 65280 is the maximum size allowed in bytes.

Prior to writing to tape, **DEPOT** checks the drive status to insure that the tape is write enabled. If it is not, **DEPOT** warns you of the problem and waits until you request it to try again (T), or stop (S). Choosing stop will take you back to the main menu.

When all is in order, **DEPOT** begins the data transfer and continues until the entire disk file is written to tape.

If translation is going on, **DEPOT** prints a dot for each tape block.

If an error occurs during writing, **DEPOT** makes ten or more attempts to write the data correctly. If unsuccessful, it displays:

An unrecoverable write error has occurred on
tape. Press any key to continue ...

You can not continue to write after a tape write error has occurred, so **DEPOT** takes you back to the main menu. A tape write error is usually caused by a tape being worn out, but it may also be due to a dirty head or other drive problem.

If the file does not completely fit on the tape, **DEPOT** writes what is already in the cache onto the tape. Then it says:

Physical End of Tape Detected.

Writing two filemarks and unloading tape.

Mount new tape.

Press any key to continue ...

When everything is in order, **DEPOT** will continue writing the rest of the file on the next tape.

Appending to an Existing Tape (Option 2)

With this option, **DEPOT** looks for two consecutive filemarks on tape and positions the recording head between them. It assumes there is no useful data beyond the second filemark. If the tape is full, **DEPOT** will warn you and go back to the main menu.

After positioning the tape to the logical end, `Appending to Tape` follows the same procedure as writing a new tape.

Reading a Tape (Option 3)

This option reads specified data from tape and (after any translation) puts it in the disk file. If either the logical EOT or physical EOT marker is encountered while positioning or reading, **DEPOT** will give a warning and allow you to continue (C) or stop (S). If the physical EOT marker is encountered while reading, you must select `Continue` in order to finish reading the present file on the present tape. If a file begins on one tape and ends on another, you must read the first part of the file (as above), and then load the next tape and read the first file as if it were a separate file. Then combine the two partial disk files into one file, using the DOS command:

copy file1.ext/b + file2.ext complete.ext **(ENTER)**

This is necessary as **DEPOT** has no way of knowing that the file continues on another tape. If you try to continue beyond physical EOT and have already passed logical EOT, it is possible for the tape to come off of the supply reel.

When **DEPOT** is reading a tape, many additional options are available which allow only selected data to be transferred. We cover each option individually and in the order usually encountered within **DEPOT**.

1. File Number. **DEPOT** locates the requested tape file by counting the filemarks from the beginning of tape. Provide this information in response to:

Number of filemarks to skip? -->

Since filemarks are only written at the end of a file, the filemark count is one less than the number of the file on tape. That is, the first file on tape is zero filemarks from the beginning of tape. And, the third file is two filemarks from the beginning of tape.

Warning!! DEPOT has no way to know whether the number of filemarks to skip is reasonable.
--

2. Blocks to Skip. **DEPOT** skips any number of blocks, allowing you to select a beginning block number within the file. Enter the number of data blocks to skip in response to:

Number of data blocks to skip? -->

DEPOT positions the tape to read the $n + 1$ data block, where n is the number of blocks to skip.

Warning!! **DEPOT** has no way to know whether the number of blocks to skip is reasonable.

3. Truncation. **DEPOT** provides an option to transfer only part of a data block, allowing easy handling of some card image files. When you request truncation, **DEPOT** transfers only a specified number of bytes of each data block. It appends a carriage return and a line feed to the data in system memory before it writes the data to disk. To request truncation, respond **y** followed by **(ENTER)** to the following prompt, and specify the number of bytes **DEPOT** should read from each record:

Truncate data blocks (Y/N) ? -->

Size of record to read ? -->

4. Disk File. When reading a tape file to disk, **DEPOT** creates a new file to receive the data.

Filename to create on disk? -->

Enter the disk file name in the form:

[d:]filename

where [d:] is an optional disk specifier and filename must include the extension and may include the path. **DEPOT** searches the file system for a file with the new name, and if one is found, prints

File <filename> already exists.

Overwrite? (Y/N)

If you respond with a **y** followed by **(ENTER)**, **DEPOT** removes the existing file and creates a new file with that name. Any other key allows you to enter a different filename.

DEPOT will warn you if the disk is too full, the path can not be found, a floppy diskette is write protected or if the existing file is "read only".

5. Reading the Tape. After **DEPOT** opens the disk file for writing, you must specify the method of reading the tape. Your choices are:

Select option for data read size.

- 1 Read until next filemark is encountered
 - 2 Read n blocks
 - 3 Read n bytes
- Your choice?

The following table describes these options:

- | | |
|---|---|
| 1 | Reading and data transfer begin at the current tape position and continue until DEPOT finds a filemark. |
| 2 | Reading and data transfer begin at the current tape position and continue until DEPOT reads the set number of data blocks. |
| 3 | Reading and data transfer begin at the current tape position and continue until DEPOT reads the specified number of bytes. |

If you select option 2 or 3, **DEPOT** prompts for the number of data blocks or bytes to read.

<p>Warning!! DEPOT has no way to know whether the number of blocks to read is reasonable. It is possible to read blocks beyond the end of the present file.</p>

Multiple Volume Tape Writes

If your tape write requires more than one tape to successfully backup all the data, **DEPOT** will write to the end of the tape, terminate the current write, write two filemarks, and prompt you for a new tape to be mounted. Once the new tape is mounted, **DEPOT** will continue the incomplete write operation.

Error Handling

DEPOT automatically retries tape errors 10 times. If the failure persists, you will be warned that an error has occurred. The message will tell you what happened and what you may be able to do about it, if it's not obvious. You usually chose how to proceed, but sometimes there is only one choice. The following table shows the possible choices when an error has occurred:

Error Messages

Key	Message	Meaning & Effect
S	Stop Trying	Discontinue present operation. Usually returns to the main menu.
T	Try Again	Another attempt is made on the failing operation.
C	Continue Beyond Error	DEPOT attempts to ignore the error. No additional retries are attempted on this error.

Some errors, such as disk failures, will prevent **DEPOT** from continuing. There is no guarantee that a retry will be successful. You can never retry errors that occur while writing to tape.

Limitations

DEPOT has the following limitations:

- The maximum block size is 65,280 bytes.
- **DEPOT** accepts block sizes less than one, but greater than zero.
- During retries on a damaged tape, **DEPOT** may lose track of the tape position.

DEPOT2

DEPOT2 (Data Exchange Program with Optional Translation—version 2) provides enhancements to the **DEPOT** utility. It offers increased speed and the convenience of command file execution. **DEPOT2** transfers data between system disk and magnetic tape, either in its binary form or with optional translation. The program assumes that data to be translated to EBCDIC is ASCII and data to be translated to ASCII is EBCDIC.

Features

DEPOT2 has the following features:

- Copies data from tape to disk
- Copies data from disk to tape
- Converts data from ASCII to EBCDIC (optional during tape writes)
- Converts data from EBCDIC to ASCII (optional during tape reads)
- Positions to an arbitrary tape location prior to read
- Permits user specified record sizes with a blocking factor
- Reads options from the command line
- Optionally reads commands from a file, permitting batch files for multiple tape operations.
- Has enhanced error handling of tape errors and disk errors.
- Has enhanced support for writing multivolume tapes.

Reading from Tape

The absence of option `/w` and `/a` specifies a read operation.

The `/r` and `/b` switches are normally only used with ASCII text files, as the automatic padding will make executable files useless.

Specify the destination file name, such as `/n outfile`, and the stopping criteria (such as reading until **DEPOT2** reads a filemark: `/m`). You use the same switch, `/n`, for both the source and destination file. The tape operation (read or write) dictates how **DEPOT2** uses the file. If the file already exists for a tape write operation, **DEPOT2** asks for permission before overwriting. If `/i` (ignore) is used, **DEPOT2** assumes permission to overwrite.

On read, you may also specify a deblocking factor, a record size, files or blocks to skip, and translation (EBCDIC to ASCII).

Record Size and Blocking Factor

Record size and blocking factor are intended for use with ASCII disk files where records are delimited by carriage returns and line feeds. While writing to tape, lines in the source file which are longer than record size will be truncated and lines which are shorter than record size will be padded with spaces; in either case, the CRLF characters are not written to tape. When reading a tape which was written with record size specified, that same record size should be specified again in order to restore the file to its original format (this assumes the written record size was large enough to prevent truncation of lines); the CRLF characters are inserted in the disk file.

Disk files which are in EBCDIC or binary may contain certain characters which are interpreted as control codes. This can cause these files to be modified when record size and blocking factor are specified. You may read EBCDIC or binary disk files to tape by specifying the blocksize (/s).

Use

When using **DEPOT2** to read a tape, remove the write enable ring for extra protection and load the tape onto the tape drive. For **DEPOT2** to write a tape, put a write enable ring on the reel and load the tape onto the drive.

Note: If you are writing to tape, make sure the tape you choose has nothing important written on it.

When the tape is loaded, place the drive online. Invoke **DEPOT2** from the command line, specifying any options, as follows

```
C:\> depot2 options
```

followed by **(ENTER)**, where *options* is one or more of the option switches specified below. Use option /c filename to tell **DEPOT2** to read filename instead of the command line for its program options. **DEPOT2's** options consist of one or more switches separated by spaces. A switch is / (slash) followed by a letter. If a switch needs an argument, put it after the letter with a space. (The space is optional if the argument begins with a number.) For example, the following option specifies that the tape block size is 1024 bytes:

```
    /s 1024
```

or

```
    /s1024
```


You may combine toggle options (ones not requiring arguments) with a single slash, as in:

`/tvM`

This specifies options translate, verbose, and stop reading at filemark detection.

The following table describes the **DEPOT2** read tape options.

DEPOT2 Read Options

Switch	Argument	Description
<code>/c</code>	filename	Reads filename for DEPOT2 commands. If it is present, DEPOT2 will ignore any other switches on the command line.
<code>/t</code>		Translate data from EBCDIC to ASCII on tape read.
<code>/n</code>	name	Use name for destination on tape read operations. name must include the extension
<code>/r</code>	val	Use record size val.
<code>/b</code>	val	Use blocking factor val. The default is 1.
<code>/v</code>		Verbose.
<code>/i</code>		Ignore errors (but print error messages).
<code>/m</code>	val	Skip val filemarks.
<code>/d</code>	val	Skip val data blocks.
<code>/B</code>	val	Read val data blocks.
<code>/R</code>	val	Read val records.
<code>/M</code>		Read until DEPOT2 detects a filemark.
<code>/p</code>		Set data bit 7 to 0.
<code>/s</code>	val	Use val block size.
<code>/k</code>		Ignore single filemarks, allowing files being read to be concatenated into a single DOS file. It must be asserted for each command line

Note: An important consideration when setting the `r` and `b` switches is that the `r` switch causes a CRLF (carriage-return line-feed) to be inserted after each record, which is useful when moving data into a database or spreadsheet.

Writing to Tape

Use option `/w` (write) or `/a` (append) to write to tape. Specify a block size (such as `/s 1024`) and a source file (such as `/n infile`) or you may specify a record size (`/r`) and a blocking factor (`/b`; the default is 1).

Use either `s` or `r` and `b`. Do not use `s` with `r` and `b`. Doing so is redundant (setting `r` and `b` defines the block size, `s`; setting `s` defines both the record size, `r`, and blocking factor, `b`). The `/r` and `/b` switches should normally only be used with ASCII text files, as the automatic padding makes executable files useless. You can also specify translation (`/t`) (ASCII to EBCDIC on tape write), the number of filemarks (`/m`) or data (`/d`) blocks to skip.

Record Size and Blocking Factor

Record size and blocking factor are intended for use with ASCII disk files in which records are delimited by carriage returns and line feeds. While writing to tape, lines in the source file which are longer than record size will be truncated and lines which are shorter than record size will be padded with spaces. When reading a tape which was written with record size specified, that same record size should be specified again in order to restore the file to its original format (this assumes the written record size was large enough to prevent truncation of lines). Disk files which are in EBCDIC or binary may contain certain characters which are interpreted as control codes. This can cause these files to be modified when record size and blocking factor are specified. You may write EBCDIC or binary disk files to tape by specifying the `/s` option.

Note: It is unnecessary—a waste of your time and effort, in fact—to use `/s` when you use `/r` and `/b`, or vice versa. Use either, as you choose.

The following table describes the **DEPOT2** write tape commands.

DEPOT2 Write Options

Switch	Argument	Description
/c	filename	Reads filename for DEPOT2 commands. If it is present, DEPOT2 will ignore any other switches on the command line.
/t		Translate data from ASCII to EBCDIC on tape write.
/n	name	Use name of disk file. name must include the extension
/r	val	Use record size val.
/b	val	Use blocking factor val. The default is 1.
/s	val	Use val block size.
/v		Verbose.
/i		Ignore errors (but print error messages).
/m	val	Skip val filemarks.
/d	val	Skip val data blocks.
/a		Append to tape.
/w		Write new tape.
/p		Set data bit 7 to 0.

Multiple File Operations

The /c ("filename") switch is very powerful, allowing you to create a **DEPOT2** command file, running a series of **DEPOT2** commands without supervision. An example of this, with more detailed explanation, follows, under Example Command File.

Note: It is unnecessary to use /s when you use /r and /b, or vice versa. Use either, as you choose.

Warning!!

Results may be unpredictable if you use both the /s and the /r and /b options. Do not use both

Options

This summary table describes all the **DEPOT2** command options.

DEPOT2 Read and Write Options

Switch	Argument	Description
/c	filename	Reads filename for DEPOT2 commands. If it is present, DEPOT2 will ignore any other switches on the command line.
/t		Translate data from EBCDIC to ASCII on tape read. Translate data from ASCII to EBCDIC on tape write.
/n	name	Use name of disk file on tape write operations. Use name for destination on tape read operations. name must include the extension
/r	val	Use record size val.
/b	val	Use blocking factor val. The default is 1.
/s	val	Use val block size.
/v		Verbose.
/i		Ignore errors (but print error messages).
/m	val	Skip val filemarks.
/d	val	Skip val data blocks.
/a		Append to tape.
/w		Write new tape.
/B	val	Read val data blocks.
/R	val	Read val records.
/M		Read until DEPOT2 detects a filemark.
/p		Set data bit 7 to 0.
/k		Ignore single filemarks, allowing files being read to be concatenated into a single DOS file. It must be asserted for each command line.
/W		Suppresses the 'wait for user approval to exit keystroke' (normally the program waits for a user keystroke before exiting, to allow you to see what has been done; this switch tells it not to wait).

Exit Codes

When **DEPOT2** finishes and returns to DOS, it returns a status byte which a batch file can examine to determine if an error has occurred. A value of zero indicates no error. Each bit indicates a type of error. More than one bit may be on at a time. If an error is detected and a retry is successful, then no error codes bit will be set. This table shows the meaning of the exit codes:

DEPOT2 Exit Codes

Decimal	Hex	Error Indication
128	80	Tape write
64	40	Tape read
32	20	Tape position (skipped too many files or blocks, etc.)
16	10	Disk write (path not found, write to "read only", etc.)
8	8	Disk read (file or path not found, etc.)
4	4	Disk other than read or write
2	2	No command switches, no controller, lacks memory
1	1	Miscellaneous (block too big, file not closed, etc.)
0	0	No Error

Multiple Volume Tape Writes

If your tape write requires more than one tape to successfully backup all the data, **DEPOT** will write to the end of the tape, terminate the current write, write two filemarks, and prompt you for a new tape to be mounted. Once the new tape is mounted, **DEPOT** will continue the incomplete write operation.

Error Handling

If an error occurs during reading or writing of a tape, **DEPOT2** automatically retries the operation 10 times. If the failure persists, you will be warned that an error has occurred. The message will tell you what happened and what you may be able to do about it, if it's not too obvious. You chose how to proceed, but sometimes there is only one choice. If you chose not to continue, **DEPOT2** will stop running the present command line. If the /i (ignore) switch is used, **DEPOT2** will continue with the next line of the command file (if any). The /i switch causes most errors to be ignored, but you will always be warned of any serious error. Some errors, such as disk failures, will prevent **DEPOT2** from continuing. There is no guarantee that a retry will be successful. You can never do additional retries on errors that occur during writing to tape. If you see the write error message, the automatic retries have already happened.

This table shows the possible choices when an error has occurred:

Error Messages

Key	Message	Meaning & Effect
S	Stop Trying	Discontinue present operation. Usually returns to the main menu.
T	Try Again	Another attempt is made on the failing operation.
C	Continue Beyond Error	DEPOT2 attempts to ignore the error. No additional retries are attempted on this error.

If the physical end of filemarker is detected while reading or positioning tape, you will be asked whether to continue the operation. This warning will normally appear only once (in command file operation, the warning will appear for all command lines which do not use the /i switch).

If the physical end of filemarker is detected while writing or appending to a tape, **DEPOT2** will finish writing all data in the cache, write two filemarks, and unload the tape. You will be warned that EOT was reached and asked to load a new tape to continue writing the unfinished file. If you select verbose (/v), you will be told the name of the file which spans two tapes.

Command Line Examples

Example 1:

This command sets the verbose option and writes the contents of disk file `input.txt` to tape, translating from ASCII to EBCDIC and using a 1024 byte block size. **DEPOT2** will ignore any errors.

```
C:\> depot2 /vwti /n input.txt /s 1024
```

Example 2:

To read this file, we instruct **DEPOT2** to read the first file (stop reading at first filemark), translating it back to ASCII. **DEPOT2** writes the data to file `out.txt`. Here is the command:

```
C:\> depot2 /tM /n out.txt
```

You do not have to specify block size (it reads a block at a time from tape), or an explicit read operation (the absence of /w and /a implies read).

Example 3:

The following command appends a file to tape, writing 80 byte records, with a blocking factor of 100 (that is, 100 records per block). It writes the contents of disk file `cardfile`:

Chapter 4—9-Track Tape Applications

```
C:\> depot2 /av /r80 /b100 /n cardfile
```

Example 4:

To read and write binary or executable files, do not specify translation. For example, the following command copies disk file `output.bin` to tape, using a 2048-byte block size:

```
C:\> depot2 /wv /n output.bin /s 2048
```

Similarly, to read a file from tape (again, without translation), use a command such as:

```
C:\> depot2 /vM /n input.bin
```

This creates disk file `input.bin`. Again, we use option `/M` to stop reading at a filemark.

Example Command File

We include a sample command file, `CMD`, on the distribution disk.

Important Note: This example shows how you can set **DEPOT2** to perform multiple and/or repetitive tasks. Note that each **DEPOT2** task requires a separate line in the command file you create, although the operations that make up a task can (and should) be kept together on one line.

Here are the contents of the sample file:

```
; This is a sample command file that illustrates
; how multiple tape operations may be specified
; for repetitive tasks. Please note that comment
; lines MUST begin with a semi-colon
; in the FIRST column of text.

; Write read.me file with fixed length records,
; 50 per block. The "w" switch indicates writing
; from the beginning of the tape.
/wv /r78 /b50 /n read.me

; Write it again as a straight image.
; The remaining write operations are specified
; with an "a" switch set to append these files
; to the existing data on tape.
/av /s 1000 /n read.me

; Write it this time with short records,
; truncating lines.
/av /r40 /b50 /n read.me

; Write it this time with long records, and
; larger block factor.
/av /r120 /b100 /n read.me
```

Chapter 4—9-Track Tape Applications

```
; Write it again with short records, and smaller
; block factor.
/av /r40 /b30 /n read.me

; Write one last time in EBCDIC, with 80 byte
; records and short blocks
/avt /r 80 /b 10 /n read.me

; Read the first file, deblock it and write it
; back to disk.
; Stop when a filemark is reached.
/vM /r78 /b50 /n test

; Read and reproduce the straight image.
/vM /n test1 /m 1

; Read file written with truncated lines, but
; only the first thirty lines.
/v /r 40 /b 30 /m 4 /n test2 /R 30

; Read the first file, deblock it and write it
; back to disk.
; This time stop after reading only one data
; block.
/v /r78 /b50 /n test3 /B 1

; Read the EBCDIC version. Note that if you use
; the DOS "TYPE" command to examine this file, it
; will appear to be double-spaced, due to
; the auto-wrap function of the display. Viewing
; the file with a text editor will eliminate
; this illusion.
/vtM /r80 /b10 /m5 /n test4
```

DOS Batch File Examples

Example 5:

This batch file will show a warning if there is ANY error:

```
echo off
depot2 /c cmdlist
if errorlevel 1 echo ** DEPOT2 RETURNED AN ERROR **
rem If DEPOT2 returns a status byte equal or ...
rem greater than 1 the "echo" message will be ...
rem printed on the screen
```

Example 6:

This batch file will warn only if there is a tape error:

Chapter 4—9-Track Tape Applications

```
echo off
depot2 /c cmdlist
if errorlevel 32 echo ** DEPOT2 RETURNED A TAPE ERROR **
rem If DEPOT2 returns a status byte equal or greater than ...
rem 32 the "echo" message will be printed on the screen
```

Limitations

DEPOT2 is limited to a 65,280 byte block size.

DEPOT2 accepts block sizes less than one, but greater than zero.

During retries on a damaged tape, **DEPOT2** may lose track of the tape position.

The /r and /b switches should normally only be used with ASCII text data files, as the automatic padding will make executable files useless.

Bugs

If the /n (name) switch is used without being followed by a legal DOS filename, strange filenames may result.

In command file operation, if the last line of the command file happens to exactly fill a tape, **DEPOT2** will prompt for an additional tape to be loaded, then exit normally.

DEPOT3

DEPOT3 (Data Exchange Program with Optional Translation, Version 3) is a flexible, programmable utility for reading 9-track tapes. At present, tapes may only be read using **DEPOT3**.

Use

To use **DEPOT3** you must prepare a script file. The script file contains keywords that are programming commands for **DEPOT3**. After creating the script file, execute by typing:

```
C:\> depot3 scriptfile (ENTER)
```

at the DOS command prompt. In this case, we placed our script in the file scriptfile

Creating Script Files

Use your favorite text editor to create a **DEPOT3** script file. A script file contains a series of lines with a keyword on each line. Keywords are always lower case. Some keywords require parameters (or arguments). The arguments follow the keyword and are separated by spaces. For example, to write a field defined by bytes 16 through 31 inclusive from each record, use the writedata command followed by the starting and ending byte number, as follows:

```
writedata 16 31
```

Note that the first byte in a record is byte 1 (not byte 0). Never use a negative byte position, as this will cause erratic behavior.

To separate this field with a carriage return line feed, use the same command, followed by a command, as follows:

```
writedata 16 31  
crlf
```

Follow these guidelines when creating your script:

- Place each command on its own line.
- In addition, all commands must be contained on a single line.
- If the command requires parameters, they follow the keyword. Separate the parameters with tabs or spaces.
- Number parameters (those that define a byte position, for example) must increase (or remain the same).
- Use a semicolon in column 1 to place comments in your script.

When creating a script, you must include these commands:

Chapter 4—9-Track Tape Applications

<code>outfile</code>	Specifies the output file (on disk). The file must not already exist.
<code>recordsize</code>	The tape block's record size. DEPOT3 makes sure that any field parameters are within the defined record size. The <code>recordsize</code> parameter must match what is actually on the tape.
<code>begdata</code>	Every command before this is performed once before processing the tape's data. Every command after <code>begdata</code> is performed once per record.
<code>enddata</code>	This marks the end of per record processing. You may place <code>echo</code> commands or a <code>chain</code> command after <code>enddata</code> .

The complete list of **DEPOT3** commands is shown at the end of the **DEPOT3** section. A particular script file will only read from one file on tape and write to one disk file. Operations on additional files will require additional script files. To avoid unnecessary rewinding, use the `chain` command, rather than running several times with different script files.

DEPOT3 automatically stops when it encounters a filemark.

Example Scripts

Example 1:

The following script creates output file `.v test` by reading a tape containing records of 130 bytes. Before **DEPOT3** reads any data from tape, it writes the lines:

Sample Script File Output

Data from tape 2044:

to the output file.

Here is the script:

```
; comments can go after semi-colons
outfile test
recordsize 130
echo "Sample Script File Output"
crLf
echo "Data from tape "
echo "2044:"
crLf
begdata
; write last name first, followed by first initial, as follows
; "lastname","i"
echo "/"
writedata 16 31
echo /","/
```

Chapter 4—9-Track Tape Applications

```
writedata 1 1
echo "/"
crlf
enddata
```

The `echo` command writes its argument to the output file. You must enclose the `echo` argument in double quotation marks (or another non-alphanumeric character). The `crlf` command writes a carriage return line feed sequence to the output file. `begdata` marks the beginning of data processing.

The script then writes the `lastname` field (bytes 16 through 31) enclosed in double quotation marks, a comma, and the first initial field (byte 1) enclosed in double quotation marks. We terminate each record with a carriage return line feed sequence (`crlf`). `enddata` signals the end of data processing.

Here is a sample of file test after executing the above script:

Sample Script File Output

Data from tape 2044:

```
"Kimelman      ,"D"
"Wittels       ,"I"
"Veyhl         ,"E"
"Harrison      ,"L"
"Schechter     ,"D"
"Sandstorm     ,"P"
"Short         ,"K"
"Rasp          ,"R"
"Lange         ,"D"
"Oswald        ,"R"
```

Example 2:

This script file shows the flexibility of the `writedata` command combined with the `crlf` command. The tape that we want to read is the same format as the previous one, but we format the data differently. Here, we place labels for the initial and last name in the output file, and separate each record with two carriage return line feed characters. Here is the script:

```
; comments can go after semi-colons
outfile test1
recordsize 130
echo "Names from Tape 2044"
crlf
crlf
begdata
; write labels indicating first initial and lastname:
; First initial:
; Lastname      :
```

Chapter 4—9-Track Tape Applications

```
echo "First initial:"
writedata 1 1
crlf
echo "Lastname      :"
writedata 16 31
crlf
crlf
enddata
```

Here is a sample of the file created from the above script:

```
Names from Tape 2044
First initial:D
Lastname      :Kimelman
```

```
First initial:I
Lastname      :Wittels
```

```
First initial:E
Lastname      :Veyhl
```

```
First initial:L
Lastname      :Harrison
```

Example 3:

It is frequently convenient to examine the label information on IBM labeled tapes. IBM labels are the first 3 blocks on a tape. Each label block is 80 bytes long.

The first block (labeled VOL1) contains volume information, the second contains Header 1 information (HDR1), and the third contains Header 2 information (HDR2). Because each block has fields defined differently, we must use the **DEPOT3** conditional (the `if` command) to process each block correctly. Also, we use the `if` command to look at the values of various fields (defined by the byte position) and print information based on these values.

Here is a script that will read such a tape and display the volume label information in the output file:

```
outfile labellook
recordsize 80
blktread 3
echo "IBM Tape Label Contents"
crlf
crlf
begdata
  toascii 1 80
```

Chapter 4—9-Track Tape Applications

```
if 1 4 .eq. "VOL1"
  echo "Volume ID: "
  writedata 1 4
  crlf echo "Serial ID: "
  writedata 5 10
  crlf
  echo "Owner ID: "
  writedata 42 51
  crlf
endif
if 1 4 .eq. "HDR1"
  echo "Data Set Identifier: "
  writedata 5 21
  crlf
  echo "Creation Date: (year - day of year)"
  crlf
  echo "
  "
  writedata 43 44
  echo " - "
  writedata 45 47
  crlf
  echo "Expiration Date: (year - day of year)"
  crlf
  echo "
  "
  writedata 49 50
  echo " - "
  writedata 51 53
  crlf
endif
if 1 4 .eq. "HDR2"
  if 5 5 .eq. "F"
    echo "Record format is fixed length."
    crlf
    echo "Block length: "
    writedata 6 10
    crlf
    echo "Logical record length: "
    writedata 11 15
    crlf
  endif
if 5 5 .eq. "V"
  echo "Record format is variable length."
  crlf
  echo "Maximum block length: "
  writedata 6 10
```

Chapter 4—9-Track Tape Applications

```
    crlf
    echo "Maximum logical record length: "
    writedata 11 15
    crlf
endif
if 5 5 .eq. "U"
    echo "Record format is undefined length."
    crlf
    echo "Maximum block length: "
    writedata 6 10
    crlf
endif
if 37 37 .eq. "A"
    echo "Data contains ASCII control
    characters." crlf
endif
if 37 37 .eq. "M"
    echo "Data contains machine control characters."
    crlf
endif
if 39 39 .eq. "B"
    echo "Blocked records."
    crlf
endif
if 39 39 .eq. "S"
    echo "Spanned records."
    crlf
endif
if 39 39 .eq. "R"
    echo "Blocked and Spanned records."
    crlf
endif
endif
enddata
crlf
echo "End of tape label data."
```

This script uses a few new commands. `blktread 3` indicates that should stop processing after the third block. `toascii 1 80` specifies that **DEPOT3** should translate byte positions 1 through 80 (the entire record) from EBCDIC to ASCII. The rest of the script uses the `if` command to compare the contents of the indicated byte positions in the record with the string in double quotation marks. This conditional allows you to execute different **DEPOT3** commands depending on the value of various

data fields. In particular, we examine the first four bytes of each record to identify the VOL1 record, the HDR1 record, and the HDR2 record.

The format for an `if` command is:

```

if start stop operator string
    command(s)
endif
    
```

where `start` is the starting byte position, `stop` is the stopping byte position, `operator` is the compare operator, and `string` is the delimited compare string. If the compare operation is true, **DEPOT3** executes all commands following the statement until it encounters a pairing statement. You may nest `if` statements as long as each `if` is paired with an `endif` statement.

DEPOT3 allows the following compare operators:

Operator	Meaning
<code>.eq.</code>	True if data field is the same as the string.
<code>.ne.</code>	True if data field is different than the string.
<code>.gt.</code>	True if data field is greater than the string.
<code>.lt.</code>	True if data field is less than the string.

The rest of the script defines certain byte positions to write to the output file with identifying labels.

Here is a sample of the output:

```

IBM Tape Label Contents
Volume ID: VOL1
Serial ID: BLONG
Owner ID:
Data Set Identifier: BOBLONG
Creation Date: (year - day of year)
                86 - 141
Expiration Date: (year - day of year)
                86 - 142
Record format is fixed length.
Block length: 00800
Logical record length: 00080
End of tape label data.
    
```

Example 4:

This script demonstrates **DEPOT3**'s flexibility. Here we request **DEPOT3** to read the label on an IBM label tape, and based on the contents of the label, write a second **DEPOT3** script to

process the tape's data. The last command we use in the script is the chain command, which executes the statements in our newly created script file.

This script assumes that the data is written in EBCDIC, and that it is fixed record format. The record size and file name are read from the label.

Here is the first script file:

```
outfile lblscript
recordsize 80
blktread 3
echo "; script created by lablread"
crlf
begdata
  toascii 1 80
  if 1 4 .eq. "HDR1"
    echo "outfile "
    writedata 5 21
    crlf
    echo "fileskip 1"
    crlf
  endif
  if 1 4 .eq. "HDR2"
    if 5 5 .eq. "F"
      echo "recordsize "
      writedata 11 15
      crlf
    endif
    if 5 5 .eq. "V"
      echo "Abort."
      crlf
    endif
    echo "begdata"
    crlf
    echo "toascii 1"
    writedata 11 15
    crlf
    echo "writedata 1"
    writedata 11 15
    crlf
    echo "crlf"
    crlf
  endif
enddata
```

Chapter 4—9-Track Tape Applications

```
echo "enddata"  
crlf  
chain lblscript
```

Here is the script produced by executing the above program:

```
;script created by lablread  
outfile BOBLONG  
fileskip 1  
recordsize 00080  
begdata  
toascii 1 00080  
writedata 1 00080  
crlf  
enddata
```

This script sets the output file to skip the first file (that is, the label) on the tape, sets the record size to 80, translates the entire record to ASCII, and writes it (byte positions 1 through 80) followed by a carriage return line feed sequence. Note that the arguments for `outfile`, `recordsize`, `toascii`, and `writedata` and were all generated dynamically from data on the tape. The script will stop (automatically) at a filemark.

Example 5.

The following script uses **DEPOT3**'s translation features to translate an ASCII **(CONTROL)+Z** (hexadecimal 1a) to a space (hexadecimal 20). You can define custom translation tables by defining a new translation table (the `transdef` command), filling the table with one or more pairs of hexadecimal numbers (using the `transfill` command), and translating specific fields within a record (using the `translate` command). When specifying pairs with the `transfill` command, the first hexadecimal number is the source character value; the second hexadecimal number is the target character value.

DEPOT3's translation mechanism also allows you to modify the internal ASCII to EBCDIC table (Table 1) and the EBCDIC to ASCII table (Table 2) by specifying new hexadecimal pairs for Table 1 or Table 2.

Here's the translation table script:

```
; script to show the translation mechanism  
outfile nozzzs  
recordsize 132  
; declare table 3  
transdef 3
```

Chapter 4—9-Track Tape Applications

```

; fill it
transfill 3 1a 20
begdata
  toascii 1 132
  translate 3 1 132
  writedata 1 132
  crlf
enddata

```

We declare table 3 (`transdef 3`), fill it with a single hexadecimal pair and use (`transfill 3 1a 20`) this table to translate the entire record (`translate 3 1 132`).

Command Reference

<code>begdata</code>	Start processing each record.
<code>blkread num</code>	Stop reading data after <code>num</code> blocks.
<code>blockskip num</code>	Skip <code>num</code> blocks before reading data.
<code>chain scriptfile</code>	Read file <code>scriptfile</code> for more DEPOT3 commands. If used, this command must be the last command in a script file or before the <code>begdata</code> command (further commands will not process). All DEPOT3 options are reset (you must respecify <code>outfile</code> , <code>recordsize</code> , <code>begdata</code> and <code>enddata</code> and any other DEPOT3 options). The new script file may also have commands to skip files and blocks. DEPOT3 will rewind the tape after the last <code>scriptfile</code> is processed.
<code>crlf</code>	Write a carriage return and line feed to the output file.
<code>echo string</code>	Write <code>string</code> to the output file. <code>string</code> must be delimited (by " or some other non-alphanumeric character, i.e. "string", 'string', etc.). You may include spaces in <code>string</code> ; it is currently limited to 150 characters.
<code>enddata</code>	Marks end of data processing in the script.
<code>endif</code>	Marks the end of an if block.
<code>endonfile</code>	Stop processing on a filemark (this happens automatically).
<code>fileskip num</code>	Skip <code>num</code> files on the input tape.
<code>if start stop op string</code>	Examine the field defined by byte positions <code>start</code> through <code>stop</code> and perform the comparison (defined by <code>op</code>) with <code>string</code> . If the conditional is true, all commands will be executed until the pairing <code>endif</code> command is found. The <code>if</code> command may be nested.
<code>lf</code>	Write a line feed to the output file.

Chapter 4—9-Track Tape Applications

outfile filename	Write data to file filename. The file must not already exist.
recordsize num	The record size is num bytes. You must define the record size in each script.
rectread num	Stop after reading num records.
strip7	Strip the seventh bit (high order) from each byte.
toascii start stop	Translate bytes start through stop from ebcdic to ascii.
toebcdic start stop	Translate bytes start through stop from ascii to ebcdic.
trace	Turn on trace facility. DEPOT3 writes each command to screen as it executes it. This is useful for debugging DEPOT3 scripts.
transdef tablenum	Declare a translation table tablenum, which must be between 3 and 10 inclusive.
transfill tablenum H1 H2 ...	Define the values of the (previously declared) translation table tablenum. The values H1 and H2 are listed as pairs of hexadecimal digits (separated by a space), where the first digit corresponds to the source value and the second digit corresponds to the new (target) hexadecimal value. You may list as many pairs as fit on a single line. You may use as many transfill commands as necessary to completely specify the table. Tables 1 and 2 are predefined ASCII to EBCDIC (Table 1) and EBCDIC to ASCII (Table 2). These tables may be modified through the transfill command by listing the source hexadecimal value followed by the new (target) hexadecimal value.
translate tablenum start stop	Translate bytes start through stop from each record by using the translation table tablenum.
writedata start stop	Write bytes start through stop from each record to the output file.

Error Handling

DEPOT3 automatically retries tape errors 10 times. If the failure persists, you will be warned that an error has occurred. The message will tell you what happened and what you may be able to do about it, if it's not too obvious. You usually chose how to proceed, but sometimes there is only one choice. There is no guarantee that a retry will be successful, especially on a damaged tape. The following table shows the possible choices when an error has occurred:

Error Messages

Key	Message	Meaning & Effect
S	Stop Trying	Discontinue present operation. Usually returns to the main menu.
T	Try Again	Another attempt is made on the failing operation.
C	Continue Beyond Error	DEPOT3 attempts to ignore the error. No additional retries are attempted on this error.

Disk errors will be explained in easy to understand terms and you will usually be able to continue after correcting the problem. After the disk error message appears, **(ENTER)** will cause DOS to retry the disk operation (when that is possible). **(ESC)** will cause the program to exit, if you wish.

If **DEPOT3** detects an error in a script file, it will display an error message with a number, then stop execution. The error numbers are described by the following table:

DEPOT3 Script Error Codes	
Error Number	Description
1	Field parameters exceed record size
2	Field specifiers out of order
3	Failed to recognize keyword
4	Ran out of memory during memory allocation
5	Parameter was expected but not supplied
6	Record size was not set before begdata keyword
7	Attempted to chain during field definition
8	if conditional was not recognized
9	String constant exceeded
10	Conditional found outside data definition
11	Conditional must begin with field definition
12	More endif statements than if statements
13	if / endif imbalance exists at end of data definition
14	Internal error at enddata: Call technical support
15	BCD result larger than input specification (future implementation)
16	Translation table number out of range
17	Table number has been previously defined
18	Transfill argument imbalance (must include pairs of hexadecimal numbers)
19	Attempting to use an undefined table number
20	Start byte position is less than 1

There are other situations which **DEPOT3** will warn you about. These warnings do not wait for a response and data transfer will continue as usual, whenever possible. The warnings include:

- 2 filemarks were found while skipping the requested number of filemarks (**DEPOT3** will not read past logical EOT).
- Two consecutive filemarks were found while skipping the requested number of blocks within a file.
- You requested reading more records than found in the file. This can be caused by using an incorrect recordsize parameter. It can also happen if the actual file (or block) size is not a multiple of the recordsize parameter.
- **DEPOT3** found no records at the requested position (there may be a file or block there which is shorter than recordsize).

FDUMP

Use **FDUMP** to examine the contents of a tape. **FDUMP** makes no assumptions about the format or block size of the tape's data. **FDUMP** is especially suited to:

- Determine the format of a tape
- Quickly sample a tape's data visually
- Troubleshoot a tape problem
- Debug a custom tape utility

Features

FDUMP is a screen-oriented, interactive tape reading utility. **FDUMP** allows you to select various functions, including reading the next block, positioning to a specific file, reading a specific block within the file, displaying a block of data in hexadecimal format (and EBCDIC or ASCII), scrolling or paging through the block, and jumping to a specific byte in a block. **FDUMP** constantly displays file number, block number, tape drive status word, block size, and whether the data is displayed in ASCII or EBCDIC. Finally, **FDUMP** displays the meaning of the status word by highlighting the description of each bit that is on or enabled.

Use

Choose a tape with data on it and load the tape onto the drive. **FDUMP** does not write to tape; it only reads data from the tape. Boot DOS and type: `c :> fdump (ENTER)`

```

FDUMP 2.3 9/2/88                               Controller: TX-8
-----
                >>> BEGINNING OF TAPE <<<

                hit F3 to read a block

-----
<ESC> to EXIT      F1 status help      F3 Next block      F5 Jmp to file      UP,DOWN Scroll
F2 Jmp to byte     F4 Ascii-EbcDic     F6 Jmp to block   PgUp PgDn Page
  
```

(Your **FDUMP** version number, date, or Controller model may differ.) At this point you may instruct to read the first block by pressing function key **(F3)**. Alternatively, position the tape head at another file before reading (press **(F5)**) or read another block within the current file (press **(F6)**). After each function, **FDUMP** updates the status information indicating the file number, block number, tape drive status word, block size, and data mode (ASCII or EBCDIC). Press function key **(F1)** to view a description of the status word.

Once **FDUMP** reads and displays a data block, you may view its contents by scrolling (press **(↑)** or **(↓)**) or paging (press **(PGUP)** or **(PGDN)**) through the screen display, or by displaying a specific byte address at the first line of the display (press **(F2)**). **FDUMP** displays the data in hexadecimal notation as well as either ASCII or EBCDIC. Press function key **(F4)** to toggle between ASCII and EBCDIC display modes. Press function key **(F3)** to read and view successive blocks.

Exit **FDUMP** at any time by typing **(ESC)**.

FDUMP's function keys are summarized in the following table:

FDUMP Function Keys

Key	Command Description
(F1)	Display Status Help
(F2)	Jump to Byte
(F3)	Read Next Block
(F4)	ASCII—EBCDIC Toggle
(F5)	Jump to File Number
(F6)	Jump to Block Number
(↑)	Scroll Display Up
(↓)	Scroll Display Down
(PGUP)	Page Display Up
(PGDN)	Page Display Down
(ESC)	Exit to DOS

We describe the function keys in more detail below:

(F1): Status Help.

FDUMP constantly displays the drive's status word in hexadecimal notation. Pressing **(F1)** displays a description of the drive's status word. **FDUMP** highlights the description of each bit that is enabled, allowing you to easily interpret its value. For example, if the status word is 5015 (hex) then bits 0, 2, 4, 12

and 14 are enabled. highlights the descriptions of IFBY (bit 0), IRDY (bit 2), IONL (bit 4), DBYT (bit 12), and IDENT (bit 14).

Status Bits.

The tape drive's status word (16 bits) has the following meaning.

Bit 0 IFBY

The drive is busy executing or completing a command. This bit is enabled during and just after a command.

Bit 1 IDBY

The drive is in DMA phase of a command. This bit is enabled during direct memory access to and from the tape drive.

Bit 2 IRDY

This bit is true unless the drive is not selected, not loaded, offline, or rewinding.

Bit 3 IFPT

The tape is write-protected (the reel lacks a write-enable ring).

Bit 4 IONL

The drive is online.

Bit 5 ISPD

The last command was high speed (for dual-speed drives only).

Bit 6 IBOT

The tape head is at the Beginning of Tape (BOT) marker.

Bit 7 IEOT

The tape head is at the End of Tape (EOT) marker. There is a few feet of usable tape beyond EOT.

Bit 8 IHER

A hard error (uncorrected) occurred on read or write.

Bit 9 ICER

A hardware correctable error occurred on read.

Bit 10 IFMK

The drive detected a filemark on the last command.

Bit 11 IRWD

The drive is rewinding.

Bit 12 DBYT

The drive dropped the data busy signal indicating that a command completed. This bit is always zero on TXi-16 controllers.

Bit 13 NRZI

The drive returned NRZI (Non-Returned to Zero Inverted) status. NRZI is an encoding mechanism common in 800 bpi

(bytes per inch) drives. Bit 13 may also indicate the alternate density in a dual-density tape drive (for example, 6250 bpi or 3200 bpi on a 1600 bpi dual-density tape drive).

Bit 14 IDENT

Either the drive detected an ID (identification) burst for PE (Phase Encoded) tape or CCG (Check Character Gate) status for NRZI tape. PE is an encoding mechanism common in 1600 bpi drives. The ID burst occurs only after a read at the Beginning of Tape. CCG status occurs with every data block read on NRZI blocks.

Bit 15 PAR

The controller detected a parity error. This indicates bad cables, controller, or tape drive. This bit is always zero on TXi-16 controllers.

(F2): Jump to Byte.

Press **(F2)** to display the data from a specific block address at the top line of the screen. prompts for the address in hexadecimal notation as follows:

Enter byte position in HEX: 0b43 <Enter>

Here we specify address 0b43. **FDUMP** now displays the data beginning at address 0b40 at the top line of the screen. If you specify an address larger than the block size, **FDUMP** simply displays the last portion of the data block.

(F3): Next Block.

Press **(F3)** to read the next block of data. If the tape is at BOT, **FDUMP** reads the first data block. Pressing **(F3)** multiple times will read successive data blocks. After a read, **FDUMP** reports the current file number, block number, tape drive status word, block size, and display mode. **FDUMP** can read any block up to 64kB.

When **FDUMP** detects a filemark, it responds with this message:

```
>>> FILEMARK DETECTED <<<
```

```
hit (F3) to read a block
```

Now press **(F3)** to read the first block of the next file.

(F4): ASCII-EBCDIC.

Press **(F4)** to toggle the display mode between ASCII and EBCDIC. You can press **(F4)** any time during the **FDUMP** session, so you do not need to know ahead of time the character code mapping of your tape.

(F5): Jump to File.

The Jump to File function allows you to position the tape head at the beginning of an arbitrary tape file. **FDUMP** defines a file to include all data blocks between filemarks. The first file is file 0. To read an arbitrary file, press **(F5)**. **FDUMP** prompts:

Enter file position in decimal: 3 <Enter>

Here we specify the fourth file (file 3). **FDUMP** will position the tape at the beginning of this file and display this message:

>>> FILEMARK DETECTED <<<

hit **(F3)** to read a block

Press **(F3)** to read the first data block. If you specify the first file (file 0), **FDUMP** responds with:

>>> BEGINNING OF TAPE <<<

hit **(F3)** to read a block

If you specify a file number larger than the number of files on tape, **FDUMP** will attempt to find it and respond with:

>>> FILEMARK DETECTED <<<

>>> LOGICAL EOT DETECTED <<<

> PROCEED FORWARD WITH CAUTION <

indicating that the specified file does not exist. This occurs when **FDUMP** reads 2 consecutive filemarks.

Use the Jump to File function to move both forwards and backwards to particular tape files.

(F6): Jump to Block.

The Jump to Block function allows you to read any block within the current file (or position the tape head at the filemark separating it from the next file). **FDUMP** defines a block to include all data separated by IRGs. The blocks may be of any size (up to 64kB), and **FDUMP** will report the size of the most recently read block on the display. **FDUMP** counts blocks beginning at 1. For example, to read the third block of the current file, press **(F6)** and responds with the following prompt:

Enter file position in decimal: 3 **(ENTER)**

Here we respond with 3 for the third block. (Remember, blocks count from 1.) **FDUMP** now displays the data from block 3.

If you specify a block number larger than the number of blocks in the current file, **FDUMP** detects a filemark, positions the tape at the first block of the next file, and displays the following message:

```
>>> FILEMARK DETECTED <<<
```

hit **(F3)** to read a block

Press **(SHIFT)+(F3)** to read and display the first block of this (that is, the next) file.

If you specify a block number larger than the number of blocks in the current file and there are no more files on the tape, **FDUMP** detects the filemark as before. However, when you attempt to read the first block of this next file, **FDUMP** detects another filemark indicating the Logical End of Tape (EOT), and responds with:

```
>>> FILEMARK DETECTED <<<
```

```
>>> LOGICAL EOT DETECTED <<<
```

```
> PROCEED FORWARD WITH CAUTION <
```

Scroll Up, Scroll Down

Use **(↑)** and **(↓)** to scroll the display a line at a time.

Page Up, Page Down

Use **(PGUP)** and **(PGDN)** to page the display up or down.

Examples

To illustrate **FDUMP**'s use, we describe the following examples:

- Reading a tape containing an unknown format, character mapping, or density
- Checking the integrity of a tape (quick visual inspection)
- Troubleshooting a tape problem
- Debugging a custom tape utility program

Example 1:

Reading a tape containing unknown data.

Remove the write enable ring from the back of the tape reel and mount the tape on the drive. Boot DOS and call **FDUMP**:

```
C:>fdump (ENTER)
```

FDUMP immediately reads the first data block. Note **FDUMP**'s status line as well as the data on the screen. Answering the following questions may help you determine the format and character mapping of the tape's data.

1. Did **FDUMP** read good data?

Inspect the data, scrolling or paging as needed. Toggle the ASCII EBCDIC switch (**(F4)**). Press **(F1)** to examine the tape drive's status word.

If Bit 8 (IHER) is true, you may have a bad spot on the tape or the tape may be written in a density incompatible with your tape drive (in this case, **FDUMP** can't read it at all). Use **(F6)** to select another block. If this block also returns abnormal status, then the density is probably incompatible.

If Bit 9 (ICER) is true, the data is probably good (as long as status Bit 8 is false).

If Bit 15 (parity error) is set, the data is probably bad even if **FDUMP** returned data.

2. What's the format of my tape?

Although programs write tapes in the general format of multiple files separated by filemarks and blocks separated by tape gaps, the data in the blocks may have meaning about how the rest of the data is formatted. For example, the first file may be a directory file with names and file locations of the following files. Or the entire tape may contain only 1 file, with directory information stored in the initial block or blocks.

The files themselves may have intrinsic formats. Card images may contain null-padded or blank-padded 80 byte records.

Use the scrolling and paging facilities and the arbitrary block and file reads to examine the tape's data. After determining format, you can use **DEPOT** or **DEPOT2** to read and translate it.

Example 2: Checking the integrity of a tape.

Using **FDUMP** is a quick way to check the contents of a tape. Again, remove the write enable ring, mount the tape in the drive, boot DOS, and call program **FDUMP**. **FDUMP** immediately reads the first data block. If the data is in ASCII, you do not need to toggle the ASCII-EBCDIC switch. Page through the block, visually inspecting the data for correctness. Read subsequent blocks and files as needed.

This method is not meant to verify data correctness. It is simply a quick method for checking overall contents.

Example 3: Troubleshooting a tape problem.

If you use another utility to read a tape, but the data is not correct or the tape does not seem to be readable, **FDUMP** may help locate the problem. Load the tape and call **FDUMP** as before. **FDUMP** immediately reads the first data block. If the tape is readable, but the data is not correct, press **(F1)** to

examine the status word. If Bit 15 (parity error) is true, check cables between the controller and the drive. Bit or bits in error (in your data) may contain clues about which cable lines or channels are bad. The least significant bit corresponds to tape channel 7.

Mount another tape (remember to remove the write enable ring) to determine if the tape itself is faulty. If other tapes also have problems, run the **xTEST** program (using a scratch tape) to gather further diagnostic information.

Example 4: Debugging a custom tape utility program.

You can use **FDUMP** to help debug your own tape utility program. For example, when you're ready to test your program, mount your test tape and write your testing information to tape. Now call **FDUMP** to examine it (block by block if desired). **FDUMP** will tell you the length of each block, how many blocks in a file, and the positions of the filemarks.

Use **(F3)** to read successive blocks. Note their length and the block number from the beginning of file. Also note the file number. As **FDUMP** reads each block, you may page through its contents as necessary. At the end of the file, **FDUMP** clearly indicates that it read a filemark, and the file number and block number reflect this status. Continue through the tape, examining its contents until you're convinced your tape utility is operating correctly.

Limitations

FDUMP has the following limitation:

Maximum data block size is 65,000 bytes.

LABEL

LABEL creates IBM and ANSI Standard Labeled Tapes, supporting Level 1 of these standards. Level 1 includes:

Files: single-file single volume or single-file multivolume tapes.

Labels: VOL1, HDR1, HDR2, EOVI, EOVI2, EOF1, and EOF2.

ANSI does not require HDR2, EOVI2, and EOF2 for Level 1.

LABEL writes these optional labels. It is allowable to write additional labels, since a Level 1 system ignores and bypasses any labels it does not process. The IBM Standard requires these labels.

Record/Block Structures: Blocks consist of one or more fixed-length records. For non-character data files, we also support undefined record format.

LABEL supports multi-file volumes for both ANSI and IBM labels.

Use

Find a blank tape, insert a write enable ring, and load it onto the drive. This procedure will write to tape; make sure the tape you choose has nothing important written on it.

When the tape is loaded, place the drive online. Type:

C:\> label (ENTER)

LABEL displays the following main menu:

```
Tape Label Writing Program
Version 2.2 9/16/88 (c) 1987 Overland Data, Inc.
SERIAL NO 88287575

A) ANSI Label
I) IBM Label
Q) Quit

Choose One -->
```

Depending on whether you choose option A or I you will create an ANSI or IBM standard labeled tape.

IBM Standard Labeled Tape

An IBM Standard Labeled Tape consists of the following format: a header file (or label) followed by a filemark, the data file followed by a filemark, and a trailer file followed by two filemarks denoting the end of tape.

When the data file does not fit onto a single tape reel, a slightly different trailer file is required, followed by a single filemark. The second and subsequent tapes contain three block header files (followed by a filemark), the data file (followed by a filemark), and the trailer file. The final tape ends with two filemarks.

Volume Header Information

The Header File consists of three 80-byte tape blocks. The first header label, VOL1, is called Volume Header Information. **LABEL** displays the following menu:

```

                                Volume Header Information

A) Volume ID      [000001]
B) Owner ID      [      ]
C) Tape Reel Size [ 600]
D) Tape Density  [1600]

M) Next Screen
Q) Quit

Type letter of item you want to change,
M (next), or Q (quit) -->
```

Volume ID

(6 bytes.) A unique identification code assigned to your tape volume. The code is usually numeric characters (000001-999999), but may be any six alphanumeric characters. If you do not change this field, **LABEL** uses the default 000001.

Owner ID

(10 bytes.) This field indicates a specific customer, person, installation, department, etc., to which the volume belongs. Any code or name is acceptable. If you do not change this field, **LABEL** uses blanks.

Tape Reel Size

Specify the size of your tape reel. This field is not used in any header file. However, **LABEL** uses this information to keep track of the amount of tape available for writing data. Providing inaccurate reel size will cause an unexpected end of file on tape.

Tape Density

Specify the recording density of your tape drive. The standard labels support tape densities of 800 bpi (bytes per inch), 1600 bpi, and 6250 bpi. This information is also used to keep track of the amount of tape available for writing data. Providing inaccurate tape length data will cause an unexpected end of file on tape.

Header 1 Information

The second header block is HDR1, called Header 1 Information. displays the following menu:

```
Header 1 Information

A) Filename on Disk [          ]
B) Dataset Identifier [          ]
C) Creation Date [00000]
D) Expiration Date [00000]
E) Data Set Security [0]
F) System Code [          ]

                                N) Next Screen
                                Q) Quit

Type letter of item you want to change,
N (next), or Q (quit) -->
```

Filename on Disk

Type in the name of the DOS file you want to put on tape. **LABEL** tries to open the file for reading. If unsuccessful, it displays an error message. You must enter data for this field before proceeding to the next screen.

Dataset Identifier

(17 bytes) Specify the dataset name for the HDR1 label. This will be the file name of your data on the labeled tape. You must enter data for this field before proceeding to the next screen.

Creation Date

Specify the year and day of the year when the data set was created. IBM uses the format yyddd where yy is the 2-digit year and ddd is the 3-digit day of the year. **LABEL** uses the default 00000 if you do not change this field.

Expiration Date

Specify the year and day of the year when the data set may be scratched or overwritten. IBM uses the format yyddd where yy is the 2-digit year and ddd is the 3-digit day of the year. **LABEL** uses the default 00000 (meaning the dataset on tape is expired and may be overwritten) if you do not change this field. Use a date in the future to protect your tape data from destruction.

Data Set Security

(1 byte) Specify the code indicating the security status of the data set as follows:

- 0 No password protection.
- 1 Password protection. Additional identification of the data set is required before it can be read, written, or deleted.
- 3 Additional identification of the data set is required before it can be read, written, or deleted.

LABEL uses the default 0 (no protection) if you do not change this.

System Code

(13 bytes) Specify a unique code that identifies your system. This field is not used or verified when processing labeled tapes. **LABEL** uses all blanks if you do not change this field.

Header 2 Information.

The third header block is HDR2 called Header 2 Information. **LABEL** displays the following menu:

```
Header 2 Information

A) Block Length  [00000]
B) Record Length [00000]
C) Job Step ID   [          ]
D) Control Characters [ ]

N) Next Screen
Q) Quit

Type letter of item you want to change,
N (next), or Q (quit) -->
```

Specify the block length which your data file will be written in, in bytes. You must specify a field value before continuing. The block length (see next field) must be an even multiple of the record length.

Record Length

Specify the record length in bytes in which your data file will be written to tape. You must specify a value for this field before proceeding to the next screen. The block length must be an even multiple of the record length.

Note: Your disk file will be read a line at a time and the carriage return - linefeed delimiters removed. Also, each line will be padded with blanks to the record size you specify. If you do not want this type of processing, specify:

B) Remove CRLF Delimiters [NO]

in the next menu. In this case, you should make the record size the same as the block size.

Job Step ID

(17 bytes) Specify the job step identification. If you do not change this field, **LABEL** uses blanks.

Control Characters

If your file contains ASCII or Machine control characters, specify A (for ASCII) or M (for Machine). A blank means that your file contains no control characters.

Write IBM Labeled Tape.

Before actually writing the tape, you may change some processing options for **LABEL**. They are:

1. Translating the data to EBCDIC
2. Removing CRLF delimiters and padding to specified record size

LABEL displays the following menu:

```

                                Write IBM Labeled Tape

A) Translate DATA to EBCDIC  [YES]
B) Remove CRLF Delimiters    [YES]

W) Write Tape

  N) Next Screen
  Q) Quit

Type letter of item you want,
N (next), or Q (quit) -->
```

Translate Data to EBCDIC

If your file is character data, you will want to translate it to the EBCDIC character set. If your file is binary data and you do not want it translated to EBCDIC, disable the translation.

Notes:

This is separate from removing CRLF delimiters. To disable CRLF processing, disable option (remove CRLF delimiters).

This applies to data only. To write ASCII headers and trailers, you must use the ANSI Label Standard (see the Main Menu).

Choose item A to toggle between options YES and NO. The default for this field is YES (translation takes place).

Remove CRLF Delimiters

If enabled, **LABEL** reads your file a line at a time, removes the CRLF delimiters, and pads each line with blanks to the record size. If your file is binary data and you do not want this line-by-line processing, disable the Remove CRLF Delimiters option.

When disabled, **LABEL** reads your file in blocks and simply writes

it to tape with no processing. Also, when disabled **LABEL** ignores the record size specified and makes it the same as the block size. The record format is still fixed (F).

Note: This item is separate from data translation to EBCDIC. To disable translation to EBCDIC, disable option (Translate DATA to EBCDIC).

Choose item B to toggle between options YES and NO. The default for this field is YES (remove CRLF delimiters).

Write Tape

When finished entering data for the program, type **W** to write.

Note: We recommend you first check all options by reviewing each menu before writing tape. Press **N** to view each screen until **LABEL** returns to the Write IBM Labeled Tape menu.

LABEL reports the number of blocks (excluding headers and trailers) it wrote to tape. If the data requires more than one tape, **LABEL** writes the trailers to tape and prompts for a new tape.

End of File Processing

When **LABEL** finishes writing the current file to tape, it asks you if you want to write another file. **LABEL** will display this menu:

```

                                End of File Processing
                                Current Filename:  [MYFILE.DAT ]
                                Do you want to write another tape (Y/N)? [ ]

```

Press Y if you have another file to write to tape. **LABEL** then returns to the Header 1, Header 2, and Write Tape Menus to obtain new information. You need only change the parameters that are different since **LABEL** saves the values from the current tape data set. Press N if you want to stop.

ANSI Standard Labeled Tape

An ANSI Standard Labeled Tape is the same as an IBM standard labeled tape, except for the following items:

- Header information is recorded in ASCII.
- The Owner ID Field consists of 14 bytes instead of 10.
- No option is provided for translating the data to EBCDIC.

Chapter 4—9-Track Tape Applications

- End of volume processing requires two filemarks after the final EOV trailer; IBM end of volume requires only one filemark.
- HDR2, EOVS, and EOF2 labels are optional. **LABEL** writes these to tape.

Each of the ANSI menus is clearly labeled ANSI.

TAPEUTL

This section contains complete instructions for running all the programs included in **TAPEUTL**. Each program is covered separately. The manual follows the main menu order (unlabeled tape, labeled tape, etc.) with each submenu choice covered in sequence within each section (disk-to-tape, tape-to-disk, etc.).

If you want to use 9-track tape by writing a program in a high level language or in assembly language, Appendix A contains information on using the tape functions available through our device driver from programs written in C, BASIC, COBOL, FORTRAN or assembly language.

The DOS filter function allows you to translate the data stream during transfers between disk and tape according to statements in a user-created translation file.

Automatic-mode execution of **TAPEUTL** under control of a command file allows you to capture a **TAPEUTL** session and 'replay' the same sequence of commands later.

TAPEUTL

The main program in the Tape Utility package is **TAPEUTL.EXE**. To run this program, simply type its name at the DOS command prompt:

```
C:> tapeutl
```

When the program is loaded the first of a number of help screens will be displayed. At this point you can either start the program proper by hitting any alphanumeric key or you can page through the help screens by using the **(PGUP)** and **(PGDN)** keys. You may want to do this the first time you run **TAPEUTL** to familiarize yourself with the help facility.

The first thing **TAPEUTL** does before displaying the first screen is find the amount of system memory available to determine if there is enough space for its own buffers. If the available memory is insufficient **TAPEUTL** will display a message saying so and stop. If this happens you will at the minimum need to reconfigure the system software on your machine. If your system needs more memory you may have to install more physical memory to run **TAPEUTL**. Check the following:

- Do you have memory-resident programs installed? Such programs reduce the amount of memory available to other programs: reboot without them and retry **TAPEUTL**.

- Some print spoolers and RAM-disks such as VDISK also use system memory: remove the device's statement from your system's CONFIG.SYS file, reboot, then retry TAPEUTL. (NOTE: if the device uses extended memory on an auxiliary plug-in board then it will not interfere with TAPEUTL.)

When you start **TAPEUTL** the main menu screen is displayed with a list of submenus you can choose from (Figure 1). Note that all **TAPEUTL** screens (except the tape block display screen) follow the general format: current submenu stays on the top half of the screen while particular options for the operation currently being done appear on the bottom half. The tape display screen is described separately below.

To perform a tape function choose the submenu from the main menu (type its letter) then choose the desired function from the submenu. Many functions (e.g., file transfer to/from tape) will present fields on the lower half of the screen (e.g., filenames and record lengths) which you fill in; when all necessary fields have been filled in press **(ENTER)** to start the function. To move from field to field use **(TAB)** (the key with two arrows on it on IBM keyboards) to move forward and **(SHIFT)+(TAB)** to move backward. To cancel a function, **(ESC)** will always put you back at the menu you came from without performing the function (think of **(ESC)** as a user's panic button). If you are at the MAIN menu **(ESC)** will end the program and return you to DOS.

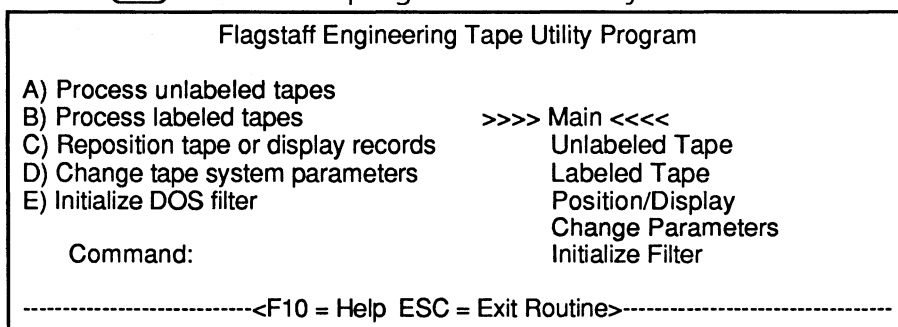


FIGURE 1. Main Menu

Processing Unlabeled Tapes

When the unlabeled tape option is chosen from the main menu the following submenu is displayed:

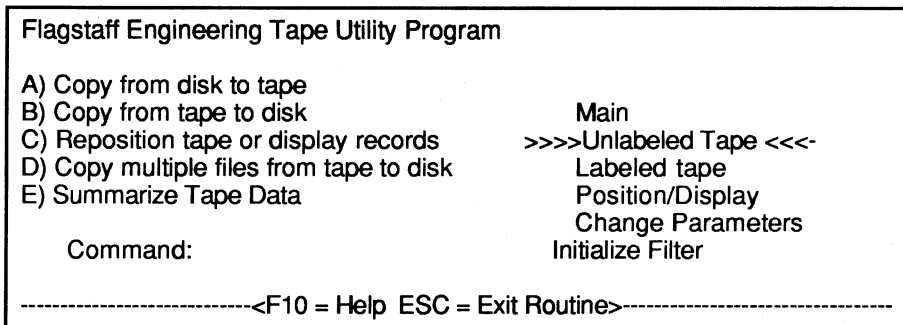


FIGURE 2. Unlabeled Tape Submenu

Although this option is primarily intended for handling unlabeled tapes it can be used to transfer data from labeled tapes if the user uses caution. This option should not be used to transfer data to labeled tape as no labels are written by the transfer routines.

The exception to this rule is if certain system specific labels need to be written on a tape; in which case the labels are written as separate files preceding and/or following the actual data file. In such a case we recommend that the user consult a system programmer familiar with the target system label format.

Copy from Disk to Tape

Use this option to copy data from a DOS (disk) file to unlabeled tape at the current tape position. Unlike the labeled tape disk-to-tape transfer this operation is always carried out at the current tape position. The tape must be correctly positioned before this operation is started; use the Reposition tape or display records option on this submenu if the tape is not where you want it.

Initially the following three or four fields will appear at the bottom half of the screen:

Input file name: this is the fully-qualified name of the DOS disk file. A wildcard DOS file name may be given for unlabeled tapes. If just a normal wildcard file name is given each input file will create a separate output file on tape. (On labeled tapes each tape file name will be the same as input DOS file name.) If a "+" is placed immediately after the wildcard name then all wildcard files will be concatenated into one file on the tape.

Translate data? Three choices are possible:

N - no translation

A - data will be translated from EBCDIC to ASCII

E - data will be translated from ASCII to EBCDIC.

Data type: this specifies how the incoming data is to be treated. Since data on tape is always written in blocks the entry in this field will control how the disk file data is broken up into these tape blocks. There are three (3) possibilities:

- 1 Fixed-length records: data is read from the disk file as records of a specified length (given by the user). The disk data is not scanned for line-end characters or other delimiters but is assumed to be divided into logical records of the user-specified length.
- 2 Variable-length records: the disk file data is assumed to be in logical records of varying length; each record terminated by an identical character sequence. With this type of transfer data is read from disk as variable-length but written to tape as fixed-length records (short records are padded).
- 3 String data: this option is used in the case where the disk data file is not divided into logical records at all (e.g., an executable program file that consists of binary data). The disk data is treated as a continuous string of data. The tape file is divided into blocks but the blocks are simply fixed-length chunks of the input file with a possible short last block.

Filter on (N/Y): this field will only appear if the translation filter has been initialized from the main menu. If Y(es) is specified, data from the input file is translated according to statements in the filter file before being written to tape. The default (predicted) value here is N(o). See the section on the DOS filter function starting on page 144 for more information.

Depending on the data type chosen for the transfer additional fields will appear. Complete screens for each data type are shown next with a description of the remaining fields.

Fixed-Length Data Transfer

```
-----<F10.= Help ESC.= Exit Routine>-----  
  
                                     <Disk0Tape>  
  
Input file name \customer\sales.dta  
Translate data? (N=no A=EBCDIC to ASCII E=ASCII to EBCDIC) N  
Data type: (F)ixed (V)ariable (S)tring f  
Record size 1024  
Records per output tape block 20  
Records to skip 0  
Records to transfer ALL  
  
File size 189952 bytes  
There is a partial record of 512 bytes: transfer it? y/n y
```

Figure 3. Disk to unlabeled tape-fixed-length transfer fields.

Four (4) additional fields appear for fixed-length transfer operations:

- 1 Record size: this specifies the length of each logical record in the disk file. Remember that **TAPEUTL** doesn't do any kind of checking on the input records but assumes that the input records are in fact this length. If you're not sure what length the file records are look at the file first with a file dump/display utility (e.g., the DFILE program) to determine the proper record size.
- 2 Records per output tape block: this is the blocking factors to be used on tape. This number multiplied by the record length gives the tape block size in bytes which can be a maximum of 65,536 bytes. If this size is exceeded the program will display a warning message and you will be asked to change either the record size or the number of records per block.
- 3 Records to skip: this field pre-filled with zero specifies an optional number of records from the input file to be skipped before the data transfer to tape starts.
- 4 Records to transfer: this field pre-filled with ALL specifies an optional limit on the number of records to be transferred to tape. If only a certain number of records are to be copied, enter the number in this field.

When a fixed-length transfer is started **TAPEUTL** gets the size of the input file from the disk directory displays it and compares this size to the record size to determine whether there will be a short record at the end of the file if there is a message will

appear at the bottom of the screen asking if you want to transfer this partial record. Answering Y will cause this last record to be padded with binary zeros (0) at the end of the file; answering N will simply drop this last record from the tape file. There will usually be a short block at the end of a fixed-length blocked record file (when writing out fixed length records and if file size is not a multiple of the record size the operator is still asked if he/she wants to truncate the partial record).

After the transfer is complete the number of tape blocks, number of logical records and number of bytes in the tape file will be displayed. A prompt will appear at the bottom of the screen asking if you want to append another file to this one; see *Appending Files on Tape*, below, for an explanation of this choice. To transfer another file simply hit

(ENTER) at this point. This will close the tape file by writing filemarks after the last block and prepare you for writing another new file after this one. Press **(ESC)** to return to the unlabeled tape submenu.

Variable-Length Data Transfer

```

-----<F10.= Help ESC.= Exit Routine>-----
                                                    <DiskTape>

Input file name \text\letter1.doc
Translate data? (N=no A=EBCDIC to ASCII E=ASCII to EBCDIC) N
Data type: (F)ixed (V)ariable (S)tring v
Record size 60
Records per output tape block 1000
Records to skip 0
Records to transfer ALL
Line delimiter 0DOA
Pad character 20

File size 48182 bytes
Data transfer in progress...
    
```

Figure 4. Disk to unlabeled tape-variable-length transfer fields.

Six additional fields appear when a variable-length transfer is chosen:

1. Record size: this is the size of the tape (output) record which is the size of the largest record in the file not including line delimiter characters. Remember that variable-length only applies to the disk (input) file; the tape file is written as

fixed-length records. If a record is shorter than the given record size it is right-filled with pad characters (see below). If a record is bigger than this size it will be split—the excess part will be written as a new record—and a warning message will be displayed after the transfer:

WARNING! nn records exceeded specified length
If this happens try the transfer again with a larger record size. No data will be lost in any case but the file may need to be reformatted to restore lines to their original form.

2. Records per output tape block: this is the blocking factor to be used on tape. This value multiplied by the record length gives the tape block size in bytes which can be a maximum of 65,536 bytes. If this size is exceeded the program will display a warning message and you will be asked to change either the record size or the number of records per block.
3. Records to skip: this field pre-filled with zero specifies an optional number of records from the input file to be skipped before the data transfer to tape starts.
4. Records to transfer: this field pre-filled with ALL specifies an optional limit on the number of records to be transferred to tape. If only a certain number of records are to be copied enter the number in this field.
5. Line delimiter: this specifies which character or string of characters are to be taken as an end-of-line marker in the input file. This entry must correspond to the actual line delimiters in the file for the transfer to be successful. The characters (up to 10) are specified as 2-digit hexadecimal values with the most common values pre-filled: 0D 0A or ASCII carriage-return and line-feed characters.
CAUTION! See the section on the DOS filter function starting on page 144 for an explanation of pitfalls to avoid if you are using a translation filter.
5. Pad character: the character which will be used to fill short records is specified here as a 2-digit hexadecimal value. This field is pre-filled with a commonly-used fill character the ASCII space (hex 20). The EBCDIC space character is hex 40.

The variable-length data transfer works like this: as the input file is read from disk TAPEUTL continually scans the data for the specified line delimiter characters. When the delimiter string is found the record is filled out to the specified record length with pad characters if necessary, then written to tape. The line

delimiters themselves are not transferred but they can be restored by the corresponding tape-to-disk transfer routines. If a record exceeds the specified length it is split as explained above in the Record size field description and a count of all such records is kept and displayed after the transfer operation.

After the transfer is complete the number of tape blocks number of logical records and number of bytes in the tape file will be displayed. A prompt will appear at the bottom of the screen asking if you want to append another file to this one; see *Appending Files on Tape* below for an explanation of this choice. To transfer another file simply hit **(ENTER)** at this point; this will close the tape file just written by writing filemarks after the last block and prepare you for writing another new file after this one. Press **(ESC)** to return to the unlabeled tape submenu.

String Data Transfer

```
-----<F10.= Help ESC.= Exit Routine>-----
                                                    <Disk0Tape>

Input file name \binary\prog1.exe
Translate data? (N=no A=EBCDIC to ASCII E=ASCII to EBCDIC) N
Data type: (F)ixed (V)ariable (S)tring S
Output tape block size 60000
Characters to skip 0
Characters to transfer ALL
Terminate on EOF (X'IA') character N

4 blocks 189952 bytes copied
Do you wish to append another DOS file? (n/y) n
```

Figure 5. Disk to unlabeled tape-string transfer fields.

When a string-data transfer is chosen four (4) additional fields appear on the lower half of the screen:

1. Output tape block size: this specifies the actual tape block size to be written up to 65 536 bytes.
2. Characters to skip: this field pre-filled with zero specifies an optional number of characters to be skipped before the data transfer to tape starts.
3. Characters to transfer: this field pre-filled with ALL specifies an optional limit on how many characters to transfer to tape. If the limit is needed enter the number of characters in this field.

4. Terminate on EOF (X'1A') character? If the data to be transferred is in a normal ASCII text file this option can be chosen (by entering Y) to end the transfer on the conventional ASCII end-of-file character (hexadecimal 1 A decimal 26 also known as Control-Z). This option should not be chosen (by entering N) if so-called binary data is to be transferred (i.e. executable programs binary image data etc.) because such data is likely to contain many hex 1 A characters which would falsely be interpreted as end-of-file.

Copy from Tape to Disk

Use this option to copy a file from tape at the current tape position to a disk file. Unlike the labeled tape tape-to-disk transfer this operation is always carried out at the current tape position. The tape must be correctly positioned before this operation is started; use the Reposition tape or display records option on this submenu if the tape is not where you want it. Initially the following three or four fields will appear at the bottom half of the screen:

Output DOS file name: this is the fully-qualified name of the DOS disk file which will be written which must include any necessary drive letters and pathnames to identify the file (for example B:\text\letter1.doc).

Translate data? Three (3) choices are possible:

N - no translation

A - data will be translated from EBCDIC to ASCII

E - data will be translated from ASCII to EBCDIC.

Data type: this specifies how the incoming data from tape will be treated when transferring it to disk. There are four choices:

- 1 Fixed-length records: the tape file is assumed to be divided into equal-length records of a size given by the user. These logical records are simply copied from the tape blocks to the disk file as they are read. No line delimiters are read or written.
- 2 Variable-length records: this really means variable-length physical blocks on tape. Each block in the tape file is treated as a separate logical record and these blocks can vary in size. In transferring these blocks to disk file records line-end delimiters (e.g. carriage-return/line feed pairs) can be added to the end of each record to produce a valid ASCII-type file—

that is, a file that can be successfully displayed using the TYPE command or printed on a line printer.

- 3 IBM Variable-length records: this is a special type of variable length record where each logical record has a 4-byte prefix which contains a binary count of characters in the record. This count is contained in the first word (2 bytes) of the prefix and the second word of the prefix is usually binary zero. This option is generally only used when transferring files from tapes written on IBM mainframe computer systems .
- 4 String data: this option is used in the case where the tape file is not divided into logical records at all (for example an executable program file which consists of binary data). The tape data is treated as a continuous string of data. The tape file is divided into blocks but the blocks are simply fixed length chunks of the input file with a possible short last block. All tape data is transferred to disk contiguously (no breaks).

Filter on (N/Y): this field will only appear if the translation filter has been initialized from the main menu. If Y(es) is specified data from the tape is translated according to statements in the filter file before being written to the disk file. The default (pre-filled) value here is N(o). See the section on the DOS filter function starting on page 144 for more information. Regardless of the type of data transfer chosen TAPEUTL will search for the given output disk file to see if one already exists. If it does the following message will appear:

DOS file already exists -do you want to erase old file (Y/N)?

If you answer Y at this point the existing file will be replaced and all data currently in the file will be lost. If you answer N the transfer will not start until you hit **ENTER** again giving you an opportunity to change the output file name.

If the target volume (fixed disk or diskette) that the file is being copied to becomes full during the transfer operation the transfer will be suspended and the following message will appear at the bottom of the screen (Note: Output breaks at a record boundary when the output DOS volume becomes full):

DOS volume full-new file name (ESC to abort)

If you want to continue the transfer you will need to enter the name of the file which will contain the next part of the tape file including any needed drive letter and pathname specifiers. The transfer will resume with the next (1 byte (character) of the tape

file on the new disk file. To end the transfer hit **(ESC)**; the data on the full disk volume will be retained and you will return to the unlabeled tape submenu.

Depending on the data type chosen for the transfer additional fields will appear below these fields. The complete screens for each data type are shown next with a description of the remaining fields.

1. Pad character: used in conjunction with the preceding option this specifies the pad character to be (optionally) stripped from the output record. This entry is specified as a 2-digit hexadecimal value corresponding to a single character and is pre-filled with the most common pad character the ASCII space (hex 20).
2. Insert line end? this field and the next set options that allow a tape file of fixed-length records to be written to disk as a file of variable-length records with line-end delimiters. If Y is chosen here then each logical record from tape will have line-end delimiters appended in the output record. By appending delimiters the file will be essentially a normal ASCII text-type file (assuming that the data is ASCII text).
3. Line end characters (In hex): in conjunction with the preceding field this specifies the line-end characters to be appended to each output record if Insert line end is chosen. These characters (up to 10) are specified as 2-digit hexadecimal values. The field is pre-filled with the most common line delimiters (carriage-return, line-feed characters (hex OD-OA decimal 13-10)).
4. Records to skip: this field pre-filled with zero specifies an optional number of records from the input file to be skipped before the data transfer to tape starts.
5. Records to transfer: this field pre-filled with ALL specifies an optional limit on the number of records to be transferred to tape. If only a certain number of records are to be copied enter the number in this field.

After the transfer is completed a message will appear stating the number of records transferred. After clearing this message you can start another tape-to-disk transfer by filling in the appropriate fields and hitting **(ENTER)**; the tape will be positioned at the start of the next file. Hit **(ESC)** to exit to the unlabeled tape submenu in order to reposition the tape etc.

Variable-Length Data Transfer

```

-----<F10.= Help ESC.= Exit Routine>-----
Maximum block . 16384                                <Tape0Disk>

output DOS file \temp\junk1.out
Translate data? (N=no A EBCDIC to ASCII E=ASCII to EBCDIC) N
Datatype: (F)ixed (V)ariable (I)BM variable (S)tring v

Record length 1000
Strip trailing pad characters (Y/N) N Pad character 20
Insert line end (Y/N) y
Line end characters (in hex) 0D 0A
Records to skip 0
Records to transfer ALL

Copy complete - 60 records, 57313 bytes - press any key to continue.

```

FIGURE 7. Unlabeled tape to disk-variable-length transfer fields.

When transferring data from tape to disk the term “variable length record” has a completely different meaning than the usual sense of variable-length logical records in a disk file. Keep in mind that all data on tape is written in blocks. All the blocks that make up a tape file do not have to be the same size. From the tape’s point of view these blocks look like records since they must be treated as contiguous chunks (the tape drive cannot stop reading or writing in the middle of a block). Hence variable length records for tape really means variable-length blocks.

All of this is to caution that it is easy to become confused about tape blocks and records. The variable-length unlabeled tape-to-disk transfer option is provided for unusual data transfer needs where each tape record (block) needs to be written to disk as a separate logical record with the options of stripping trailing pad characters and adding line-end delimiters to each output record.

If you are transferring a tape file to disk that was originally written by TAPEUTL (or some other source) from a file of variable-length records you probably need to use the fixed-length transfer option above. This can reconstruct the file in its original form including proper record length and line-end delimiters (assuming that no records were split during the disk-to-tape transfer).

IBM Variable-Length Record Transfer

```

-----<F10.= Help ESC.= Exit Routine>-----

```

```

Maximum block . 16384                                     <Tape0Disk>

Output DOS file \temp\junk2.out
Translate data? (N=no A=EBCDIC to ASCII E-ASCII to EBCDIC) N
Data type: (F)ixed (V)ariable (I)BM variable (S)tring i

Record length 50
Strip trailing pad characters (Y/N) N Pad character 20
Insert line end (Y/N) y
Line end characters (in hex) 0D 0A
Records to ship 0
Records to transfer ALL

Copy complete - 99 records 3997 bytes - press any key to continue:

```

FIGURE 8 Unlabeled tape to disk-IBM variable-length transfer fields.

Files containing IBM variable-length records consist of blocks which are prefixed by a 4-byte block length header. This header contains the total number of bytes in the block including the header. Each record is prefixed by a 4-byte record length header containing the number of bytes in the record including the header. The file and header layouts are shown below.

IBM VARIABLE-LENGTH RECORD LAYOUTS

block length header	record 1 header	record 1 data	record 2 header	record 2 data
------------------------	--------------------	------------------	--------------------	------------------

Figure 9. IBM Variable-length Record File Block Structure.

length count MSB	length count LSB	binary zero	binary zero
---------------------	---------------------	----------------	----------------

Figure 10. IBM Variable-length Record Block/Record Header Fields.

Both block and record headers are 4 bytes (2 words) long with the block or byte count in the first word with the most-significant byte (MSB) first least-significant byte (LSB) second. This is the opposite of how the IBM PC stores binary words in memory. The second word of the header should always be zero (NOT the ASCII character to !). Remember that both the block and record length headers include their own length so that these counts are actually 4 greater than the actual block or record length.

The additional fields that appear on the bottom half of the screen for this type of transfer are the same as for simple variable-length record transfers. See the preceding section for a

detailed description of these fields and of the transfer operation.

String Data Transfer

```
-----<F10.= Help ESC.= Exit Routine>-----
Maximum block 16384                                <Tape0Disk>
Output DOS file \temp\string.dta
Translate data? (N=no A=EBCDIC to ASCII E=ASCII to EBCDIC) N
Datatype: (F)ixed (V)ariable (I)BM variable (S)tring S
Characters to skip 0
Characters to transfer ALL
Copy complete - 189952 bytes transferred - press any key to continue:
```

FIGURE 11. Unlabeled tape to disk-string data transfer fields.

The unlabeled tape-to-disk transfer operation works the same way as a string transfer going from disk to tape data is simply transferred byte-for-byte from tape and written out to the output disk file. Differing block sizes have no effect on the data transfer and data is neither read nor written as logical records but as one continuous string of data.

When a string tape-to-disk transfer is chosen two additional fields appear on the bottom half of the screen:

1. Characters to skip: this field pre-filled with zero specifies an optional number of characters to be skipped before the data transfer to tape starts.
2. Characters to transfer: this field pre-filled with ALL specifies an optional limit on how many characters to transfer to tape. If such a limit is needed enter the number of characters in this field.

Reposition Tape or Display Records

This option chosen from the unlabeled tape submenu is exactly the same as the option of the same name on the main menu; see the section describing this option starting on page 134.

Copy Multiple Files from Tape to Disk

Option D is provided to allow you to concatenate multiple tape files into a single disk file. This is similar to what happens in the opposite direction—disk to tape—when you append a disk file to

a file that has been copied to tape but not yet closed. The specified number of files are read from tape and all the data goes into one disk file.

The multiple-file tape-to-disk transfer operations use the same additional fields as the single-file tape-to-disk transfers; for a particular type of transfer (for example variable-length records) see the appropriate section above. When multiple-file transfer is specified there is one additional field that appears near the bottom of the screen for all data types:

1. Number of files to transfer: this field pre-filled with the value ALL specifies the number of files to be transferred starting at the current tape position. If the logical end-of-tape (two successive filemarks) is found before this number of files is transferred the operation is ended and the disk file closed. When replacing the value in this field be sure to overtype all 3 characters in the field using spaces if less than 3 digits are given.

Summarize Tape Data

This option has been added to the unlabeled menu which allows users to read from the beginning of the tape to the logical end of the tape and summarize each file. A file is defined as data delimited by filemarks. These items are displayed for each file a) minimum block size found b) maximum block size found c) the number of blocks of data in the file and d) the total number of bytes in the file.

Processing Labeled Tapes

Before using the labeled tape operations you should be familiar with the basic concepts of labeled tape formats and processing. Read Chapter 1 of this manual for more information.

When the labeled tape option is chosen from the main menu the following submenu appears and stays on the top half of the screen throughout all labeled tape operations:

```

Flagstaff Engineering Tape Utility Program

A) Copy file from disk to tape
B) Copy file from tape to disk
C) List files on the tape
D) Initialize volume label
E) Switch OS/DOS label type

Main
Unlabeled Tape
>>>Labeled Tape <<<<
Label type OS/DOS
Change Parameters
Initialize Filter

Command: d

-----<F10.= Help ESC.= Exit Routine>-----
    
```

Figure 12. Labeled tape submenu.

About Tape Positioning During Labeled Tape Operations

All the labeled tape operations whether disk-to-tape or tape-to-disk rely on reading the HDR and EOF labels written before and after each file. This may cause delays during transfer operations due to the time necessary for TAPEUTL to find the correct position on the tape. This section is meant to explain how TAPEUTL finds its way around a labeled tape so that you'll understand what is going on when nothing seems to be happening. **DO NOT** attempt to write files on a tape that has no volume label!

The program does not attempt to read the volume label until option A B or C is chosen. If no label exists the position is not underlined—it simply gives an error message and rejects the request. After entering labeled routines the current logical position is the first data file for either read or write.

When you first enter the labeled tape submenu the tape is automatically rewound if it's not already at the load point Then the volume label is read and displayed on the bottom of the screen. The tape is then positioned just before the first file on the tape if there is one. If there are no files on the tape the tape is positioned properly to begin writing the first file. If there is no volume label the tape position will be undefined. Again, **DO NOT** attempt to write files on a tape that has no volume label (if you need to do so you should be in the Unlabeled Tape-section of this manual)!

If you choose to transfer from the current tape position ("C") then no repositioning need be done: TAPEUTL simply assumes that the file you want is the next one on the tape.

Two more items of special note:

- unlike the directory on a disk or diskette it is possible to have non-unique filenames on a tape. If there is more than one file with the same name on a labeled tape and you specify that file by name TAPEUTL will always find the first file and never find the second file. To position to files with identical names you must use the position-by-sequence number option.
- whenever you exit the labeled tape submenu and later re-enter it TAPEUTL always reinitializes the tape position by rewinding if necessary then positioning just past the volume label. Be aware of this if you need to go to another submenu (for example if you want to display a tape block) because the current position may no longer be where you think it should be.

Copy from Disk to Tape

```

Flagstaff Engineering Tape Utility Program

A) Copy file from disk to tape
B) Copy file from tape to disk
C) List files on the tape
D) Initialize volume label
E) Switch OS/DOS label type

                                Main
                                Unlabeled Tape
                                >>>>Labeled Tape <<<<
                                Label Type OS/DOS
                                Change Parameters
                                Initialize Filter

Command: a

                                <F10 . Help ESC Exit Routine>
                                <Disk0Tape>

Tape position (C)urrent/file#/filename/(E)nd C
Input file name \tape\file001.dta          Tape name File # 1
Translate data? (N=no A=EBCDIC to ASCII E=ASCII to EBCDIC) N
Data type: (F)ixed (V)ariable (S)tring S   Filter on (N/Y) N
    
```

Figure 13. Labeled tape submenu-common disk-to-tape fields.

This option is used to create a tape file from data read from a disk file or files with header and EOF labels automatically written before and after the tape file (see Chapter 2 for a discussion of tape labels).

When this option is first chosen from the labeled tape submenu the tape will automatically be positioned correctly to write the first file on the tape (just after the volume label). This is true whether or not there are any files on the tape. If there are existing files on the tape that need to be preserved you must reposition the tape to avoid overwriting this first file: see the

Tape position field description below for an explanation. DOS and OS labels as well as multivolume labeled input and output files are supported.

DOS labels are essentially the same as OS labels except HDR2/EOV2/ EOF2 label records are not present. Therefore format record size and block size are unknowns. DOS labels also have a corresponding ANSI (ASCII) format.

Initially the following five or six fields will appear on the bottom half of the screen:

1. Tape position: this allows the tape to be repositioned before the disk-to-tape transfer starts. There are four possible options:
2. (C)urrent position: as one might guess this is simply where the tape already is. Keep in mind that this current position is at the beginning of the tape when the disk-to-tape option is first chosen from the labeled tape submenu (see above). C is the pre-filled default option.
3. File #: by entering a number here TAPEUTL will look for that file by sequence number on tape starting at the first file. If the file is found the tape will be positioned at the start of that file ready to start overwriting the file. If the number given is greater than the total number of files on the tape a File Not Found message will be displayed.
4. Filename: if the name of a file is entered in this field TAPEUTL will attempt to locate a file with that name written in its HDR1 label. If the file is found the tape will be positioned at the start of the next file as described above; if no file by that name is found a File Not Found message will be displayed.
5. Input file name: this is the fully-qualified name of the DOS disk file which must include any necessary drive letters and pathnames to identify the file (e.g., B:\text\letter1.doc).
Translate data? Three (3) choices are possible:
 - N - no translation
 - A - data will be translated from EBCDIC to ASCII
 - E - data will be translated from ASCII to EBCDIC.
6. Data type: this specifies how the incoming data is to be treated. Since data on tape is always written in blocks the entry in this field will control how the disk file data is placed into the tape blocks. There are three (3) possibilities:
 - Fixed-length records: data is read from the disk file as records of a specified length (given by the user). The disk

data is not scanned for line-end characters or other delimiters but is assumed to be divided into logical records of the user-specified length.

- Variable-length records: the disk file is assumed to be in logical records of varying length, each record terminated by an identical sequence of character(s). With this type of transfer, data is read from disk as variable-length but written to tape as fixed-length records (short records are padded out with a fill character).
 - String data: this option is used in the case where the disk data file is not divided into logical records at all (e.g., an executable program file which consists of binary data). The disk data is treated as a continuous string of data. The tape file is divided into blocks, but the blocks are simply fixed length chunks of the input file, with a possible short last block.
7. Filter on (N/Y): this field will only appear if the translation filter has been initialized from the main menu. If Y(es) is specified, data from the input file is translated according to statements in the filter file before being written to tape. The default (pre-filled) value here is N(o). See the section on the DOS filter function on page 144 for more information.

Depending on the data type chosen for the transfer, additional fields will appear below these fields. The complete screens for each data type are shown next, with a description of the remaining fields.

Fixed-Length Data Transfer

```

-----<F10.= Help ESC.= Exit Routine>-----
                                                    Disk0Tape>

Tape position (C)urrent/file#/filename/(e)nd e
Input file name \onfile\onfile.exe           Tape name ramboloon
Translate data? (N=no, A=EBCDIC to ASCII, E=ASCII to EBCDIC) N
Data type: (F)ixed, (V)ariable, (S)tring f
Record size 256
Records per output tape block 100
Records to skip 0
Records to transfer ALL

8 blocks 742 records 189952 bytes copied
Do you wish to append another DOS file? (n/y):
    
```

Figure 14. Disk to labeled tape-fixed-length transfer fields.

Four additional fields appear for fixed-length transfer operations:

1. Record size: this specifies the length of each logical record in the disk file. Remember that TAPEUTL doesn't perform any checking of the input records, but assumes that the input records are, in fact, this length. If you're not sure what length the file records are, look at the file first with a file dump/display utility (for example, Flagstaff Engineering's DFILE program) to determine the proper record size.
2. Records per output tape block: this is the blocking factor to be used on tape. This number multiplied by the record length gives the tape block size in bytes, which can be a maximum of 65,536 bytes (65,519 for the OD3210). If this size is exceeded, the program will display a warning message and you will be asked to change either the record size or the number of records per block.
3. Records to skip: this field, pre-filled with zero, specifies an optional number of records from the input file to be skipped before the data transfer to tape starts.
4. Records to transfer: this field, pre-filled with ALL, specifies an optional limit on the number of records to be transferred to tape. If only a certain number of records are to be copied, enter the number in this field.

When a fixed-length transfer is started, TAPEUTL gets the size of the input file from the disk directory, displays it, and compares this size to the record size to determine whether or not there will be a short record at the end of the file. If there is, a message will appear at the bottom of the screen asking if you want to transfer this partial record. Answering Y will cause this last record to be padded with binary zeros (0) at the end of the file; answering N will simply drop this record from the tape file. There will usually be a short block at the end of a fixed-length, blocked record file.

After the transfer is complete, the number of tape blocks, number of logical records and number of bytes in the tape file will be displayed. A prompt will appear at the bottom of the screen asking if you want to append another file to this one; see *Appending Files on Tape* below for an explanation of this choice. To transfer another file, simply hit **(ENTER)** at this point. This will close the tape file by writing EOF labels after the last block and

prepare you for writing another, new file after this one. Press **(ESC)** to return to the labeled tape submenu.

Variable-Length Data Transfer

```

-----<F10.= Help ESC.= Exit Routine>-----
                                                    <Disk0Tape>
Tape position (C)urrent/file#/filename/(e)nd C
Input file name \doc\taputil2.asc      Tape name databait!!
Translate data? (N=no, A=EBCDIC to ASCII, E=ASCII to EBCDIC) N
Data type: (F)ixed, (V)ariable, (S)tring v
Record size 80
Records per output tape block 200
Records to skip 0
Records to transfer ALL
Line delimiter OD 0A
Pad character 20

2 blocks 238 records 19040 bytes copied
      WARNING! 6 records exceeded specified length
      —press any key to continue:
    
```

Figure 15. Disk to labeled tape-variable-length transfer fields.

Six additional fields appear when a variable-length transfer is chosen:

1. Record size: this is the size of the tape (output) record, which is the size of the largest record in the file not including line delimiter characters. Remember that variable-length only applies to the disk (input) file; the 2 blocks 238 records 19040 bytes copied tape file is written as fixed-length records. If a record is shorter than the given record size, it is right-filled with pad characters (see below). If a record is longer than this size, it will be split—the excess part will be written as a new record—and a warning message will be displayed after the transfer:

WARNING! nn records exceeded specified length
 If this happens, try the transfer again with a larger record size. No data will be lost in any case, but the file may need to be reformatted to restore lines to their original form.
2. Records per output tape block: this is the blocking factor to be used on tape. This value multiplied by the record length gives the tape block size in bytes, which can be a maximum of 65,536 bytes (65519 bytes for the OD3210). If this size is exceeded, the program will display a warning message and

you will be asked to change either the record size or the number of records per block.

3. Records to skip: this field, pre-filled with zero, specifies an optional number of records from the input file to be skipped before the data transfer to tape starts.
4. Records to transfer: this field, pre-filled with ALL, specifies an optional limit on the number of records to be transferred to tape. If only a certain number of records are to be copied, enter the number in this field.
5. Line delimiter: this specifies which character or string of characters are to be taken as an end-of-line marker in the input file. This entry must correspond to the actual line delimiters in the file for the transfer to be successful. The characters (up to 10) are specified as 2-digit hexadecimal values, with the most common values pre-filled: 0DOA, or ASCII carriage-return and line-feed characters.

CAUTION! See the section on the DOS filter function starting on page 144 for an explanation of pitfalls to avoid if you are using a translation filter.

6. Pad character: the character which will be used to fill short records is specified here as a 2-digit hexadecimal value. This field is pre-filled with a commonly-used fill character, the ASCII space (hex 20). The EBCDIC space character is hex 40.

The variable-length data transfer works in this manner: as the input file is read from disk, TAPEUTL continually scans the data for the specified line delimiter characters. When the delimiter string is found, the record is filled out to the specified record length with pad characters if necessary, then written to tape.

The line delimiters themselves are not transferred, but they can be restored by the corresponding tape-to-disk transfer routines. If a record exceeds the specified length, it is split as explained above in the Record size field description, and a count of all such records is kept and displayed after the transfer operation. For a variable-length transfer, data remaining at the end of a file will always be written as a short block on tape if necessary.

After the transfer is complete, the number of tape blocks, number of logical records and number of bytes in the tape file will be displayed. A prompt will appear at the bottom of the screen asking if you want to append another file to this one; see *Appending Files on Tape* below for an explanation of this choice.

To transfer another file, simply hit **(ENTER)** at this point. Press **(END)** to return to the labeled tape submenu.

String Data Transfer

```
-----<F10.= Help ESC.= Exit Routine>-----
                                                    <Disk0Tape>

Tape position (C)urrent/file#/filename/(E)nd e
Input file name \onfile\onfile.exe           Tape name calafragilistic!
Translate data? (N=no, A=EBCDIC to ASCII, E=ASCII to EBCDIC) N
Data type: (F)ixed, (V)ariable, (S)tring S
Output tape block size 32760
Characters to skip 0
Characters to transfer ALL
Terminate on EOF (X'1A') character N

6 blocks 139952 bytes copied
Do you wish to append another DOS file? (n/y):
```

Figure 16. Disk to labeled tape-string data transfer fields.

When a string-data transfer is chosen, four additional fields appear on the lower half of the screen:

1. Output tape block size: this specifies the actual tape block size to be written, up to 65,536 bytes (65,519 bytes for the OD3210).
2. Characters to skip: this field, pre-filled with zero, specifies an optional number of characters to be skipped before the data transfer to tape starts.
3. Characters to transfer: this field pre-filled with ALL specifies an optional limit on how many characters to transfer to tape. If such a limit is needed enter the number of characters in this field.
4. Terminate on EOF (X'1A') character? If the data to be transferred is in a normal ASCII text file this option can be chosen (by entering Y) to end the transfer on the conventional ASCII end-of-file character (hexadecimal 1A decimal 26 also known as Control-Z). This option should not be chosen (by entering N) in so-called binary data is to be transferred (i.e. executable programs image data, etc.) because such data is likely to contain many hex 1A characters which would falsely be interpreted as end-of-file.

A string data transfer will simply copy all the data from the disk file byte-for-byte to the tape file with no processing or alteration

unless translation between ASCII and EBCDIC is chosen or if the data is filtered (see the section on the DOS filter function starting on page 144). When there is any question about the form of the disk file this option is the safest way to transfer data to assure that no data is lost.

After the transfer is complete the number of tape blocks and number of bytes in the tape file will be displayed. A prompt will appear at the bottom of the screen asking if you want to append another file to this one; see *Appending Files on Tape* below for an explanation of this choice. To transfer another file simply hit **(ENTER)** at this point. Press **(ESC)** to return to the labeled tape submenu.

Copy from Tape to Disk.

```

Flagstaff Engineering Tape Utility Program
A) Copy file from disk to tape
B) Copy file from tape to disk
C) List files on the tape Unlabeled
D) Initialize volume label
E) Switch OS/DOS label type

Main
Tape
>>>>Labeled tape <<<<
Label Type OS/DOS
Change Parameters
Initialize Filter

Command: b

-----<F10.= Help ESC.= Exit Routine>-----
Maximum block = 16384
Tape position (C)urrent/file#/filename) C
Output DOS file
Translate data? (N=no, A=EBCDIC to ASCII, E=ASCII to EBCDIC) N
Data type: (F)ixed, (V)ariable, (I)BM variable, (S)tring Filter on (N/Y) N
    
```

Figure 17. Labeled tape Submenu-common tape-to-disk fields.

This option on the labeled tape submenu allows tape files to be transferred to disk. When this option is first chosen from the labeled tape submenu, the tape will automatically be positioned to read the first file on the tape, as indicated by a message that appears at the bottom of the screen. The first field that appears on the lower half of the screen asks you where you would like the tape to be positioned when the transfer starts. Filling in this field and hitting **(ENTER)** will cause this positioning to take place; to choose the pre-filled option of C(urrent), just hit **(ENTER)**. The positioning options are described below.

After the tape has been positioned, the other 4 or 5 fields common to all tape-to-disk transfers will appear (the 5th field

only appears if you have initialized the filter: see the section on the DOS filter function starting on page 144). Fill in these fields, then press **(ENTER)** to start the transfer operation.

1. Tape position: specifies the position of the tape before the transfer takes place. There are three possible choices:
 - (C)urrent: this leaves the tape in its current position. When you first choose this transfer option from the labeled tape submenu the tape will be positioned at the start of the first file. After a transfer operation the tape will be positioned at the start of the next file on the tape.
 - File #: this positions the tape at the start of a selected file by sequence number on tape. If the requested file is not found before the logical end of tape is found an error message will be displayed.
 - Filename: this positions the tape at the start of a selected file by tape file name. If the named file is not found on the tape (by reading the tape's HDR1 labels) a message will be displayed.
2. Output DOS file: this is the fully-qualified name of the DOS disk file which will be written which must include any drive letters and pathnames needed to identify the file (e.g., B:\text\letter1.doc). If the drive and pathnames are omitted the file is written in the current directory on the current default drive.
 - Translate data? three choices are possible:
 - N - no translation
 - A - data will be translated from EBCDIC to ASCII
 - E - data will be translated from ASCII to EBCDIC
3. Data type: this specifies how the incoming data from tape will be treated in transferring it to disk. There are four possible choices:
 - Fixed-length records: the tape file is assumed to be divided into equal-length records of a size given by the user. These logical records are simply copied from the tape blocks to the disk file as they are read. No line delimiters are read or written.
 - Variable-length records: this really means variable-length physical blocks on tape. Each block in the tape file is treated as a separate logical record and these blocks can vary in size. In transferring these blocks to disk file records

line-end delimiters (e.g. carriage-return line feed pairs) can be added to the end of each record to produce a valid ASCII-type file—that is a file that can be successfully displayed using the TYPE command or printed on a line printer.

- IBM Variable-length records: this is a special type of variable length record where each logical record has a 4-byte prefix which contains a binary count of characters in the record. This count is contained in the first word (2 bytes) of the prefix and the second word of the prefix is usually binary zero. This option is generally only used when transferring files from tapes written on IBM mainframe computer systems.
4. String data: this option is used in the case where the tape file is not divided into logical records at all (for example an executable program file which consists of binary data). The tape data is treated as a continuous string of data. The tape file is divided into blocks but the blocks are simply fixed-length chunks of the input file with a possible short last block. All tape data is transferred to disk contiguously (with no breaks).
 5. Filter on (N/Y): this field will only appear if the translation filter has been initialized from the main menu. If Y(es) is specified data from the tape is translated according to statements in the filter file before being written to the disk file. The default (pre-filled) value here is N(o). See the section on the DOS filter function starting on page 144 for more information.

Regardless of the type of data transfer chosen TAPEUTL will search for the given output disk file to see if one already exists. If it does the following message will appear:

```
DOS file already exists
do you want to erase old file (Y/N)?
```

If you answer Y at this point the existing file will be replaced and all data currently in the file will be lost. If you answer N the transfer will not start until you hit **(ENTER)** again giving you an opportunity to change the output file name.

If the target volume (fixed disk or diskette) that the file is being copied to becomes full during the transfer operation the transfer will be suspended and the following message will appear at the bottom of the screen:

```
DOS volume full
- new file name (ESC to abort)
```

If you want to continue the transfer, you will need to enter the name of the file which will contain the next part of the tape file, including any needed drive letter and pathname specifiers. The transfer will resume with the next byte character) of the tape file on the new disk file. To end the transfer, hit **(ESC)**; the data on the full disk volume will be retained, and you will return to the unlabeled tape submenu.

Fixed-Length Data Transfer

```
-----<F10.= Help ESC.= Exit Routine>-----
Maximum block = 16384                                <Tape0Disk>
Tape position (C)urrent/file#/filename) jazzercise
Output DOS file \temp\jazz
Translate data? (N=no A=EBCDIC to ASCII, E=ASCII to EBCDIC) N
Data type: (F)ixed, (V)ariable, (I)BM variable, (S)tring F Filter on (N/Y)N

Record length 50
Strip trailing pad characters (Y/N) N Pad character 20
Insert line end (Y/N) N
Line end characters (in hex) 0DOA
Records to skip 0
Records to transfer ALL

Copy complete -1310 records transferred
press any key to continue:
```

FIGURE 18. Labeled tape-to-disk fixed-length transfer fields.

Seven additional fields will appear for a fixed-length tape-to-disk transfer operation:

1. Record length: this specifies the length of one logical tape record. When the tape file is located, the record length given in the HDR2 label is put in this field. To override this length, type in the new length here.
2. Strip trailing pad characters? If Y is chosen here then TAPEUTL will remove all trailing pad characters in the output record (up to the first non-pad character from the end). Usually if this option is chosen the pad character will be a space and this option is chosen to eliminate wasted file space. If N is chosen no characters are removed from the output record. In either case only the output (disk) file is affected, not the input (tape) data.

3. Pad character: used in conjunction with the preceding option this specifies the pad character to be (optionally) stripped from the output record. This entry is specified as a 2-digit hexadecimal value corresponding to a single character and is pre-filled with the most common pad character, the ASCII space (hex 20).
4. Insert line end? this field and the next set options that allow a tape file of fixed-length records to be written to disk as a file of variable-length records with line-end delimiters. If Y is chosen here then each logical record from tape will be have line-end delimiters appended in the output record. By appending delimiters the file will be essentially a normal ASCII text-type file (assuming that the data is ASCII text).
5. Line end characters (In hex): used in conjunction with the preceding option this specifies the line-end characters to be appended to each output record if `Insert line end` is chosen. These characters (up to 10) are specified as 2-digit hexadecimal values. The field is pre-filled with the most common line delimiters the ASCII carriage-return and line-feed characters (hex OD-OA decimal 13-10)
6. Records to skip: this field pre-filled with zero specifies an optional number of records from the input file to be skipped before the data transfer to tape starts.
7. Records to transfer: this field pre-filled with ALL specifies an optional limit on the number of records to be transferred to tape. If only a certain number of records are to be copied enter the number in this field.

After the transfer is completed a message will appear stating the number of records transferred. After clearing this message you can start another tape-to-disk transfer by filling in the appropriate fields and hitting **ENTER**; the tape will be positioned at the start of the next file. Hit **ESC** to exit to the labeled tape submenu.

Variable-Length Data Transfer

```
-----<F10.= Help ESC.= Exit Routine>-----  
Maximum block- 16384                                     <Tape0Disk>  
Tape position (C)urrent/file#/filename) 4  
Output DOS file \temp\slaboom  
Translate data? (N=no, A=EBCDIC to ASCII, E=ASCII to EBCDIC) N  
Data type: (F)ixed, (V)ariable, (I)BM variable, (S)tring v Filter on (N/Y)N  
  
Record length 256  
Strip trailing pad characters (Y/N) N Pad character 20  
Insert line end (Y/N) N  
Line end characters (in hex) 0DOA  
Records to ship 0  
Records to transfer ALL  
  
Copy complete - 8 records, 266241 bytes  
press any key to continue:
```

FIGURE 19. Labeled tape to disk-variable-length transfer fields.

Seven additional fields will appear for a variable-length tape-to-disk transfer operation:

1. Record length: this field is for notation purposes only. If the tape file was formatted with fixed-length records, the record length from the file's HDR2 label will be prefilled here; otherwise, this field will be blank. This field is ignored by TAPEUTL, except that there must be a non-zero value entered here.
2. Strip trailing pad characters? If Y is chosen here, then TAPEUTL will remove all trailing pad characters in the output record (up to the first non-pad character from the end). Usually, if this option is chosen, the pad character will be a space, and this option is chosen to eliminate wasted file space. If N is chosen, no characters are removed from the output record. In either case, only the output (disk) file is affected, not the input (tape) data.
3. Pad character: used in conjunction with the preceding option, this specifies the pad character to be (optionally) stripped from the output record. This entry is specified as a 2-digit hexadecimal value corresponding to a single character, and is pre-filled with the most common pad character, the ASCII space (hex 20).
4. Insert file end? this field and the next set options that allow a tape file of fixed-length records to be written to disk as a

file of variable-length records with line-end delimiters. If Y is chosen here, then each logical record from tape will be have line-end delimiters appended in the output record. By appending delimiters, the file will be essentially a normal ASCII text-type file (assuming that the data is ASCII text).

5. Line end characters (In hex): used in conjunction with the preceding option, this specifies the line-end characters to be appended to each output record if Insert line end is chosen. These characters (up to 10) are specified as 2-digit hexadecimal values. The field is pre-filled with the most common line delimiters, the ASCII r carriage-return and line-feed characters (hex 0D-0A, decimal 13-10).
6. Records to skip: this field, pre-filled with zero, specifies an optional number of records from the input file to be skipped before the data transfer to tape starts.
7. Records to transfer: this field, pre-filled with ALL, specifies an limit on the number of records to be transferred. If only a some records are to be copied, enter the number in this field.

After the transfer is completed, a message appears stating the number of records transferred. After clearing this message, you can start another tape-to-disk transfer by filling in the fields and hitting **(ENTER)**; the tape will be positioned at the start of the next file. Hit **(ESC)** to exit to the labeled tape submenu.

IBM Variable-Length Record Transfer

```

-----<F10.= Help  ESC.= Exit Routine>-----
Maximum block . 50000                                <Tape0Disk>
Tape position (C)urrent/file#/filename) C
Output DOS file \temp\ibmvar
Translate data? (N=no A=EBCDIC to ASCII E=ASCII to EBCDIC) N Data type:
(F)ixed (V)ariable (I)BM variable (S)tring i Filter on (N/Y)N

Record length 50
Strip trailing pad characters (YIN) N Pad character 20
Insert line end (Y/N) N
Line end characters (in hex) 0DOA
Records to skip 0
Records to transfer ALL

Copy complete - 3 records 65757 bytes -
press any key to continue:
    
```

Figure 20. Labeled tape to disk-IBM variable-length transfer fields.

IBM variable-length record files consist of blocks prefixed with a 4-byte block length header containing the number of bytes in the block and each record is prefixed with a 4-byte record length header containing the number of bytes in the record. The layout of these blocks and headers is shown in Chapter 1.

The additional fields that appear on the bottom half of the screen for this type of transfer are the same as for simple variable-length record transfers. See the preceding section for a detailed description of these fields and of the transfer operation.

String Data Transfer

```

-----<F10.= Help ESC.= Exit Routine>-----
Maximum block 50000                                <Tape0Disk>
Tape position (C)urrent/file#/filename) C
Output DOS file FROWN
Translate data? (N=no, A=EBCDIC to ASCII, E=ASCII to EBCDIC) N
Data type: (F)ixed, (V)ariable, (I)BM variable, (S)tring s Filter on (N/Y)N

Characters to skip 0
Characters to transfer ALL

Copy complete - 13250 bytes transferred
press any key to continue:
    
```

Figure 21. Labeled tape to disk-string data transfer fields.

The labeled tape-to-disk string data transfer operation works the same way as a string transfer going from disk to tape: data is simply transferred byte-for-byte from tape and written out to the output disk file. Differing block sizes have no effect on the data transfer, and data is neither read nor written as logical records, but as one continuous string of data.

When string data is written to a labeled tape, the record type in the HDR2 label is set to U for Undefined, and no record length (a length of zero) is recorded in the label. Two additional fields appear on the bottom half of the screen:

1. Characters to skip: this field, pre-filled with zero, specifies an optional number of characters to be skipped before the data transfer to tape starts .
2. Characters to transfer: this field, pre-filled with ALL, specifies an optional limit on how many characters to transfer to tape. If such a limit is needed, enter the number of characters in this field.

List Files on the Tape

This option gives a directory listing of the tape. By reading each file's header labels (HDR1 and HDR2), TAPEUTL lists each file in sequence and displays the following information:

- Sequence number: the number of the file, starting at 1.
 - Name: the name of the file recorded in the HDR1 label, up to 17 characters long.
 - Type: F for fixed-length records V for IBM variable-length records U for undefined type.
 - Block Size: the physical size of the tape blocks (maximum size—file may contain short blocks at the end or in the middle).
 - Record size: the logical record length recorded in the file's HDR2 label (always zero for type "U" files).
 - Number of blocks: number of physical tape blocks in the file.
- If the screen fills up with entries and there are more files to display, the display will stop and the following message will appear:

More - press any key to continue:

Hit any key to display the next screenful of entries. When the last file on the tape has been found, the message:

Directory list completed

will appear at the bottom of the screen. The display will remain on screen until you hit any alpha/numeric key to return to the labeled tape submenu.

-----<F10.= Help ESC.= Exit Routine>-----					
seq.	name size	type size	block	record	number blocks
1	GORILLA	U	20000	0	1
2	SETDN	U	200	0	1
3	ZRAY1	U	20000	0	4
4	TURGIL	F	6000	60	11
5	SCUMBY99	F	12800	128	2
Directory list completed - press any key to continue:					

Figure 22. Directory Listing for Labeled Tape

Initialize Volume Label

This submenu option prepares a tape for writing labeled files to in This is analogous to formatting a diskette by using the DOS FORMAT command). A standard ANSI or IBM label is written at the beginning of the tape, and the tape is positioned to write the first file. You can also choose to completely erase the tape after the volume label to eradicate any existing data; however, this operation takes several minutes.

Because this operation writes a new volume label on the tape, any data on the tape should be considered destroyed even if you choose not to erase to the end of the tape. This is because volume labels may not be written in precisely the same place by different systems, even by different runs on the same system. If the first file's HDRI label is overwritten by the VOL1 label, the data can be considered "trashed." If a tape is accidentally initialized, you may still be able to recover files from it. Use the Reposition Tape and Display Records option from the unlabeled tape submenu to examine the existing data on the tape.

When initializing a labeled tape, four fields appear on the bottom half of the screen:

- 1 Type of label: (this can be one of two types):
 - A — writes an ANSI label (in ASCII).
 - B — writes an IBM label (standard OS/VS) encoded in EBCDIC.
- 2 Volume ID: this is a 1 to 6-character name which will be written as the tape volume name. This is the name by which the tape would be identified to a minicomputer or mainframe operating system. When entering alphabetic characters, TAPEUTL changes all lowercase letters to uppercase. TAPEUTL does no checking on names, but the host system may impose rules for valid names.
- 3 System Name: A system name may be entered when initializing tapes. This field may be up to ten characters in length on IBM volumes and up to fourteen characters long on ANSI volumes. This system name is placed in volume VOL1 record. If no name is entered here, the ID field in the VOL1 label will be filled with spaces.
- 4 Erase to end of tape? This field is pre-filled with N. If Y is entered, the tape will be physically erased to the end marker

to eradicate any existing data. This operation can take several minutes to complete. If this option is chosen and you decide to stop while the tape is being erased, hold down both **(SHIFT)** keys simultaneously to stop the erase operation. This may cause an error message to be displayed, but hitting any key (or **(ESC)**) should clear this message and return you to the labeled tape submenu.

After these fields are filled in and you press **(ENTER)** to start the initialization, the message:

Tape volume label written

should appear at the bottom of the screen. - If the erase to end option was used, the tape will run for several minutes, after which you will be back at the labeled tape submenu.

Positioning the Tape and Displaying the Records

The tape positioning and display submenu looks like this:

```
-----<F10.= Help  ESC.= Exit Routine>-----  
  
A) Forward space by files          G) Rewind and unload  
B) Back space by files            H) Write filemark  
C) Forward space by blocks        I) Display tape record on screen  
D) Back space by blocks          J) Erase to end of tape  
E) Space to logical end of tape  
F) Rewind  
  
Command:
```

Figure 23. Tape positioning and Display Submenu.

All the operations on this submenu will work with either unlabeled or labeled tapes. When using these functions however TAPEUTL doesn't know whether the tape is labeled or not. Therefore tape labels are treated the same as separate data files when the space forward or backward and display block functions are used. You can display a tape's VOL HDR and EOF labels just as you can look at any other data on tape.

NOTE: Any repositioning done here on labeled tape is lost when you return to the labeled tape menu! When processing a labeled tape the rewind function on this submenu should be used only to reposition the tape for the purpose of displaying tape data (using the display option on the submenu) or when doing transfers to or from unlabeled tape. Do not use this rewind

function in conjunction with labeled tape transfer operations (tape-to-disk or vice versa); when the labeled tape submenu is re-entered after leaving this submenu the tape will automatically be repositioned anyway. See the section on processing labeled tape for further explanation.

How the Repositioning Operations Work

On this submenu notice that there are two types of spacing (repositioning) operations: by blocks or by files (filemarks). Each can work either forward or backward. All spacing operation quantities (blocks or files) are counted from the current tape position. The following explains how positioning works.

Remember that a tape consists of data blocks separated by interrecord gaps, and that files are separated by file marks, or filemarks. Remember that labeled tapes have special records (labels) which are used to record tape and file information. Refer to Chapter 1 for an explanation of tape data layouts.

Spacing Forward/Backward by Files

When spacing by files, TAPEUTL scans the tape and counts filemarks encountered until the specified number of filemarks has passed (the default, if no number is entered with this command, is one filemark). The tape will then be left positioned just after the last filemark found.

If the reposition by filemarks operation is backwards, the first filemark counted is the one at the start of the file the tape is currently positioned at regardless of what the current position is within that file. If the current position is at the beginning of the file, then a backspace-by-one operation will position the tape at the beginning of the previous file, unless the current file is the first one on the tape.

Some examples of spacing by files: (note that the tape is always positioned at an IRG between blocks or files)

Spacing example 1.

BEFORE:

	file 1	file 1	...	T	file 2	file 2	T	file 3	...
	block 1	block 2	...	M	block 1	block 2	M	block 1	...

old current position ↑

AFTER: (command given was "backspace 2 files"):

file 1	file 1	...	T	file 2	file 2	T	file 3	...
block 1	block 2	...	M	block 1	block 2	M	block 1	...

↑ new current position

Spacing example 2.

BEFORE:

file 1	file 1	T	file 2	file 2	T	file 3	T	file 4	T
block 1	block 1	M	block 1	block 2	M	block 1	M	block 1	M

↑ old current position

AFTER: (command given was "forward space 3 files"):

file 1	file 1	T	file 2	file 2	T	file 3	T	file 4	T
block 1	block 1	M	block 1	block 2	M	block 1	M	block 1	M

new current position ↑

Spacing Forward/Backward by Blocks

When spacing in either direction by blocks, TAPEUTL scans the tape from the current position and counts the number of blocks that have passed. When this number matches the number of blocks given in the command, the spacing operation is stopped and the tape is positioned at the start of a tape block. (If the tape is labeled, this could be either a data block or a tape label.)

If a filemark is found during the space-by-blocks operation, the operation is halted and the message `File Mark Found` will appear at the bottom of the screen. The tape will then be positioned at the start of the file after the filemark that was found. If the space operation was in the backwards direction, this will be the same file. If the space direction was forward, this will be the start of the next file. Unless a filemark is found, the space-by-blocks operation always stays within the same file. To space within another tape file, you must first get to that file using the space-by-files option (see previous section).

Remember that a tape consists of data blocks separated by interrecord gaps, and that files are separated by file marks, or filemarks. Keep in mind also that labeled tapes have special records (labels) which are used to record tape and file information. Refer to section one for a basic explanation of tape data layouts.

Some examples of spacing by blocks:

Spacing example 3. (Note that TAPEUTL jumps over filemarks in the forward direction.)

BEFORE:

	file 1	file 1	file 1	file 1	T	file 2	file 2	T	...
	block 1	block 2	block 3	block 4	M	block 1	block 2	M	...

↑ new current position

AFTER (command given was "forward space 6 blocks"):

	file 1	file 1	file 1	file 1	T	file 2	file 2	T	...
	block 1	block 2	block 3	block 4	M	block 1	block 2	M	...

new current position ↑

Spacing example 4.

BEFORE:

	file 1	file 1	file 1	file 1	T	file 2	file 2	T	...
	block 1	block 2	block 3	block 4	M	block 1	block 2	M	...

old current position ↑

AFTER (command given was "backspace 4 blocks"):

	file 1	file 1	file 1	file 1	T	file 2	file 2	T	...
	block 1	block 2	block 3	block 4	M	block 1	block 2	M	...

↑ new current position

Note that the tape is positioned at the start of the file if a filemark is found before the given number of blocks are skipped.

Space to Logical End of Tape

This function reads the tape until the logical end-of-tape is found. This is written on the tape as two consecutive tape (file) marks following the last file on the tape. Note that this is not the same as the physical end-of-tape. The logical end of tape may be very close to the start of the tape if there is not much data on the tape.

If there is no logical end-of-tape, the drive will continue to run until the tape runs off the supply reel. If this happens, wind the tape back on the supply reel and wind the tape onto it by hand

at least 30-40 turns, then use the drive's Load control to re-load the tape.

Rewind Tape

This function simply issues the rewind command to the tape controller. While the tape is rewinding, a message will be displayed at the bottom of the screen, and TAPEUTL will wait until the tape is actually rewound before returning control to you. If you want to do something else before a long rewind operation is done, hitting **(SHIFT)+(SHIFT)** will cause the message:

Rewind stopped

to appear, and you can choose another function. However, this doesn't stop the rewind operation itself, and you must wait until the tape is rewound before starting any tape operations. You can use such functions as changing the tape parameters, which don't directly send commands to the tape drive.

Rewind and Unload Tape

This function is identical to the rewind function (above), except that the tape is physically unloaded from the drive following the rewind. After this command executes, you must reload a tape in the drive to do further tape processing .

Write Tapemark

This function will write a filemark at the current tape position, then space the tape after the filemark. THIS FUNCTION SHOULD BE USED WITH CAUTION!! It is provided only for special circumstances, when you are certain about the tape's current position and need one or more filemarks written there. If you write two filemarks in a row, you create a logical end-of-filemarker. This can be used when re-writing a tape that already has data on it, so that you can be sure that no data will be read past the new logical end-of-tape.

NOTE: All disk-to-tape transfer functions create double filemarks at the end of the file, which leaves the tape with a valid logical end-of-filemarker.

Displaying Tape Records

This function can be used with both labeled and unlabeled tapes to view the contents of any tape records, including tape labels. Tape data is displayed in both hexadecimal and character (ASCII or EBCDIC) formats, in lines of 16 bytes on the screen. When

you choose this option, data is displayed starting at the current tape position. If the tape is positioned at a filemark (end of tape file), the first record after the filemark will be displayed.

Remember that a tape record is actually a tape block.

The tape display function has a different screen display than the rest of TAPEUTL. The top part of the screen shows 20 lines (320 bytes) of data in hex/character formats, with the displacement (distance from the start of the block) shown in decimal at the left. The current character representation (ASCII or EBCDIC) is shown at the top right. The bottom line shows the block number and block length, and contains a command entry field.

While in the display function, you can space forward by tape blocks, displaying data from any part of each block, until a filemark is read. When the end of a file is hit, TAPEUTL displays the following message:

```
Tape EOF (filemark) found -
press any key to continue:
```

The current screen will remain until you hit a key, after which you will be returned to the tape reposition/display submenu. If you choose the display option again without repositioning the tape, the first block of the next file (if any) will be displayed. There are several special keys you can use to select tape blocks to display and to select the position within the block. They are:

(PGUP)—displays the previous screen of data from this block.

(PGDN)—displays the next screen of data from this block.

(HOME)—displays the previous block from the tape.

(END)—displays the next block from the tape.

(↓)—scrolls the screen up one line (displays next 16 data bytes).

(↑)—scrolls the screen down one line (displays previous 16 bytes).

(F1)—toggles the character display between ASCII and EBCDIC.

(ESC)—exits the display routine.

In addition, there are two positioning commands that can be entered on the display command line:

nnn—display data starting at decimal offset **nnn** into the current tape block

Bnnn—display tape block number **nnn**.

(type these commands, then hit **(ENTER)**)

For the second command, the operation is the same as the space by blocks options from the reposition/display submenu. It

a filemark is read before the specified tape block is read, the operation will be ended. Also note that the display routine may not keep track of the correct block location in the file if you forward space to a filemark, then backspace by blocks within the file.

CHANGING THE DISPLAY BETWEEN ASCII AND EBCDIC

The **(F1)** key toggles the display of characters in the file between ASCII and EBCDIC representations. This affects only the right-hand part of the display (the hex display stays the same). If you are displaying data that should be readable text and you can't see any recognizable alphabetic characters, you may be using the wrong character set. ASCII characters displayed as EBCDIC usually appear as "garbage," and vice versa. Binary data (binary data files, executable programs, etc.) will always appear as meaningless bytes, although you should be able to read embedded program prompts, messages, and other text.

TAPE RECORD DISPLAY SCREENS

Display screen 1.

This screen shows the HDR1 label from an ANSI-labeled tape file. Notice that the data is encoded in ASCII, and that the label is 80 bytes

DISP	HEX DATA	ASCII
0	48 44 52 31 47 49 52 41 46 46 45 20 20 20 20 20	HDR1 GIRAFFE
16	20 20 20 20 20 59 4F 46 4F 4F 4C 30 30 30 31 30	YOFOOL00010
32	30 30 31 30 30 30 31 30 30 20 38 36 32 39 30 20	001000100 86290
48	30 30 30 30 30 20 30 30 30 30 30 20 20 20 20	00000 000000
64	20 20 20 20 20 20 20 20 20 20 20 20 20 20 20	

COMMAND: BLOCK 2 LENGTH 80 Press ESC to exit

Display screen 2.

This screen shows a data block from an ASCII text file. Notice that each line of text is ended by the bytes 0D 0A (carriage return-line feed).

DISP	HEX DATA	ASCII
0	0B 2E 74 3A 70 63 77 32 63 6F 72 6F 2E 70 72 6Dt:pcw2ccoro.prm
16	0D 0A 09 2E 77 3A 34 37 0D 0A 0C 00 0A 0D 0A 0Dw:47.....
32	0A 0D 0A 0D 0A 20 20 20 20 20 20 20 20 20 15 46 4C FLAGSTAFF
48	41 47 53 54 41 46 46 20 45 4E 47 49 4E 45 45 52	ENGINEERING, INC
64	49 4E 47 0D 0A 0D 0A 0D 0A 20 20 20 20 20 20 20
80	20 20 20 20 20 20 15 54 41 50 45 20 55 54 49 4CTAPE UTIL
96	49 54 59 0D 0A 0D 0A 0D 0A 20 20 20 20 20 20 20	ITY.....
112	20 20 20 20 20 15 55 53 45 52 27 53 20 4D 41 4EUSER'S MAN
128	55 41 4C 0D 0A 20 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A	UAL.....
144	0D 0A 0D 0A 00 0A 0D 0A 0D 0A 0D 0A 00 0A 0D 0A
160	0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 20 20 20 20
176	20 20 20 20 20 20 20 20 20 20 20 20 20 1C 43 6FCo
192	70 79 72 69 67 68 74 20 31 39 38 36 0D 0A 2C 20	pyright 1987.....
208	20 20 20 20 20 20 20 20 20 20 20 20 1C 46 4C 41 47FLAG
224	53 54 41 46 46 20 45 4E 47 49 4E 45 45 52 49 4E	STAFF ENGINEERIN
240	47 00 0A 20 20 20 20 20 20 20 20 20 20 20 20 20	G.....
256	20 20 20 20 20 20 1C 42 6F 78 20 3139 37 30	1120 KAIBAB LANE
272	0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20 20
288	1C 46 6C 61 67 73 74 61 66 66 2C 20 41 5A 20 38	.FLAGSTAFF, AZ 8
304	36 30 30 32 0D 0A 20 20 20 20 20 20 20 20 20 20	6001.....

COMMAND: BLOCK 1 LENGTH 7081 Press ESC to exit

Changing the Tape System Parameters

The change tape parameters submenu provides a way to change a number of system values. Some are hardware-related and change the way the tape drive and controller operate, while others are software parameters.

Parameters are changed by entering the right letter in the command field and pressing **(ENTER)**. If you use **(ESC)** to exit this submenu before changing a parameters, no changes are made.

-----<F10.= Help ESC.= Exit Routine>-----

- A) Set low tape speed (default)
- B) Set high tape speed
- C) Set short interrecord gap for write (default)
- D) Set long interrecord gap for write
- E) Set error retry count (default = 16)
- F) Set tape address (O = default)
- G) Set 1600 bits/inch density (default)
- H) Set 3200 bits/inch density
- I) Set maximum buffer size for read

Command:

Figure 24. Change Tape Parameter Submenu.

Each of the parameters on this menu are explained in the following sections.

Set low tape speed (default)

This option selects the low tape speed for drives that have software selectable speeds (not all drives do). Speed selection is highly drive-dependant, and this option may not be usable on your drive. On drives that have 2 software-selectable speeds, low speed is usually 25 ips (inches per second). Other smart drives have several speeds, but their speed is a function of the data transfer rate and is not selectable by this option. Refer to your drive's operating manual to determine if you can select tape speed.

Low speed is the default set by TAPEUTL at startup time.

Set high tape speed

For drives with software-selectable speeds, this option sets the high speed. This option may have no effect, since speed selection is highly drive-dependent (see above). If high speed is available, it is usually 100 ips (inches per second).

Caution! If your computer is an IBM AT or compatible (not a PC or XT), you may not be able to use high speed tape transfer. Strangely enough, the AT's highest possible data transfer rate is slower than that of the PC or XT (our testing has found the effective upper limit to be near 100 kB/sec). The data transfer rate is a function of both tape speed and tape density:

$$\text{transfer rate} = \text{density} \times \text{speed}$$

For example, at 100 ips and at a density of 1600 bpi, the data transfer rate would be 160 kB/sec, which is too fast for an AT. You would need to use low tape speed (25 ips), which would bring the transfer rate down to 40 kB/sec.

Set Short Interrecord gap for write

This option sets the default IRG, approximately 0.6 inches. The IRG setting affects tape write operations only.

Set long Interrecord gap for write

You can set your drive to write with a long IRG between blocks. The long IRG is about 1.2 inches, twice as long as the standard short IRG. The only reason to use a long IRG would be to keep the tape drive in streaming mode during write operations, where the drive is operating continuously rather than starting

and stopping between writes. With the current implementation of TAPEUTL, no increase in speed is gained with this option.

Set error retry count

This option sets the number of retries that TAPEUTL will attempt before reporting a hard tape error. The default number of retries is 16. You should increase this number if you are trying to read or write a marginal quality tape. Keep in mind that doing so will increase the time required for all transfer operations, as multiple reads and writes may be required before a good operation occurs.

To set a new number of retries, follow the option letter by the new count. For example:

e 24

will set the error retry count to 24. Typing the option letter only will display the current value of the retry count.

Set tape address

The tape address is the logical unit number of a particular tape drive, 0 through 7. This address is selected at the drive by either switches or jumpers, and the address used by TAPEUTL must correspond to this address. For most users with a drive, this address will be 0; the default set by TAPEUTL at startup.

By changing the tape address, you can use more than one tape drive in a single session with TAPEUTL. Once the address is set, it remains set until reset or until you exit TAPEUTL.

Set 1600 bits/inch density

This option sends a command to the tape drive to set the tape density for both reads and writes to 1600 bpi, the default set by TAPEUTL at startup. Note that some tape drives will not respond to a software density-select command.

Set 3200 bits/inch density

If the drive in use will operate at 3200 bpi, this option will select this higher density for both reads and writes. Otherwise, this option will have no effect.

Set maximum buffer size for read

This option sets the size of the buffer used by TAPEUTL for tape reads. The default size at startup is set to 16 kB (16,384 bytes). The maximum size selectable is 64 kB (65,536 bytes).

To set the buffer size to a new value, follow the option letter by the new buffer size—for example:

i 32000

will set the buffer size to 32,000 bytes. Typing the option letter only will display the current maximum buffer size.

If you receive a Tape block exceeds maximum error message during a tape operation, increase the buffer size with this option, then retry the operation.

The buffer size should not be set much larger than necessary, as this will slow down the average data transfer rate.

The DOS Filter Function

TAPEUTL has a data filter that can convert characters in the data stream, during both tape-to-disk and disk-to-tape transfers. The filter consists of translation statements in a file. Each statement specifies that a certain character or string of characters is to be changed to another character or string of characters each time it appears in the data stream. This file is an ASCII text file which can be created or modified by any text editor or word processor which handles ASCII files (DisplayWrite 3, Wordstar in non-document mode, EDLIN, etc.).

Initializing the DOS Filter

The filter function is set up from the main menu by choosing option E from the main menu. The following single prompt will appear on the bottom half of the screen:

Code translation file:

Simply enter the name of the file containing the filter translation statements, including any needed drive letter or pathname qualifiers. Press **(ENTER)** to submit the file to TAPEUTL. After the file is found, it will be read into TAPEUTL's workspace. If the file is read with no errors, the following message will appear:

Code translation filter initialized:

If the given file can't be found, or if the translation file contains invalid statements, an error message will appear, and you can re-enter the filename or exit this option by use of **(ESC)**.

How and When the Filter Functions

Whether the data filtering function operates before or after the other processing functions (handling line-end characters and ASCII/EBCDIC translation) depends on whether the transfer is

from disk to tape or vice versa. The following illustrations show the flow of processing for both types of transfer.

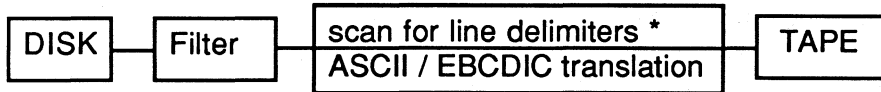


Figure 25. Disk-to-Tape Processing Flow

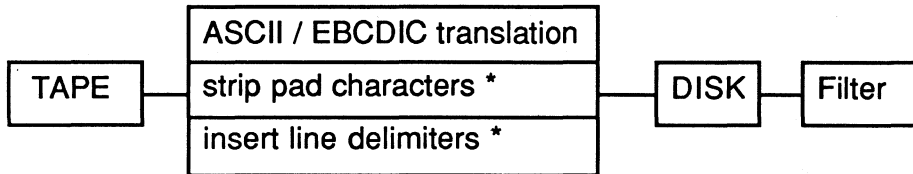


Figure 26. Tape-to-Disk Processing Flow

* —these operations do not apply to string-type data transfers

The filter function is shown in a dotted box indicating this part of the processing flow is optional.

Understanding when the filter function is applied to the data flow is extremely important to insure the proper results. You can easily get into trouble with filtering, especially if any changes are made to line-end delimiter characters or if filtering changes the length of data records.

It is safest to use filtering on string-type data, since no scanning for (or insertion of) line-end delimiters occurs. If you use variable length records, care must be taken in altering data with the filter. There are three situations in particular to be aware of:

- 1 If line-end delimiters are altered during a disk-to-tape transfer, then the new line-end characters are the ones which must be specified on the transfer screen, since the filter alters the data before the line-end processing routines.
- 2 If the filtering process has the potential for changing the length of data records, the maximum record length should be given as the new longest record length, after any filtering has been done. Otherwise, records may be incorrectly split.
- 3 If any characters which would be treated as pad characters are altered by filtering (e.g., trailing spaces), the new character must be given as the pad character to strip during the transfer operation.

Filter Translation Statements

All statements must begin in the first position on the line, and all string specifiers must be separated by at least one space. All

statements can include a comment field, indicated by square brackets below (the brackets themselves are not required), separated from the end of the statement by at least one space.

The maximum number of translation statements in a file is 2,000, not counting single character (1-to-1) statements. The maximum number of bytes in translation strings is 12,288. The command file text can be a total of 24,576 bytes long.

Each translation statement can be one of three possible types:

- simple string replacement
- toggle string replacement
- wildcard string replacement

The first two types (character strings) are indicated in one of three ways:

1. ASCII character string:

A'... ..'

The periods indicate any number (up to 64) of characters. The apostrophes are delimiters that mark the beginning and end of the string. Any character can be used for the delimiters, so long as it doesn't occur within the string itself. The delimiter character must be the first character after the A . To include an apostrophe in the string, use another character as a delimiter: A/doesn't/

2. EBCDIC character string:

E'... ..'

EBCDIC strings are specified the same as ASCII strings (above), except that the EBCDIC equivalents of the characters will be searched for or replaced in the data stream.

3. Hexadecimal byte string:

X'xx xx ... xx'

Each "xx" indicates a byte as a pair of hexadecimal digits (0-9 and A-F). The string can include up to 64 bytes of data. The delimiter must be the first character after the X and there must be one space between each pair of hex digits. This form of string specifier can be used where the desired bytes for searching or replacement cannot be written as characters in the statement itself; for example, carriage returns, tabs, line feeds, etc.

FILTER TRANSLATION STATEMENT TYPES

1. Simple String Translation Statements

This is a line in the file in the form:

```
old string new string [comment]
```

where old string and new string are each one of the string specifier forms given above. The simple replacement statement simply does a one-or-one replacement: each occurrence of old string in the data stream is replaced by new string. This statement can also be used to delete old string by simply omitting new string in the statement.

2. Toggle String Replacement Statements

The toggle statement has the form:

```
T old string new string 1 new string 2 [comment]
```

The T must be the first character on the line. Each of the 3 strings in the toggle command can be any of the 3 string specifier types given above. The toggle replacement statement does alternating replacement of character strings. When old string is first found in the data stream, it is replaced by new string 1. The next time it is found, it is replaced by new string 2. the next by new string 1, and so on. This function is useful for translating printer control codes (boldface, underline, etc.) which must be turned on at the start of some text, then turned off at the end of the text.

3. Wildcard String Replacement Statements

The wildcard statement has the form:

```
W old string new string [comment]
```

For the wildcard statement only, special string specifiers are used. Each statement starts with the character W. These are the same as the other specifiers described above, except that the question mark character has a special meaning within these strings. Whenever a ? is given in a wildcard string specifier, any character in the data stream will match at this position. The new string specifier can also contain ? characters. If present, they will be replaced by characters from the data stream at the corresponding position in the string. Using wildcard translation, you can convert strings where only part of the string must match exactly.

For example, the translation statement:

```
WA Dave ?????? A Chris??????
```

will cause the following translations:

<u>old string</u>		<u>new string</u>
Dave Johnson	→	Chris Johnson
Dave posters	→	Chris posters
Dave # 12345	→	Chris # 12345

You can use ASCII, EBCDIC or hexadecimal string specifiers with the wildcard statement. To specify a character in a hex string, use the hex equivalent of the ASCII question mark, 3F. The only limitation is that you cannot specifically search for or replace a question mark with this type of statement. The question mark character itself, however, will match if it is found in the data stream where there is a question mark in the string specifier.

Special Note on ASCII vs. EBCDIC Character Strings

When the terms "ASCII character" and "EBCDIC character" are used, these refer to characters in the data stream to or from tape—NOT characters in the translation statement file. The text in the translation file itself must be all ASCII, which is the native character set of the IBM PC.

If a string specifier statement refers to an EBCDIC "e", this means that it will match or replace with the character that corresponds to the EBCDIC "e" character. The "e" in the translation file is in ASCII, always.

Translation Statement Examples

`A/' / A' /`

This statement replaces all ASCII quotation marks in the data stream with ASCII apostrophes.

`X' 0D 0A' X' 0D 25'`

replaces the hexadecimal byte string 0D 0A (ASCII carriage return-line feed) with the byte string 0D 25 (EBCDIC CR/LF).

`TX' 02' X' 1B6531' X' 1B6530'`

alternately replaces the hexadecimal byte value 2 with the second and third byte strings (these strings are toggle printer control codes).

`WA' file.???' A' file???.XYZ'`

searches for the ASCII text string file. followed by any three (3) characters. The three characters are then moved to a new position in the string.

Special Note on the Order of Translations

The filter function first performs all multiple-character translations, then all single-character (one-to-one) translations. Therefore, it makes no difference which order any one-to-one translation statements appear in the translation file. However, the character strings to be replaced will be searched for in the order that they appear in the translation file. To ensure that the proper translations are made, you should realize that this order is important. In the same string of characters are being searched for in two translations of differing lengths, the longer string must be specified first. Otherwise, the shorter string will be found and translated first, and the longer string will never be found.

For example, if the following two statements are in the translation file:

```
A'ab' E'ab'
A'abnormal' E'unusual'
```

the ASCII string `abnormal` will never be found because the string `ab` will have already been translated to EBCDIC.

Reversing the order of these statements would allow the longer string to be found and translated. The following example shows an application of filtering, illustrating the translation file, and the input and output streams:

Translation file:

```
A'bad' A'good'
A'good' A'bad'
A'dogs' A'people'
a''' a'''
TA'flip' A'flip' A'flop'
WA'Dave ???????' A'Chris ???????'
```

Input data: Some dogs are very bad, even on good days. Dave Johnson had one once that wouldn't even fetch his paper, so he sold it to Dave Oliver (who then gave it to Dave Jones). Like most dogs, his tail would go 'flip flip flip flip flip'.

Output data: Some people are very good, even on bad days. Chris Johnson had one once that wouldn't even fetch his paper, so he sold it to Chris Oliver (who then gave it to Chris Jones). Like most people, his tail would go "flip flop flip flop flip".

Automatic Operations of TAPEUTL

TAPEUTL has a facility to allow you to create command files which can later be used to run the program in automatic mode. This feature can be used to relieve the PC user of the task of entering commands manually for repetitive tape transfer jobs. For instance, if you process a certain kind of tape regularly, you can create an automatic command file for the processing session, then simply run TAPEUTL under the control of this command file whenever those tapes are processed.

The command file is created by running TAPEUTL in a special record mode. In this mode, all commands and keystrokes entered from the keyboard are stored and written to the command file. When the program is later run under control of the command file, all these commands and keystrokes are executed in their original sequence.

The automatic-mode feature includes options which allow pauses in automatic execution, during which the user can file in fields in screens before continuing in automatic mode.

The Automatic Mode Status Marker

When you are in automatic mode, either during recording or playback, an additional status marker appears at the top left of the screen. There are four different markers:

- Rec—current session is being recorded
- Man—manual mode has been entered during either record or playback
- Auto—session is being run in automatic execution mode
- ****—error was encountered during automatic execution; reverts to manual mode.

Recording Your TAPEUTL Session

To create an automatic execution command file, give the name of a file which doesn't currently exist on the DOS command line, as shown here:

```
C:tapeutl newfile.cmd
```

where `newfile.cmd` is the name of the file where the automatic execution commands will be written (the reason the file cannot already exist is that an existing file will be taken by TAPEUTL as containing commands to run in automatic mode).

When TAPEUTL is invoked this way, every keystroke from that point is recorded and saved in the command file. Therefore, you

should plan your session carefully so that no extraneous or unnecessary command paths are taken. When TAPEUTL plays back a session, it just blindly follows your original path.

Providing for User Input in Automatic Mode

When you are recording a session, you can cause the automatic execution to be temporarily suspended, to allow the user to enter information in screens (for example, file names). You do this by pressing **(F3)** at the point where you want automatic mode to be suspended. When you do this in record mode, the automatic status marker will change to `Man`, to indicate that manual mode has been entered. During recording, you can continue filling in fields, but these entries will no longer be written to the command file. This manual mode will continue up to the point where you hit the **(ENTER)** key during both record and playback. When **(ENTER)** is pressed, the program will revert back to record mode, or back to automatic execution during playback.

When **(F3)** is used to enter manual mode, the cursor position at that point is saved. When the command file is executed later, this becomes the initial cursor position during playback. You can use this fact to position the cursor to any desired field when recording your session.

Ending Automatic Execution Recording

There are two ways to end the recording of a session:

- by normal termination of TAPEUTL (using **(ESC)** to exit back to DOS from the main menu). This closes the command file, and will cause an exit to DOS at the end when the command file is later used to run TAPEUTL.
- by using the **(F4)** key during record mode. This closes the command file, and will cause TAPEUTL to enter manual mode (as indicated by the marker `Man`) during later automatic execution. At this point, the user has complete control, just as under normal program operation.

Running TAPEUTL from a Command File

To cause automatic execution of TAPEUTL, simply enter the name of a previously recorded command file on the DOS command line:

```
C:tapeut1 autofile.cmd
```

This command will cause TAPEUTL to be run under the control of the keystrokes stored in the given command file. The status marker `Auto`, will appear at the upper left, and you will see screens flash by as the stored commands are executed. The program may stop during automatic execution for one of two reasons:

- if the marker `Man` appears at the upper left, that means that manual mode was entered when the command file was recorded, either temporarily (using **F3**), or permanently (using **F4**). During playback, you now have control and must fill in the necessary fields and press **ENTER** to continue. If you still have control after **ENTER** was pressed, then the original session was ended with **F4** and you are now permanently in manual mode. Otherwise, control will revert back to automatic execution.
- if the marker `****` appears, that means that an error was encountered during the current session. Either an actual error occurred (file not found, tape error, etc.), or your session took a path that diverged from the original recorded session. In either case, TAPEUTL reverts back to manual mode permanently, and you now have control over the program from the keyboard.

ASCII Batch Command File

TAPEUTL commands may be placed in an ASCII batch file created with any text editor or generated by a program. Each command entered when running the program must be entered in the batch file with the exception of the initial **ESC** for paging past the initial copyright screen (the copyright screen is automatically skipped when running from a batch file).

Special keys such as **TAB**, **ESC**, **F1** through **F10**, **PGUP**, **PGDN**, etc. may be assigned by entering the key code preceded and followed by tilde sign (~). As an example, the escape key may be entered as `~ESC~`. Either upper or lower case letters may be used.

The following special key codes are recognized:

Chapter 4—9-Track Tape Applications

ESC	Escape key
TAB	Tab key (right tab)
STAB	Shifted tab (left tab)
F1 - F10	Function keys
SF1 - SF10	Shifted function keys
CF1 - CF10	Control function keys
AF1 - AF10	Alternate function keys
A0 - A9	Alternate numeric keys
A-	Alternate hyphen
A=	Alternate equal
Ax	Alternate any key (x any letter key)
UPAR	Up arrow
DNAR	Down arrow
LFAR	Left arrow
RTAR	Right arrow
HOME	Home key
END	End key
PgUp	Page up
PgDn	Page down
INS	Insert
DEL	Delete
CLAR	Control left arrow
CRAR	Control right arrow
CPGU	Control page up
CPGD	Control page down
CHOM	Control home
CEND	Control end

The old batch routines for TAPEUTL are still implemented. When the program is initiated and a batch file name is entered on the command line, it is processed as before. If the file does not exist, the program goes into automatic mode and tries to execute a file which was created before learn mode. If it does not exist, the program goes into auto mode and tries to execute a file which was created before in the learn mode. To execute an ASCII batch file, the file name must be preceded by an equal sign, such as:

```
C : >TAPEUTL =FILE.ASC
```

Tape Data Extraction

Tape Data Extraction is designed to extract data from 1/2-inch magnetic tape files and transfer the data to an IBM PC file.

Data is extracted and transferred field-by-field from tape files with fixed length records.

Formats supported are ANSI unlabeled and IBM labeled tape. Various types of conversions may be performed on fields from the input tape file. Field conversions are specified by the operator and then stored in a control file for use by the conversion program. TDE provides the following types of data manipulation:

Character fields

- ASCII-to-EBCDIC conversion
- EBCDIC-to-ASCII conversion
- User specified translation

Numeric fields

- Input fields may be converted from or to any one of eight different numeric formats.

Field Insertion

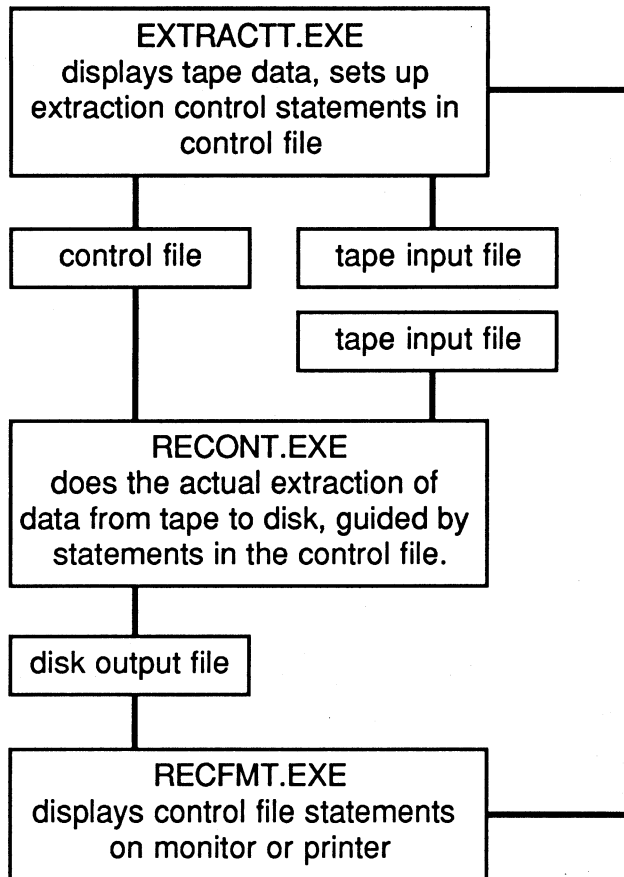
- Literal character strings or numeric fields may be inserted into the output records.

Input record select/reject

- Input records may be selected or rejected based on comparisons made on input record fields.

How Programs in This Package Work Together

This package contains several separate programs plus a batch file called TDE.BAT which can be used to run the data extraction programs as a single jobstream. The following figure shows schematically how these various programs are related:



The batch file TDE.BAT runs both EXTRACTT and RECONT in sequence for one-of-a-kind conversions. If a standard conversion needs to be run repeatedly the control statement file created by EXTRACTT can be reused and only RECONT need be run each time.

System Requirements

Use of the TAPE DATA EXTRACTION programs require the following minimum system configuration:

- 1 IBM PC/XT/AT with a minimum of 512 kB of RAM. All available memory is used as a work buffer during the conversion process which results in faster conversion times on machines equipped with larger amounts of memory. A fixed disk is highly recommended. TAPE DATA EXTRACTION may be used

on diskette-only machines but normal output file sizes may exceed the capacity of standard diskettes. Multi-volume diskette output is supported but since read/write times to diskette are much longer than to fixed disk a substantial loss in speed will occur.

- 2 An Overland Data controller card and cable.
- 3 9-track tape drive (Pertec/Cipher standard).

Program Descriptions

TAPE DATA EXTRACTION consists of the following programs and files:

EXTRACTT.EXE
RECONT.EXE
TDE.BAT
RECFMT.EXE
INSUTIL.EXE

EXTRACTT.EXE

This is a full screen menu driven editor that allows the user to specify the conversions to be specified for a tape data file. The program stores the conversion parameters in a control statement file on disk or diskette.

RECONT.EXE

This program performs the actual file conversion and transfer. The program requires a control statement file specifying the field conversions to be performed.

TDE.BAT

This batch file will execute the programs EXTRACTT.EXE and RECONT.EXE in sequence. This allows the operator to enter the conversion parameters for a tape file and then automatically perform the conversion on the file without loading another program. TDE.BAT is an ASCII text file and may be modified or created by any simple text editor.

RECFMT.EXE

This program will display the conversion parameters contained in a control statement file created by EXTRACTT.

INSUTIL.EXE

This program is used to set screen display colors for EXTRACTT.EXE.

EXTRACTT.EXE Program

EXTRACTT allows you to examine tape file records define fields for data transfer and creates a control statement file containing conversion commands. The control statement file will be used by the RECONT program at conversion time to guide the tape-to disk transfer operation.

The name of a control statement file to be created or an existing control statement file may be indicated when EXTRACTT is run. A maximum of 250 statements may be edited with EXTRACTT. The control statement file name should be entered after the program name to use this option. Entering the following line at the DOS system prompt would cause EXTRACTT to use a control statement file named CONPARM.DAT:

```
C extractt conparm.dat
```

In the above example EXTRACTT would search for a DOS file named CONPARM.DAT. If the file is found the control statements in the file will be read into memory for editing or modification. If the control file does not exist it will be created by the EXTRACTT program.

If a control file name is not given at program load time the file name will be requested on the first screen of the program.

General Features

The EXTRACTT program provides the user with a complete set of editing screens that allow the user to create a control statement file that will support most conversion requirements.

EDITING SCREENS

INITIAL—Initial program screen: specify tape file and record type.

MAIN—Display tape file records and select editing subscreens.

SELECT—Select or reject records by field comparison.

MOVE—Move a field from input record to output record.

NUMERIC—Translate numeric input field to a numeric output field.

FILL—Fill a field with ASCII EBCDIC or hexadecimal values.

LITERAL—Insert ASCII EBCDIC or hexadecimal values in output records.

FINAL—Specify output file name and record type and length.

HELP—Numeric field specifiers.

The **(ESC)** key may be pressed on any screen to return to the next higher level without performing any of the screen

functions. If **(ESC)** is pressed on the FINAL screen no control file will be written.

On each screen the cursor may be moved from field to field using the **(TAB)** key or **(SHIFT)+(TAB)** to move in reverse. After all screen prompts have been filled in with data the data is submitted to the program by pressing the **(ENTER)** key.

Screen Description—INITIAL

This screen is displayed immediately after program load. The screen is used to specify the tape input file to be converted the control statement file name that will contain the conversion parameters for the tape file specified and tape drive address and speed setting.

Screen Entries

INPUT FILE NAME

Must be entered. The tape file is specified as follows:

- # — this will access a tape file by number as recognized by counting tape (file) marks from the beginning of the tape.
- name — this will access a labeled tape file by file name.

RECORD TYPE

This must be supplied and must be either F (fixed length records) or V (variable length records). The only exception for this prompt is when accessing labeled tape files with fixed length records. The record type and length will be read from the tape label and automatically displayed.

RECORD LENGTH

Must be entered and must be between 1 and 9999. The only exception is for labeled tapes with fixed length records.

TAPE SPEED HIGH/LOW

Optional. Default is L (usually 25 ips depending on the tape drive). High speed (100 ips) may be indicated by entering an H. High speed may not be supported on your drive.

ADDRESS

Optional. Default tape drive address is zero (0). May be any number between 0 and 7.

COMMAND FILE NAME

Required. May be any desired DOS file name. The file name must be fully qualified including any needed drive letters.

Labeled Tape Notes:

When accessing labeled tapes with fixed length records in the input file only the input file name need be entered: The screen will display the record type and length. At this point the values may be accepted as is or the values obtained from the tape label may be overridden and then entered.

Existing Control Statement Files:

When accessing an existing control file this screen will be filled in from the control file. Press **(ENTER)** to accept all entries as is or type over any desired fields then press **(ENTER)**.

Screen Description—MAIN

This screen is displayed after the INITIAL screen has been completed.

This screen is the main editing screen for creating conversion control statements that will be written to the control statement file. The screen supports a field edit function that allows the operator to define specific fields for conversion.

Screen Divisions

The screen is divided into an upper and lower section. The upper section of the screen is the record display area. The lower section of the screen is the function and edit menu.

Upper Screen

The first data record from the tape input file will be displayed on the upper half of the screen. The Input file may be scrolled through record by record. Both forward and reverse scrolling is possible using **(PGDN)** to scroll forward and **(PGUP)** to reverse scroll through the file.

Data records may be displayed as ASCII or EBCDIC characters. The data display may be toggled between ASCII and EBCDIC by pressing **(F5)**.

Data fields in the input records must be defined for conversion. Several edit functions are provided to accomplish this.

A field is defined by a beginning and ending field position in the record. The beginning and ending field position must be marked to indicate a valid field.

Field Marking

To mark fields for conversion press the **(F9)** to enter edit mode. The cursor will move to the file record displayed at the top of the screen. The cursor may be moved through the data bytes of the record by using the arrow keys (note that (spacebar) = **(→)**). This allows positions in the input record to be selected by pressing **(F1)** to mark the beginning and end of a field to be moved translated or used for a select/reject statement. The **(F2)** key is used to remove a mark that is under the cursor. To remove all marks press **(CTRL)+(F2)**. To return to the bottom of the screen after a field has been marked press **(F9)** again. Use **(ESC)** to remove marks and return.

Each marked field requires a control statement for the specific conversion to be performed on that field. Note that the markers for selecting data from the input record MUST be set BEFORE going to the specific routines which create data conversion statements. The statement number of the control statement being created is displayed in the upper left corner of the bottom portion of each of the screens). After a field has been marked select a conversion for the field by entering the number for the desired type of control statement to be created (1 - 5). The program will display the statement creation screen selected and wait for input. After all screen entries have been completed press **(ENTER)** to save the control statement. The program will re-display the MAIN screen and display the next statement number to be created.

To edit existing control statements enter an E followed immediately by the statement number to display a specific statement (example: E5 will display control statement number 5). The statement and the appropriate statement creation screen will be displayed. The next statement occurring before or after the current statement may be displayed by pressing **(END)** or **(HOME)**, or **(ESC)** may be used to return to the MAIN menu.

Once a statement has been displayed in edit mode it may be altered by typing new entries over the existing entries and pressing **(ENTER)**. The statement may be deleted by pressing

(ALT)+D. The program will then display the next control statement.

Control statements are generally created in the same sequential order as the fields being defined for conversion. New statements are automatically added to the end of the control statement file. Control statements may be inserted rather than added if desired. To insert a new control statement between existing control statements display the record which follows the record to be inserted. Press **(ESC)** then enter an option of "1" followed by the type of statement you wish to insert.

For example to insert a select/reject statement preceding statement number 12 enter E12 (display statement 12). Press **(ESC)** and then enter 11 (insert statement type 1).

Control Statement File Sequence:

In general all select/reject control statements should be entered first. If the remaining statements are created in the same sequential order in which they are to appear in the output record the program will keep track of the next available output record location. The program will pre-fill the beginning and ending field position specifiers for the statement creation screens being used.

After all field conversion specifications have been defined for output records enter option "6" to exit the editing functions. The program will display the FINAL screen.

Screen Description—FILL

This screen is used to create fields in the output record that will be titled with a specific character. The fill character may be specified as ASCII EBCDIC or hexadecimal.

Screen Entries

FILL CHARACTER

Entry is required. This is the character to fill the output field with. If it is being specified in hex then it must be two valid hexadecimal digits.

ASCII/EBCDIC/HEX

This entry indicates the character type. The entry defaults to ASCII. If specified the character type must be one of the following:

Chapter 4—9-Track Tape Applications

- A ASCII character
- E EBCDIC character
- X Hexadecimal value.

BEGINNING LOCATION

Must be specified. This entry specifies the beginning byte position in the output record that will be filled. The program will pre-fill this entry with the next available output position.

ENDING LOCATION

Must be specified. This entry specifies the ending byte position in the output record that will be filled.

Screen Description—INSERT

This screen is used to create a literal field in the output record . The character string may be specified as ASCII EBCDIC or hexadecimal.

Screen Entries

ASCII/EBCDIC/HEX

This indicates the character type. The entry defaults to ASCII . If specified the character type must be one of the following:

- A ASCII character
- E EBCDIC character
- X Hexadecimal value.

BEGINNING LOCATION IN OUTPUT -

This entry must be specified. The program will pre-fill this entry with the next available output position.

DATA

One or more characters must be entered. If the string type is A or E the string must begin and end with a delimiter character which does not appear in the actual string. For example the entry HELLO/ indicates the string HELLO with slashes used as delimiters.

If a Hexadecimal string type is specified the data must be entered as a series of two hex digit bytes separated by single spaces.

Screen Description—MOVE

This screen is for moving a field from the input record to the output record and optionally translating it. Note below that the user may optionally supply the translate table.

Input Location Start

Entry must be specified. This will be filled in with the location of the first marker position if the markers are set before this routine is entered.

LENGTH

Entry must be specified. This will be calculated from the positions of the first and second markers if the markers are set before this routine is entered.

ASCII/EBCDIC

This defaults to no translation. If specified the translation type must be one of the following:

- A ASCII to EBCDIC translation
- E EBCDIC to ASCII translation

OUTPUT LOCATION START

This entry indicates the starting position in the output record where the data is to be moved to. The program will pre-fill this with the next available output record position.

NAME

This entry may be used to assign a descriptive name or comment to the field being moved. The name entered will be displayed if the command statement is later edited. The field name will also be displayed by the program RECFMT. The name entered is stored in the control statement being created and will not appear in the actual output file.

Screen Description—SELECT

This is used to specify parameters for selecting or rejecting records from the input file. Note that if more than one of these statements is generated, the ORDER they are specified in is important. Once a SELECT condition is true the record is accepted without looking at following select/reject statements.

Once a REJECT condition is true the record is rejected without looking at any following select/reject statements. Records which are not specifically selected or rejected are processed.

Any record which is neither selected or rejected will be processed if at least one reject statement has been entered. If there are one or more select statements and no reject statements all records not specifically selected will be rejected.

Select/Reject

An entry is required.

S select record.

R reject record.

Records will be selected or rejected when the specified screen conditions are true.

Compare

Indicates type of compare operation for selection or rejection of records. Entries must be one of the following:

EQ equal

NE not equal

GT greater than

LT less than

GE greater than or equal

LE less than or equal

In these comparisons the record data (first operand) is compared to a data constant (second operand).

Field

Specifies the type of field to compare. Valid entries are:

A/E/X/Z/L/T/C/G/P/B/R.

A ASCII text (assumed if no entry)

E EBCDIC text

X Hexadecimal data is supplied.

The remaining indicators are numeric field specifiers. See the Help Screen description, page 168, for the list of numeric field types supported.

Compare Data

One or more bytes must be entered in the format specified above. If the string type is A or E then this must be a string of characters beginning and ending with a delimiter character. The delimiter character must not appear within the string.

Hexadecimal character must be entered as two hex digits per byte with one space between each byte and the next. If a numeric compare is specified enter the value to be compared to using a leading minus (-) if required.

Beginning Position in Input Record

Must be specified. The program will fill this in from the position of the first marker if it was set prior to entering this routine.

Numeric Field Length

Entry must be specified for numeric compares. This will be filled in from the field markers if the markers are set before this routine is entered.

Numeric Field Translation

This entry specifies a text translation of the input numeric field before it is evaluated. This is not normally necessary but is NEVER VALID for packed decimal or binary fields. If entry is specified the indicator must be either:

- A ASCII to EBCDIC translation
- E EBCDIC to ASCII translation

Screen Description—NUMERIC

This screen is for specifying conversion of a numeric field from input to output. To see a list of the types of numeric fields supported press F10 to toggle between the numeric field help screen and displaying the data record.

Input Type of Field

Must be one of the eight valid numeric field types. See field type description.

Input Beginning Position

This is pre-filled with the position of the first marker if the marker was set before entering this screen.

Input Field Length

This entry is pre-filled by the program based on the position of the first and second markers if they were set before entering this screen.

Input Translation

This entry is optional and not normally required. If specified it causes a CHARACTER translation to be performed on the input

field before numeric evaluation. If the input is specified as packed decimal or binary no translation is valid. Valid indicators are:

- E translate from EBCDIC to ASCII
- A translate from ASCII TO EBCDIC.

Output Type of Field

Must be one of the eight valid numeric field types. See field type description.

Output Beginning Position

This is pre-filled by the program with the next available output record position.

Output Field Length

The length specified must be sufficient to contain the converted values and varies depending on the type of field specified. If **(ENTER)** is pressed without entering this field the program will calculate the required field length based on the type and length of the input field and the type of output field specified. The calculated value will be displayed and may be accepted by pressing **(ENTER)** again or the value may be changed before pressing **(ENTER)**.

Output Translation

This is optional and not normally needed. If specified a CHARACTER translation will be performed on the output field after it has been generated by the numeric conversion. Translation is never valid for packed decimal or binary output fields. Valid indicators are:

- E translate from EBCDIC to ASCII
- A translate from ASCII TO EBCDIC.

Once a numeric new conversion statement has been generated the program will return to this screen. The entries on the screen will be pre-filled to convert the next sequential bytes of the input record in the same length and type as the previous command statement. This is done because numeric fields often occur in several sets in a record. To accept the pre-filled entries press **(ENTER)** to generate the next statement. The entries may be modified on the screen to generate a command statement with new parameters or the **(ESC)** key may be pressed to return to the main menu.

Name

This entry may be used to assign a descriptive name or comment to the field being converted. The name entered will be displayed if the command statement is later edited. The field name will also be displayed by the program RECFMT. The name entered is stored in the control statement being created and will not appear in the actual output me.

Screen Description—FINAL

This screen is entered when option six is chosen from the main screen.

Output Data File

Specifies the output DOS file to be generated by the conversion.

Record Type

This entry is required and must be one of the following:

F for fixed length records.

V for variable length records.

If V is specified the program adds a carriage return and line feed (hexadecimal 0D 0A) to the end of each output record.

Record Length

This entry is required. The program pre-fills this field with the highest output record location previously specified by any of the conversion statements generated.

Skip

This entry is optional. Enter the number of records to skip before starting to process the file.

Process

This entry is optional. Enter the maximum number of records to process from the input file.

Speed

This entry is optional. Specify L or H to force the tape drive to low or high speed.

Address

This entry is optional. For tape input files only. Specify a tape unit address from 0 to 7. If not entered it defaults to address 0.

ASCII to EBCDIC Table

This entry is optional. If entered the conversion program reads the first 256 bytes of the file (table) indicated and uses the data for any requested ASCII to EBCDIC conversions.

EBCDIC to ASCII Table

This entry is optional. If entered the conversion program reads the first 256 bytes of the file (table) indicated and uses the data for any requested EBCDIC to ASCII conversions.

Sample

This entry is optional. If entered this entry indicates a record sample rate. For example if a five is entered every fifth record will be processed rather than all records from the input file.

Screen Description—HELP

(Numeric Field Types)

The help screen explains the numeric field indicators used by the program. A short field description appears on the screen for each indicator.

The following types of numeric fields are supported. The letter code to the left must be used in screen entries to indicate a particular field type. When specified as input these formats are expected by the program. When specified as output these formats will be generated by the program.

Z

Zoned decimal. EBCDIC characters with a possible zone over the low order digit to specify the sign. A hex D zone specifies a negative number. The normal positive zone is hex C or hex F . The maximum length is fifteen (15) digits.

L

ASCII numeric characters with a possible leading + or - sign. The sign character may be preceded by blanks in the input field and the lack of a (-) sign is taken to mean a positive number. On output the first position of the field will be either (+) or (-) and the digit positions will be left zero filled. The maximum total length is sixteen (16) bytes.

T

ASCII numeric characters with a possible trailing + or - sign. The lack of a (-) sign is taken to mean a positive number. On output the last position of the field will either be (+) or (-) and the digit positions will be left zero filled. The maximum total length is sixteen (16) bytes.

C

ASCII numeric characters with a possible trailing CR in the last two positions for negative numbers. The lack of a trailing CR is taken to mean a positive number. On output the last two positions will be either blank or CR. The maximum total length is seventeen (17) bytes.

G

ASCII numeric characters with a possible trailing DB in the last two positions for negative numbers. The lack of a trailing DB is taken to mean a positive number. On output the last two positions will be either blank or DB. The maximum total length is seventeen (17) bytes.

P

Packed decimal. One decimal digit per nibble with a sign in the lowest order nibble. The negative sign is a hex D and the positive sign is normally a hex C or F. The maximum length is 8 bytes which is fifteen (15) decimal digits.

B

Binary with the most significant byte first and least significant byte last. The maximum length is eight (8) bytes.

R

Binary with the least significant byte first and the most significant byte last. This is the most common format for binary fields in microcomputers. The maximum length is eight (8) bytes.

RECONT.EXE Program

Once a control file has been generated by EXTRACTT the control file may be submitted directly to RECONT by entering the control file name after RECONT on the DOS command line.

The input and output file specifications in the control statement file may be overridden while still using the remainder of the statements in the control file. To do this just enter input and output file names on the command line after the control file name. For example:

```
C:>recont sample.ct1 c:\newinput.dat a:newout.dat
will execute the program RECONT using the control statements
in the file SAMPLE.CTL. However regardless of the files named
as input and output files the input data will be taken from
C:NEWINPUT.DAT and the output created will be
A:NEWOUT.DAT.
```

Error Statements

If an error is detected in an input control statement then the following message is displayed and the program is terminated.

```
nn ERROR ON CONTROL STATEMENT. SEE
DOCUMENTATION. PROGRAM ABORTED.
```

Error Messages Listed by Number

- 01—The second position of a translation table statement is not A or E.
- 02—The first statement is not a file definition statement.
- 03—INSERT. No trailing delimiter was found within 60 characters.
- 04—Unused.
- 05—The input file type is not F or V .
- 06—The output file type is not F or V.
- 07—The input file record length is invalid.
- 08—The output file record length is invalid.

Input Field Errors

- 09—An input field translation has not been specified with a blank A or E.
- 10—The input field type was not recognized.
- 11—The input field length was too long.
- 12—The the input field length was zero.
- 13—The input field extends beyond the end of the record.

Output Field Errors

- 14—The output field translation is not blank A or E .
- 15—The output field type was not recognized.
- 16—The output field length was too long.
- 17—The output field length was zero.

18—The output field extends beyond the end of the record.

Field Conversion Errors

19—FILL field translation is not blank or A.

20—FILL field extends beyond the end of the record.

21—INSERT field translation indicator is not blank or A .

RECONT.EXE

21—INSERT field translation indicator is not blank or A .

22—INSERT field character string extends beyond the end of the record.

23—MOVE field translation indicator is not blank A or E.

24—MOVE input field extends beyond the end of the record.

25—MOVE output field extends beyond the end of the record.

26—FILL hexadecimal field translation indicator is not a blank.

27—FILL hexadecimal field extends beyond the end of the record.

28—SELECT/REJECT compare type is not EQ/NE/GT/LT/GE/LE

29—SELECT/REJECT string does not (completely) fall within the specified input record.

30—SELECT/REJECT field length is invalid.

31—SELECT/REJECT string type is not E X blank or numeric.

32—SELECT/REJECT string contains invalid characters.

33-36—Unused.

37—FINAL skip or process count is invalid.

38—SELECT/REJECT numeric pre-translation is not AVE/blank.

TDE.BAT

The batch file TDE.BAT executes both the EXTRACTT and RECONT programs so that a control file may be built and executed in one step. When using this batch program the name of the control file must be entered immediately after the program name:

```
C>TDE SAMPLE.CTL
```

Once a control file has been built for converting a particular record format it need not be created again. It may be submitted directly to RECONT to convert another file with the same record format. (See RECONT program operation section.)

RECFMT.EXE Program

This program displays the conversion parameters contained in control statement files created by the EXTRACTT. EXE program. This display provides a complete field description of the records output by EXTRACTT. The display may be used to verify correct

processing of an input file or used to obtain an output record layout. The output record layout is usually required by an operator who must process the output file.

The name of the control file to be displayed must be entered on the DOS command line following the program name when RECFMT is loaded.

An output device must also be indicated on the DOS command line when RECFMT is loaded. If a device is not specified output defaults to LPT1 (the printer). To direct output to the display screen specify CON as the device parameter.

DOS Device Names for Use With RECFMT:

CON—console (display monitor)

PRN—printer (default address)

LPTn—printer n (1 st printer is the same as PRN).

Contrary to what IBM's manuals state, it is not necessary to use a colon after the device name (such as PRN:).

Run RECFMT by typing its name at the DOS command prompt followed by the command file name and output device parameters:

```
C>recfmt control.dat con
```

The above command displays the statements in control file CONTROL.DAT to the display screen. RECFMT outputs a line-by-line description of the control file command statements.

Each statement from the file is displayed in the following format:

Statement number.
Input field beginning position
Input field ending position
Output field beginning position
Output field ending position
Operation performed (MOVE FILL etc.)
Field name assigned by EXTRACTT or user.

INSUTIL.EXE

(COLOR AND PARAMETERS INSTALLATION PROGRAM)

This program allows the user to change screen colors and parameters used by the Flagstaff Engineering TAPEUTL or EXTRACTD programs. In response to the request for NAME OF PROGRAM, enter the full program name, for example, TAPEUTL.EXE . Colors are specified for three types of display attributes:

- NORMAL used for displaying most prompts, text, etc.
- ENHANCED used for operator entry fields and error messages
- REVERSED used to highlight selected data (EXTRACT only)

When a new color is requested, pressing the return leaves it unchanged. The following color values should be used to indicate the desired display colors:

FLAGSTAFF ENGINEERING - INSTALL PROGRAM OPTIONS

Program name: TAPEUTL.EXE

Screen display colors	Foreground	
0= Black	8= Dark Grey	0 x x x x x x x x x x x x x x x
1= Blue	9= Light Blue	1 x x x x x x x x x x x x x x x
2= Green	A= Light Green	2 x x x x x x x x x x x x x x x
3= Cyan	B= Light Cyan	3 x x x x x x x x x x x x x x x
4= Red	C= Light Red	4 x x x x x x x x x x x x x x x
5= Magenta	D= Light Magenta	5 x x x x x x x x x x x x x x x
6= Brown	E= Yellow	6 x x x x x x x x x x x x x x x
7= Light Grey	F= Bright White	7 x x x x x x x x x x x x x x x
Normal	Background 1 Foreground 7	8 x x x x x x x x x x x x x x x
Enhanced	Background 1 Foreground F	9 x x x x x x x x x x x x x x x
Reversed	Background 7 Foreground 1	A x x x x x x x x x x x x x x x
		B x x x x x x x x x x x x x x x
		C x x x x x x x x x x x x x x x
		D x x x x x x x x x x x x x x x
		E x x x x x x x x x x x x x x x
		F x x x x x x x x x x x x x x x

Press <ESC> to exit without update

Color Select Screen

After colors and parameters have been chosen, the same screen will display lines using the colors selected in this manner:

This is a normal line (x, the color you selected)

This is a bright line (x, the color you selected)

This is a reverse line (x, the color you selected)

You can also set default parameters of your choosing with Tape Utility. The options chosen from this screen will be the default values that the program will use to pre-fill these fields whenever TAPEUTL is run.

```
FLAGSTAFF ENGINEERING - INSTALL PROGRAM OPTIONS

Program name: TAPEUTL.EXE

System parameters -----
Read buffer length 16384   Tape retries 15   Tape address 0

Tape to disk defaults -----
Translate data? (N=no, A=EBCDIC to ASCII, E= ASCII to EBCDIC ) N
Data type: (F)ixed, (V)ariable, (I)BM variable, (S)tring S
Record length 0   Strip pad characters (Y/N) N   Insert line end (Y/N) N
Pad character Line end characters 0D 0A

Disk to tape defaults -----

Translate data? (N=no, A=EBCDIC to ASCII, E- ASCII to EBCDIC ) N
Data type: (F)ixed, (V)ariable, (I)BM variable, (S)tring S
Block size 4096   Record size 128   Records per block 32
Pad character Line end characters 0D 0A

Press <ESC> to exit without update
```

Parameters Installation Screen

At the end, this screen displays verified updates were made thus:

```
Color defaults updated
Taputl defaults updated
C>
```

Tape Backup

TAR

TAR is a tape file archive program for Overland Data's TX-8, TXi-16 or XL/2 tape controller. It produces standard **TAR**-format tapes readable by other systems with a **TAR** command. **TAR** is a standard format on UNIX and XENIX systems, thus tapes produced by this MS-DOS **TAR** program are readable on most UNIX or XENIX systems.

TAR has existed since UNIX Version 7, with very little change. It is an excellent data interchange program for UNIX and XENIX systems, and now, MS-DOS systems, since it can be invoked from the command line and requires no further assistance.

Features

TAR is a command line archive program, providing a way to store many files in a single archive, which can be kept in a file or stored using an I/O device such as a tape drive. It is useful for making backups or preparing files for transfer to another system.

TAR does the following:

- Writes a set of files to tape or another file as a single archive.
- Extracts one or more files from an archive.
- Can create a "table of contents" of the files in an archive.
- Can specify the file names to access in an archive by listing them on the command line, specifying a directory hierarchy, or specifying a file that contains a list of file names (one per line).

When reading a tape archive, this version of **TAR** implements limited error handling. It automatically retries after certain kinds of errors. There are no options for changing error handling capabilities.

Options

Specify **TAR** options in the command line by keyletters. All option keyletters form the first argument to **TAR**. The arguments for those options which require arguments form the second, third, etc. arguments. The files to be manipulated are listed by name after all other arguments.

Note: When creating an archive, it is generally a bad idea to specify full path names. When you extract the files with full path names, **TAR** puts them exactly where they were taken from. A better idea is to go to the top level directory, and use relative file names, letting you extract the files to another hierarchy.

For example:

```
C:\> tar cvfb archive 10 file1 file2
```

cvfb specifies keyletters **c**, create archive; **v**, verbose mode; **f**, the archive name is the next argument and **b**, the blocking factor is the next argument (following the archive name). **c** and **v** require no arguments, so the first argument, **archive**, is the archive name and **10** is the blocking factor. If **f** and **b** were entered in reverse order (**cvbf**), then the arguments would also need to be entered in the reverse order. **file1** and **file2** are the list of files to archive.

TAR supports the following options:

- b** Specifies a blocking factor for the archive. The block size will be N x 512 bytes. Larger blocks usually run faster and let you put more data on a tape. The default blocking factor is 20 (the most commonly used value).
Requires a numerical argument (as shown above).
Note: You may specify a blocking factor from 1 to 20.
- B** When reading an archive, **TAR** reblocks as it reads. Normally, **TAR** reads each block with a single read system call. Using this option, it does multiple reads until it gets enough data to fill the specified block size. **B** can also be used to speed up the reading of tapes that were written with small blocking factors, by specifying a large blocking factor with **b** and having **TAR** read many small blocks into memory before it processes them.
No argument is needed by this option.
- c** Creates an archive from a list of files or directories. Specifying a directory name archives that directory and all the files it contains, including any subdirectories.
No argument is needed by this option.
- D** Each time that **TAR** produces a message, this option prints the record (tape block) number where the message

occurred. This is especially useful when reading damaged archives, helping pinpoint the damaged section.

No argument is needed by this option.

- f** Specifies the archive filename (the filename is a required argument for this option). If the filename used is a dash (“-”) the archive is read from the standard input or written to the standard output. If the option is not used, **TAR** writes to the 9-track tape controller.
- k** When extracting files from an archive, this keeps existing files on the destination medium intact, rather than overwriting them with the version from the archive.
No argument is needed by this option.
- m** When extracting files from an archive, this sets each file’s modified timestamp to the current time, rather than extracting each file’s modified timestamp from the archive.
No argument is needed by this option.
- t** Lists a table of contents for an existing archive. If file names are specified, it lists only files matching the specified names.
No argument is needed by this option.
- T** Allows you to specify that the file names **TAR** will operate on will be read from the file used as the argument, (with only one file name listed per line in that file). The file is a required argument for this option. If the argument is a dash (“-”) then the list is read from the standard input.
- v** This option instructs **TAR** to be verbose about the files being processed or listed. It provides additional information about each file, like size and date. Without this option, **TAR** lists only the file names.
No argument is needed by this option.
- w** This option allows you to define the time when **TAR** will run. The time is set on a 24-hour clock, so you can define any time, up to 24 hours in the future, for **TAR** to run. The argument form must be two digits, followed by a colon, which must be followed by two digits. Thus, midnight is 24:00, and 1:30 a.m. is 01:30.
- W** Suppresses the ‘wait for user approval to exit keystroke’ (normally the program waits for a user keystroke before

exiting, to allow you to see what has been done; this switch tells it not to wait).

No argument is needed by this option.

- x** Extracts files from an existing archive. If file names are specified, it extracts only the files matching those names, otherwise it extracts all the files in the archive.

No argument is needed by this option.

Note: **TAR** will read and write multi-volume tape sets.

Warnings

In order to be compatible with UNIX and XENIX, **TAR** writes file path names using forward slash notation (/), instead of the DOS convention of using a back slash (\). With this **TAR** program you may specify path names using either notation during a create, but you must use the forward slash to extract files.

TAR does not convert MS-DOS style text files (carriage-return line-feed delimited text lines) to UNIX style text files (line-feed delimited). **TAR** also does not convert UNIX style text files to MS-DOS style text files.

TAR converts UNIX-style file names which are illegal under DOS to a legal form, naming such files T\$nnnn.GEN, where nnnn is a four digit number starting with 0000 and running through 9999.

Examples

The following command writes a **TAR** tape to the default device (the TX-8 controller).

```
C:\ tar cv .
```

Options **c** means create, and **v** means verbose mode. The dot or period (.) identifies all the files and directories in the current directory as the files to be archived.

The next command creates an archive of all files and directories under directories **usr** and **bin**.

```
C:\ tar cv usr bin
```

This specifies the relative path name for **usr** and **bin** instead of using a full (absolute) path name (such as **\usr** or **\bin**).

To see a table of contents listing of the archive created in the previous command, we use the following command.

```
C:\ tar Dtv
```

Option **D** tells us the exact block number where each file starts, **t** provides the listing, and **v** means verbose mode.

Alternatively, the listing could be stored in a separate file, one file per line, with the following command:

```
C:\ tar t > filelist
```

This places the table of contents in the file **filelist**, using the DOS redirection symbol **>**.

To extract files from an archive, use option **x**. The following example extracts all the files from the default archive.

```
C:\ tar xv
```

Option **x** extracts the files and **v** means verbose mode.

The following command creates an archive using the installed device driver, **mt0**.

```
C:\ tar cvfbT mt0 30 filelist
```

Option **c** means create the archive, **v** is verbose mode, **f** specifies that the next argument is the archive device (since this is the first option requiring an argument, the first argument, the archive device name, is **mt0**), **b** means the next argument is the blocking factor (since this is the second option requiring an argument, the second argument, the blocking factor, is the numerical value **30**) and **T** means the next argument (after the blocking factor) names a file, **filelist**, containing a list of file names to archive.

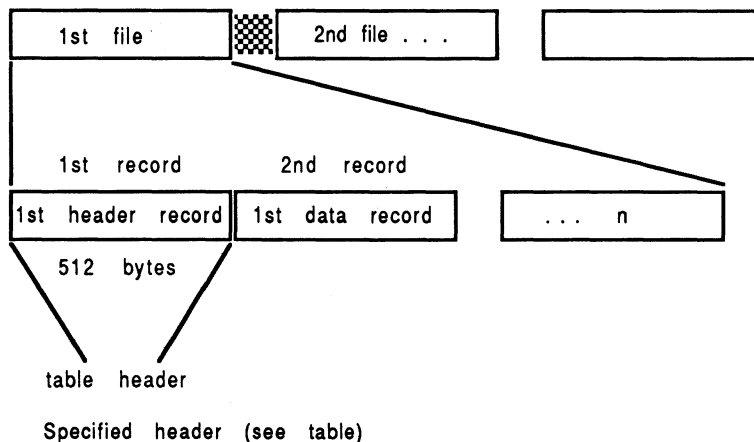
Note: Check that you have installed the device driver before using this command; **TAR** will otherwise write the archive to a simple file called **mt0**.

Archive Format

This table shows the fields for this version of **TAR**, the number of bytes each field can contain and whether DOS uses the field. The name is in ASCII, all other header fields are octal numbers represented with ASCII characters.

Name	100 bytes	Used by DOS
File Protection Mode	8 bytes	Used by DOS
User ID	8 bytes	Not used by DOS
Group ID	8 bytes	Not used by DOS
File Size	12 bytes	Used by DOS
Modification Time	12 bytes	Used by DOS
Check Sum	8 bytes	Used by DOS
Link Flag	1 byte	Not used by DOS
Link Name	100 bytes	Not used by DOS

Files concatenated are all 512 byte records (may be blocked differently from tape records).



Limitations

Tar continues to scan all files on tape even after the requested files have been found. This is normal operation, and is necessary, since you can append to a backup—a later version of the file might exist further down the tape.

Controller Specifications

TX-8 and XL/2 Specifications

TX-8 and XL/2 Input Output Map

Default base port address as shipped (hex):

TX-8 and XL/2 Input Port Map	
Offset From Base Port	Port Function
BASE+0	Tape Status Byte A
BASE+1	Tape Status Byte B
BASE+2	Tape/TX Error Status
BASE+3	Tape Read Data
BASE+4	TX Cache Count Read (Low Byte)
BASE+5	TX Cache Count Read (High Byte)
BASE+6	Reserved
BASE+7	Reserved

TX-8 and XL/2 Output Port Map	
Offset From Base Port	Port Function
BASE+0	Tape Command Byte
BASE+1	Tape Control Byte
BASE+2	TX Control Byte
BASE+3	TX Extended Control Byte
BASE+4	TX Cache Count Set (Low Byte)
BASE+5	Tape Write Data
BASE+6	TX Cache Count Set (High Byte)
BASE+7	TX Adder Offset Set

TX-8 and XL/2 Status Input Bit Map

Bit	Signal	Latched?
PORT 0—TAPE STATUS A		
0	FBY	no
1	DBY	no
2	RDY	no
3	FPT	no
4	ONL	no
5	SPD	no
6	BOT	no
7	EOT	yes
PORT 1—TAPE STATUS B		
0	RESERVED	
1	IRQ	yes
2	INRZ (low true)	no
3	IRWD (low true)	no
4	IDENT	yes
5	ERROR	yes
6	DBYT	yes
7	FMK	yes
PORT 2—TAPE/TX ERROR STATUS		
0	TOGGLE-Signature	n/a
1	RESERVED	
2	RESERVED	
3	OVERFLOW	yes
4	TAPE parity	yes
5	MEMORY parity	yes
6	CER	yes
7	HER	yes
PORT 3—TAPE READ DATA		
0-7	READ DATA	yes
PORT 4—TX CACHE COUNT LOW		

Appendix A—Controller Specifications

0-7	CACHE COUNT (Low Byte)	yes
PORT 5—TX CACHE COUNT HIGH		
0-7	CACHE COUNT (High Byte)	yes
PORT 6—RESERVED		
PORT 7—RESERVED		

TX-8 and XL/2 Output Bit Map

BASE+0 (Tape Command Byte):

This port is used to send commands to the tape drive as follows:

BASE+0 : TAPE COMMAND BYTE						
IHSP (D5)	ERASE (D4)	EDIT (D3)	WFM (D2)	WRT (D1)	REV (D0)	Function Initiated
x	0	0	0	0	0	Read 1 block fwd
x	0	0	0	0	1	Read 1 block back
x	0	1	0	0	1	Read reverse edit
x	0	0	0	1	0	Write 1 block fwd
x	0	1	0	1	0	Overwrite
x	0	0	1	1	0	Write file mark
x	1	0	0	1	0	Erase variable block
x	1	0	1	1	0	Erase fixed block
x	1	1	1	1	0	Security erase
x	1	0	0	0	0	Skip fwd 1 block
x	1	0	0	0	1	Skip back 1 block
x	1	1	0	0	0	No operation
x	1	1	1	0	0	Select 1600 bpi †
x	1	1	1	0	1	Select 3200 bpi †
0	x	x	x	x	x	Select 25 IPS †
1	x	x	x	x	x	Select 100 IPS †

† These operations are drive dependent. Check the tape drive manufacturer specifications.

The data written to the tape command port is latched and a pulse approximately 750 nanoseconds wide is produced and presented to the IGO interface line concurrently with the data. Data bits D6 and D7 are not used.

BASE+1 (Tape Control Byte):

BASE+1 : TAPE CONTROL BYTE		
0	TAD1	Drive Select LSB.
1	TAD0	Drive Select.
2	FAD	Drive Select MSB.
3	UNLD	Drive Unload.
4	FEN	Clear formatter.
5	RWD	Rewind to BOT .
6	RTH1	Read Threshold 1 (Reserved)
7	RTH2	Read Threshold 2 (Reserved)

BASE+2 (Tape Control Byte):

BASE+2 : BOARD CONTROL BYTE		
Proprietary		

BASE+3 (Pulse Decodes):

BASE+3 : PULSE DECODES		
Proprietary		
Value of 2	STATRST	Status Reset

BASE+4 (Load Cache Count, Low Byte):

BASE+4 : LOAD CACHE COUNT (LB)		
--------------------------------	--	--

BASE+5 (Tape Write Data):

BASE+5 : TAPE WRITE DATA		
--------------------------	--	--

BASE+6 (Load Cache Count, High Byte):

BASE+6 : LOAD CACHE COUNT (HB)		
--------------------------------	--	--

BASE+7 (Adder Input):

BASE+7 : ADDER INPUT		
STATRST then LOW byte, HIGH byte		

TX-8 User Options

Address

The TX-8 controller occupies eight consecutive I/O locations. An eight position dip switch (labeled on the board) allows you to address the board at any eight byte boundary within the 1024 byte address range of the IBM PC/XT/AT. Generally, addresses from 200 (hexadecimal) to 3FF (hexadecimal) are available for board addressing.

The following table shows the board addressed as shipped. Dip switch numbers 1 through 8 correspond to address lines A10 through A3, respectively, with address lines A2, A1, and A0 hardwired to logic 0. Switches set to the position represent a logic 0. Those set to the OFF position represent a logic 1. The table translates each On/Off setting to the equivalent binary. By combining the correct group of switches, the table shows the equivalent hexadecimal value.

Line	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Switch	1	2	3	4	5	6	7	8	-	-	-
Position	ON	OFF	OFF	ON	OFF	OFF	ON	ON	-	-	-
Binary	0	1	1	0	1	1	0	0	0	0	0
Hex	3			6				0			

The following diagram shows the board's address dip switch setting as shipped. Black boxes towards the top indicate position ON; that is, logic 0. Black boxes towards the bottom indicate position OFF; that is, logic 1.

The TX-8 software will automatically check for the board addressed at the default I/O port, 360 hexadecimal. If it is not detected, it will then check (in order) the following addresses:

300H 310H 320H 330H 340H 350H

You can change the I/O base port on a TX-8 controller without running `CONFIG` if the new base port you select appears in the table below.

Appendix A—Controller Specifications

DIP Switch	1	2	3	4	5	6	7	8
Address Line	A10	A9	A8	A7	A6	A5	A4	A3
360 (default)	ON	OFF	OFF	ON	OFF	OFF	ON	ON
300	ON	OFF	OFF	ON	ON	ON	ON	ON
310	ON	OFF	OFF	ON	ON	ON	OFF	ON
320	ON	OFF	OFF	ON	ON	OFF	ON	ON
330	ON	OFF	OFF	ON	ON	OFF	OFF	ON
340	ON	OFF	OFF	ON	OFF	ON	ON	ON
350	ON	OFF	OFF	ON	OFF	ON	OFF	ON

Set the DIP switch (located in the upper right corner of the board) as shown, and the I/O base port reconfiguration is complete. Normally, the software running with the controller would also have to be reconfigured, but the supplied utility programs (as well as `mt0` and **ALLTAPE**) include an automatic configuration routine which looks for the TX-8 controller at each listed address in turn, and configures accordingly if the board is found.

The auto-configure feature is activated by selecting a base I/O address of 0 in `CONFIG` (this is the factory default).

Note: The TX-8 I/O address auto-configure feature is available on program versions 2.0 and above only, and is not available for `FLASHBAK`.

Unlisted base port addresses require you to determine the corresponding DIP switch settings, change the switches, and run `CONFIG`.

If the default address of 360 hexadecimal conflicts with another device in your PC, set the controller's I/O address dip switch to one of the above settings and the software will automatically detect the new I/O address.

Interrupt Level

Interrupts may be generated upon completion of any data transfer command. The current MS-DOS software does not require interrupts (the default setting uses the timer tick interrupt, IRQ 8). To enable interrupts, shunt one of the following pin pairs. Interrupts will be generated only when Data Busy goes false. Writing to the command port will reset this interrupt.

Appendix A—Controller Specifications

Interrupt Vector	Pin
IRQ3	A (3)
IRQ4	B (4)
IRQ5	C (5)
IRQ6	D (6)
IRQ7	E (7)
IRQ9	F (9)
IRQ10	H (10)
IRQ11	J (11)
IRQ12	K (12)
IRQ14	L (14)
IRQ15	M (15)

Interrupt pins are labeled either with the above Letter (such as A), or simply with the Interrupt Vector Number (such as 3).

XL/2 User Options

Configuration of the XL/2 Tape Controller is simple and easy.

There are no switches to set or jumpers to move. Once the card is inserted correctly into the computer, the software supplied on the IBM Reference Diskette in concert with the Automatic Configuration File supplied by Overland Data (file @7E76.ADF on the UTILITIES disk) will handle the entire process.

This section describes each of the configurable settings available. Normally you let the Configuration Program automatically select the appropriate settings. In some cases you may want to change them to solve a configuration conflict or to better match your environment.

The XL/2 options that are configurable and their default settings are listed below.

Option	Default Setting
I/O Address	360 hexadecimal
Interrupt Level	10 decimal
Drive Address	0
Pin J2-50	Low
Streaming or Start/Stop	Streaming

Address

The XL/2 uses 8 contiguous I/O addresses and can be addressed to any base address on even boundaries of 8 (0000 to FFF8 hex). Much of the address space below 400 hex is used by IBM.

The default address used by the XL/2 is 360 hex. If this conflicts with any other device (indicated by next to the selection), make another selection.

Interrupt Level

The IBM PS/2 has 11 interrupt levels available on the Micro Channel bus. This selection allows you to select any one of these, or to disable the hardware interrupts. The Overland Data software chains into the DOS system clock interrupt (IRQ 8) if the hardware interrupt is disabled.

The default interrupt level is 10 (decimal). If this conflicts with another hardware device, select another interrupt level.

Drive Address

This selection allows you to specify which drive address the XL/2 will select when the unit is first powered on. Normally, you should specify drive address 0.

J2-50 Density Speed Select

Pin number 50 on the J2 connector is used on some drives to select either one of two speeds or one of two tape densities. Refer to the tape drive documentation for the interpretation of this signal by your tape drive. The default setting sets this signal to low.

Streaming or Start/Stop

The selection of streaming or start/stop mode alters the sequence of commands issued by the software to the tape transport. The normal selection is streaming (for streaming tape drives). A few older drives may have trouble with commands being issued before the formatter-busy (IFBSY) line indicates that the formatter is not busy. This problem sometimes causes the tape drive to quit functioning until it is reset. In these rare cases, try the start/stop setting.

If you select start/stop on a streaming drive, you may experience a degradation in performance. This will manifest itself as an increase in tape repositioning.

Translation Tables

Control character representations used in these translation tables:

ACK	Acknowledge	IGS	Interchange Group Separator
BEL	Bell	IL	Idle
BS	Backspace	IRS	Interchange Record Separator
CAN	Cancel	IUS	InterchangeUnit Separator
CC	Cursor Control	LC	Lower Case
CR	Carriage Return	LF	Line Feed
CU1	Customer Use 1	NAK	Negative Acknowledge
CU2	Customer Use 2	NL	New Line
CU3	Customer Use 3	NUL	Null
DC1	Device Control 1	PF	Punch Off
DC2	Device Control 2	PN	Punch On
DC4	Device Control 4	RES	Restore
DEI	Delete	RS	Reader Stop
DLE	Data Link Escape	SI	Shift In
DS	Digit Select	SM	Set Mode
EM	End of Medium	SMM	Start of Manual Message
ENQ	Enquiry	SO	Shift Out
EOT	End of Transmission	SOH	Start of Heading
ESC	Escape	SOS	Start of Significance
ETB	End of Transmission Block	SP	Space
ETX	End of Text	STX	Start of Text
FF	Form Feed	SUB	Substitute
FS	Field Separator	SYN	Synchronous Idle
HT	Horizontal Tab	TM	Tape Mark
IFS	Interchange File Separator	UC	Upper Case
VT	Vertical Tab		

Appendix A—Controller Specifications

EBCDIC-to-ASCII

In	Disp. Char	Out	Disp. Char	In	Disp. Char	Out	Disp. Char
0x00	NUL	0x20	SP	0x24	BYP	0x20	SP
0x01	SOH	0x01	SOH	0x25	LF	0x0a	LF
0x02	STX	0x02	STX	0x26	ETB	0x17	ETB
0x03	ETX	0x03	ETX	0x27	ESC	0x1b	ESC
0x04	PF	0x20	SP	0x28		0x20	SP
0x05	HT	0x09	HT	0x29		0x20	SP
0x06	LC	0x20	SP	0x2a	SM	0x20	SP
0x07	DEL	0x7f	DEL	0x2b	CU2	0x20	SP
0x08		0x20	SP	0x2c		0x20	SP
0x09		0x20	SP	0x2d	ENQ	0x05	ENQ
0x0a	SMM	0x20	SP	0x2e	ACK	0x06	ACK
0x0b	VT	0x0b	VT	0x2f	BEL	0x07	BEL
0x0c	FF	0x0c	FF	0x30		0x20	SP
0x0d	CR	0x0d	CR	0x31		0x20	SP
0x0e	SO	0x0e	SO	0x32	SYN	0x16	SYN
0x0f	SI	0x0f	SI	0x33		0x20	SP
0x10	DLE	0x10	DLE	0x34	PN	0x20	SP
0x11	DC1	0x11	DC1	0x35	RS	0x1e	RS
0x12	DC2	0x12	DC2	0x36	UC	0x20	SP
0x13	TM	0x13	DC3	0x37	EOT	0x04	EOT
0x14	RES	0x20	SP	0x38		0x20	SP
0x15	NL	0x20	SP	0x39		0x20	SP
0x16	BS	0x08	BS	0x3a		0x20	SP
0x17	IL	0x20	SP	0x3b	CU3	0x20	SP
0x18	CAN	0x18	CAN	0x3c	DC4	0x14	DC4
0x19	EM	0x19	EM	0x3d	NAK	0x15	NAK
0x1a	CC	0x20	SP	0x3e		0x20	SP
0x1b	CU1	0x20	SP	0x3f	SUB	0x1a	SUB
0x1c	IFS	0x1c	FS	0x40	SP	0x20	SP
0x1d	IGS	0x1d	GS	0x41		0x20	SP
0x1e	IRS	0x1e	RS	0x42		0x20	SP
0x1f	IUS	0x1f	US	0x43		0x20	SP
0x20	DS	0x20	SP	0x44		0x20	SP
0x21	SOS	0x1b	ESC	0x45		0x20	SP
0x22	FS	0x1c	FS	0x46		0x20	SP
0x23		0x20	SP	0x47		0x20	SP

Appendix A—Controller Specifications

In	Disp. Char	Out	Disp. Char	In	Disp. Char	Out	Disp. Char
0x48		0x20	SP	0x6d	_	0x5f	_
0x49		0x20	SP	0x6e	>	0x3e	>
0x4a	c	0x7e	~	0x6f	?	0x3f	?
0x4b		0x2e	.	0x70		0x20	SP
0x4c	<	0x3c	<	0x71		0x20	SP
0x4d	{	0x28	{	0x72		0x20	SP
0x4e	+	0x2b	+	0x73		0x20	SP
0x4f		0x7c		0x74		0x20	SP
0x50	&	0x26	&	0x75		0x20	SP
0x51		0x20	SP	0x76		0x20	SP
0x52		0x20	SP	0x77		0x20	SP
0x53		0x20	SP	0x78		0x20	SP
0x54		0x20	SP	0x79		0x60	`
0x55		0x20	SP	0x7a	:	0x3a	:
0x56		0x20	SP	0x7b	#	0x23	#
0x57		0x20	SP	0x7c	@	0x40	@
0x58		0x20	SP	0x7d	'	0x27	'
0x59		0x20	SP	0x7e	=	0x3d	=
0x5a		0x21		0x7f	"	0x22	"
0x5b	\$	0x24	\$	0x80		0x20	SP
0x5c	*	0x2a	*	0x81	a	0x61	a
0x5d	}	0x29	}	0x82	b	0x62	b
0x5e	;	0x3b	;	0x83	c	0x63	c
0x5f	~	0x7e	~	0x84	d	0x64	d
0x60	-	0x2d	-	0x85	e	0x65	e
0x61	/	0x2f	/	0x86	f	0x66	f
0x62		0x20	SP	0x87	g	0x67	g
0x63		0x20	SP	0x88	h	0x68	h
0x64		0x20	SP	0x89	i	0x69	i
0x65		0x20	SP	0x8a		0x20	SP
0x66		0x20	SP	0x8b		0x20	SP
0x67		0x20	SP	0x8c		0x20	SP
0x68		0x20	SP	0x8d		0x20	SP
0x69		0x20	SP	0x8e		0x20	SP
0x6a		0x20	SP	0x8f		0x20	SP
0x6b	,	0x2c	,	0x90		0x20	SP
0x6c	%	0x25	%	0x91	j	0x6a	j

Appendix A—Controller Specifications

In	Disp. Char	Out	Disp. Char	In	Disp. Char	Out	Disp. Char
0x92	k	0x6b	k	0xb7		0x20	SP
0x93	l	0x6c	l	0xb8		0x20	SP
0x94	m	0x6d	m	0xb9		0x20	SP
0x95	n	0x6e	n	0xba		0x20	SP
0x96	o	0x6f	o	0xbb		0x20	SP
0x97	p	0x70	p	0xbc		0x20	SP
0x98	q	0x71	q	0xbd		0x5d]
0x99	r	0x72	r	0xbe		0x20	SP
0x9a		0x5e	^	0xbf		0x20	SP
0x9b		0x20	SP	0xc0		0x7b	{
0x9c		0x20	SP	0xc1	A	0x41	A
0x9d		0x20	SP	0xc2	B	0x42	B
0x9e		0x20	SP	0xc3	C	0x43	C
0x9f		0x20	SP	0xc4	D	0x44	D
0xa0		0x20	SP	0xc5	E	0x45	E
0xa1		0x20	SP	0xc6	F	0x46	F
0xa2	s	0x73	s	0xc7	G	0x47	G
0xa3	t	0x74	t	0xc8	H	0x48	H
0xa4	u	0x75	u	0xc9	I	0x49	I
0xa5	v	0x76	v	0xca		0x20	SP
0xa6	w	0x77	w	0xcb		0x20	SP
0xa7	x	0x78	x	0xcc		0x20	SP
0xa8	y	0x79	y	0xcd		0x20	SP
0xa9	z	0x7a	z	0xce		0x20	SP
0xaa		0x20	SP	0xcf		0x20	SP
0xab		0x20	SP	0xd0		0x7d	}
0xac		0x20	SP	0xd1	J	0x4a	J
0xad		0x5b	[0xd2	K	0x4b	K
0xae		0x20	SP	0xd3	L	0x4c	L
0xaf		0x20	SP	0xd4	M	0x4d	M
0xb0		0x20	SP	0xd5	N	0x4e	N
0xb1		0x20	SP	0xd6	O	0x4f	O
0xb2		0x20	SP	0xd7	P	0x50	P
0xb3		0x20	SP	0xd8	Q	0x51	Q
0xb4		0x20	SP	0xd9	R	0x52	R
0xb5		0x20	SP	0xda		0x20	SP
0xb6		0x20	SP	0xdb		0x20	SP

Appendix A—Controller Specifications

In	Disp. Char	Out	Disp. Char	In	Disp. Char	Out	Disp. Char
0xdc		0x20	SP	0xee		0x20	SP
0xdd		0x20	SP	0xef		0x20	SP
0xde		0x20	SP	0xf0	0	0x30	0
0xdf		0x20	SP	0xf1	1	0x31	1
0xe0		0x5c	\	0xf2	2	0x32	2
0xe1		0x20	SP	0xf3	3	0x33	3
0xe2	S	0x53	S	0xf4	4	0x34	4
0xe3	T	0x54	T	0xf5	5	0x35	5
0xe4	U	0x55	U	0xf6	6	0x36	6
0xe5	V	0x56	V	0xf7	7	0x37	7
0xe6	W	0x57	W	0xf8	8	0x38	8
0xe7	X	0x58	X	0xf9	9	0x39	9
0xe8	Y	0x59	Y	0xfa		0x20	SP
0xe9	Z	0x5a	Z	0xfb		0x20	SP
0xea		0x20	SP	0xfc		0x20	SP
0xeb		0x20	SP	0xfd		0x20	SP
0xec		0x20	SP	0xfe		0x20	SP
0xed		0x20	SP	0xff		0x20	SP

Appendix A—Controller Specifications

ASCII-to-EBCDIC

In	Disp. Char	Out	Disp. Char	In	Disp. Char	Out	Disp. Char
0x00	NUL	0x00	NUL	0x24	\$	0x5b	\$
0x01	SOH	0x01	SOH	0x25	%	0x6c	%
0x02	STX	0x02	STX	0x26	&	0x50	&
0x03	ETX	0x03	ETX	0x27	'	0x7d	'
0x04	EOT	0x37	EOT	0x28	(0x4d	(
0x05	ENQ	0x2d	ENQ	0x29)	0x5d)
0x06	ACK	0x2e	ACK	0x2a	*	0x5c	*
0x07	BEL	0x2f	BEL	0x2b	+	0x4e	+
0x08	BS	0x16	BS	0x2c	,	0x6b	,
0x09	HT	0x05	HT	0x2d	-	0x60	-
0x0a	LF	0x25	LF	0x2e	.	0x4b	.
0x0b	VT	0x0b	VT	0x2f	/	0x61	/
0x0c	FF	0x0c	FF	0x30	0	0xf0	0
0x0d	CR	0x0d	CR	0x31	1	0xf1	1
0x0e	SO	0x0e	SO	0x32	2	0xf2	2
0x0f	SI	0x0f	SI	0x33	3	0xf3	3
0x10	DLE	0x10	DLE	0x34	4	0xf4	4
0x11	DC1	0x11	DC1	0x35	5	0xf5	5
0x12	DC2	0x12	DC2	0x36	6	0xf6	6
0x13	DC3	0x13	TM	0x37	7	0xf7	7
0x14	DC4	0x3c	DC4	0x38	8	0xf8	8
0x15	NAK	0x3d	NAK	0x39	9	0xf9	9
0x16	SYN	0x32	SYN	0x3a	:	0x7a	:
0x17	ETB	0x26	ETB	0x3b	;	0x5e	;
0x18	CAN	0x18	CAN	0x3c	<	0x4c	<
0x19	EM	0x19	EM	0x3d	=	0x7e	=
0x1a	SUB	0x3f	SUB	0x3e	>	0x6e	>
0x1b	ESC	0x27	ESC	0x3f	?	0x6f	?
0x1c	FS	0x22	FS	0x40	@	0x7c	@
0x1d	GS	0x1d	IGS	0x41	A	0xc1	A
0x1e	RS	0x1e	IRS	0x42	B	0xc2	B
0x1f	US	0x1f	IUS	0x43	C	0xc3	C
0x20	SP	0x40	SP	0x44	D	0xc4	D
0x21	!	0x5a	!	0x45	E	0xc5	E
0x22	"	0x7f	"	0x46	F	0xc6	F
0x23	#	0x7b	#	0x47	G	0xc7	G

Appendix A—Controller Specifications

In	Disp. Char	Out	Disp. Char	In	Disp. Char	Out	Disp. Char
0x48	H	0xc8	H	0x6d	m	0x94	m
0x49	I	0xc9	I	0x6e	n	0x95	n
0x4a	J	0xd1	J	0x6f	o	0x96	o
0x4b	K	0xd2	K	0x70	p	0x97	p
0x4c	L	0xd3	L	0x71	q	0x98	q
0x4d	M	0xd4	M	0x72	r	0x99	r
0x4e	N	0xd5	N	0x73	s	0xa2	s
0x4f	O	0xd6	O	0x74	t	0xa3	t
0x50	P	0xd7	P	0x75	u	0xa4	u
0x51	Q	0xd8	Q	0x76	v	0xa5	v
0x52	R	0xd9	R	0x77	w	0xa6	w
0x53	S	0xe2	S	0x78	x	0xa7	x
0x54	T	0xe3	T	0x79	y	0xa8	y
0x55	U	0xe4	U	0x7a	z	0xa9	z
0x56	V	0xe5	V	0x7b	{	0xc0	
0x57	W	0xe6	W	0x7c		0x4f	
0x58	X	0xe7	X	0x7d	}	0xd0	
0x59	Y	0xe8	Y	0x7e	~	0xa1	
0x5a	Z	0xe9	Z	0x7f	DEL	0x07	DEL
0x5b	[0xad		0x80		0x00	NUL
0x5c	\	0xe0		0x81		0x00	NUL
0x5d]	0xbd		0x82		0x00	NUL
0x5e	^	0x5f	¬	0x83		0x00	NUL
0x5f	_	0x6d	_	0x84		0x00	NUL
0x60	`	0x7d	`	0x85		0x00	NUL
0x61	a	0x81	a	0x86		0x00	NUL
0x62	b	0x82	b	0x87		0x00	NUL
0x63	c	0x83	c	0x88		0x00	NUL
0x64	d	0x84	d	0x89		0x00	NUL
0x65	e	0x85	e	0x8a		0x00	NUL
0x66	f	0x86	f	0x8b		0x00	NUL
0x67	g	0x87	g	0x8c		0x00	NUL
0x68	h	0x88	h	0x8d		0x00	NUL
0x69	i	0x89	i	0x8e		0x00	NUL
0x6a	j	0x91	j	0x8f		0x00	NUL
0x6b	k	0x92	k	0x90		0x00	NUL
0x6c	l	0x93	l	0x91		0x00	NUL

Appendix A—Controller Specifications

In	Disp. Char	Out	Disp. Char	In	Disp. Char	Out	Disp. Char
0x92		0x00	NUL	0xb7		0x00	NUL
0x93		0x00	NUL	0xb8		0x00	NUL
0x94		0x00	NUL	0xb9		0x00	NUL
0x95		0x00	NUL	0xba		0x00	NUL
0x96		0x00	NUL	0xbb		0x00	NUL
0x97		0x00	NUL	0xbc		0x00	NUL
0x98		0x00	NUL	0xbd		0x00	NUL
0x99		0x00	NUL	0xbe		0x00	NUL
0x9a		0x00	NUL	0xbf		0x00	NUL
0x9b		0x00	NUL	0xc0		0x00	NUL
0x9c		0x00	NUL	0xc1		0x00	NUL
0x9d		0x00	NUL	0xc2		0x00	NUL
0x9e		0x00	NUL	0xc3		0x00	NUL
0x9f		0x00	NUL	0xc4		0x00	NUL
0xa0		0x00	NUL	0xc5		0x00	NUL
0xa1		0x00	NUL	0xc6		0x00	NUL
0xa2		0x00	NUL	0xc7		0x00	NUL
0xa3		0x00	NUL	0xc8		0x00	NUL
0xa4		0x00	NUL	0xc9		0x00	NUL
0xa5		0x00	NUL	0xca		0x00	NUL
0xa6		0x00	NUL	0xcb		0x00	NUL
0xa7		0x00	NUL	0xcc		0x00	NUL
0xa8		0x00	NUL	0xcd		0x00	NUL
0xa9		0x00	NUL	0xce		0x00	NUL
0xaa		0x00	NUL	0xcf		0x00	NUL
0xab		0x00	NUL	0xd0		0x00	NUL
0xac		0x00	NUL	0xd1		0x00	NUL
0xad		0x00	NUL	0xd2		0x00	NUL
0xae		0x00	NUL	0xd3		0x00	NUL
0xaf		0x00	NUL	0xd4		0x00	NUL
0xb0		0x00	NUL	0xd5		0x00	NUL
0xb1		0x00	NUL	0xd6		0x00	NUL
0xb2		0x00	NUL	0xd7		0x00	NUL
0xb3		0x00	NUL	0xd8		0x00	NUL
0xb4		0x00	NUL	0xd9		0x00	NUL
0xb5		0x00	NUL	0xda		0x00	NUL
0xb6		0x00	NUL	0xdb		0x00	NUL

Appendix A—Controller Specifications

In	Disp. Char	Out	Disp. Char	In	Disp. Char	Out	Disp. Char
0xdc		0x00	NUL	0xee		0x00	NUL
0xdd		0x00	NUL	0xef		0x00	NUL
0xde		0x00	NUL	0xf0		0x00	NUL
0xdf		0x00	NUL	0xf1		0x00	NUL
0xe0		0x00	NUL	0xf2		0x00	NUL
0xe1		0x00	NUL	0xf3		0x00	NUL
0xe2		0x00	NUL	0xf4		0x00	NUL
0xe3		0x00	NUL	0xf5		0x00	NUL
0xe4		0x00	NUL	0xf6		0x00	NUL
0xe5		0x00	NUL	0xf7		0x00	NUL
0xe6		0x00	NUL	0xf8		0x00	NUL
0xe7		0x00	NUL	0xf9		0x00	NUL
0xe8		0x00	NUL	0xfa		0x00	NUL
0xe9		0x00	NUL	0xfb		0x00	NUL
0xea		0x00	NUL	0xfc		0x00	NUL
0xeb		0x00	NUL	0xfd		0x00	NUL
0xec		0x00	NUL	0xfe		0x00	NUL
0xed		0x00	NUL	0xff		0x00	NUL

Limited Warranty

Products covered by this Warranty are limited to products manufactured by Overland Data Inc. This includes tape controller board products only, and specifically excludes tape transports or tape drives. Overland Data warranties for tape drive or tape transport products are specific to the make and model of transport delivered, and are covered individually.

Overland Data Inc. warrants to the original purchaser of this Overland Data Inc. tape coupler product that it is to be in good working order for a period of two years from the date of purchase from Overland Data Inc. or an authorized Overland Data Inc. dealer. Should this product malfunction during the warranty period, Overland Data Inc. will, at its option, repair or replace it at no charge, provided that the product has not been subjected to misuse, abuse, or non-Overland Data Inc. authorized alterations, modifications, and/or repairs.

Should a product require Limited Warranty service during the warranty period, a Returned Material Authorization (RMA) must be obtained prior to returning the product to Overland Data Inc. Any item returned for repair under the terms of this warranty must be accompanied by proof of purchase. You agree to insure the product or assume the risk of loss or damage in transit. You also agree to prepay shipping charges to Overland Data Inc.

ALL EXPRESS AND IMPLIED WARRANTIES FOR THIS PRODUCT INCLUDING, BUT NOT LIMITED TO, THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE TWO YEAR PERIOD. Some states do not allow limitations on how long an implied warranty lasts, so the above limitations may not apply to you.

UNDER NO CIRCUMSTANCES WILL OVERLAND DATA INC. BE LIABLE IN ANY WAY TO THE USER FOR DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF, OR INABILITY TO USE, SUCH PRODUCT. Some states do not allow the exclusion or limitation of incidental or consequential damages for consumer products, so the above limitation or exclusion may not apply to you.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH MAY VARY FROM STATE TO STATE.

The limited warranty applies to hardware products only.

Overland Data, Inc.

5600 Kearny Mesa Rd.,
San Diego, CA 92111-1383

TEL: (619) 571-5555

FAX: (619) 571-0982
