ELECTROVALUE

Tape Drive Software

Table of Contents

Preface

Electrovalue has been serving customers in the tape drive business since 1985. Electrovalue owes its success to many people and companies. In particular, I would like to thank the people whose contributions made this software possible.

W Gregg Stefancik – The programmer who coded most of the software starting with versions for the Apple II computer. Gregg wrote many programs no longer being sold by Electrovalue.

Webb Linzmayer – who specified the features and overall layout of the programs, tested the software, wrote most of the manual you are reading, fixed minor bugs, and coded simple features.

Rich Fisch – who designed and wrote the original version of D2T.

Sylvanus (Van) Zimmerman V – who fixed minor bugs and added features.

J David Dishman – who added features.

Rob Bain – who added features.

Ryan Johnson – who added features.

Leo Y Yochim – who wrote the section titled 'The History of Magnetic Tape"

In particular, I wish to thank Webb Linzmayer for letting me carry on the Electrovalue tradition. There are others too who helped make this possible.

Sincerely,

Joseph Minuti

Introduction

This software is designed to work with Pertec interface cards sold by Electrovalue for use with Pertec interface 9 track tape drives and various other SCSI ½:" tape cartridge drives including, 3480, 3490 and 3490e drives. Although it is not necessary to read this entire manual to use the software, I encourage you to take the time to go through this manual. You are sure to find some work-saving tips and procedures. The FX, D2T, and CUBS programs have a low/high speed mode. This mode has no effect unless you have a dual speed drive (eg F880).

Notes and warranty notice

Windows, Windows 95, Windows 98 are trademarks of Microsoft Corporation of Redmond Washington.

Electrovalue Technologies Inc. (Electrovalue) provides this manual "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Electrovalue Technologies Inc. may make improvements or changes in the product(s) and / or programs described in this manual at any time. These changes will be incorporated in new editions of this publication.

The History of Magnetic Tape

By Leo Y Yochim

[Leo is the founder of Printronic Corporation of America, a leading data processing company]

The mainframe computer was invented in the late 1940's. Two professors from the University of Pennsylvania, who then started their own company called UNIVAC, invented it. Therefore, the first computer that was used scientifically was called UNIVAC and came out of Philadelphia.

In those early days, the first storage devices were electronic radio tubes. Each tube, which took up a space about six inches by three inches, could store one character. In those days, they also stored data in mercury delay lines. These were tubes of mercury about ten feet long into which you electrically induced the waves that represented the numerical data which would travel down the tube of mercury and recycle until you needed that data.

Data was also stored on the face of CRT tubes. Instead of pictures, the CRT tube would have data. The data was stored in binary code in the face of tubes and would be reusable. Also, magnetic drums 6 inches in diameter and one to two feet long could also store data. Of course data was also stored on punch cards, paper rolls, and probably a few other kinds of media in those days.

There was no easy way when the computer was first build to store data economically until magnetic tape was designed. The first magnetic tape was probably designed in the late 1940's by UNIVAC. It was a roll of steel about 1/60 of an inch thick and about 1,000 feet long. It was wound on a roll and it weighed approximately 35 pounds. This was not very acceptable for developing data on magnetic tape and selling it around the country.

This particular roll of steel magnetic tape could store just 100 characters in one inch of tape. By the way the tape on the first steel rolls was just ¾ " wide and it was not a satisfactory answer. In the early 1950's a tape was made with a Mylar backing and an iron oxide that had the same properties of steel but weighed only two to three pounds. So magnetic tape became much more useful. It was smaller, and had a 2500' not 1000', therefore, you could store over twice the data in a substantially less space.

Remember: The key word here was 100 characters per inch

In the early 50's they learned how to double the capacity of the data in one inch of magnetic tape and they could store 200 characters in one inch of magnetic tape. As time went by, better inventions came along and they increased the capacity to 556 numbers. Remember these are not upper and lower case characters but either an alpha character or a number. We could only store 48 characters in one frame and we could have as we said 100 frames, 200 frames, and 556 frames into the mid-1950's. Toward the end of the 1950's we got all the way up to 800 characters per inch. That is an eight times growth factor in less than ten years.

In the beginning of the 1960's they learned how to store double the 800 or 1600 characters per inch but they referred to it as bits per inch. So, 1600 characters per inch became 1600 BPI. The next growth jumped over the 3200 standard and went all the way up to 6250. So in one inch of magnetic tape they could store 6250 characters. Therefore, in about 15 years, we increased the capacity of storage on magnetic tape 600 times.

You don't get 600 times the mileage per gallon of gasoline from the late 1940's through the middle 1960's!

The early magnetic tapes had to be manually threaded onto the take-up reel. Sometime, in the early 1960's, they invented a tape device on which you could mount the tapes and you could automatically feed the tape into position. The take-off reel would run the tape into position.

Technology changed in the 1970's when they designed and invented the cartridge. Magnetic tape was then stored in a cartridge 4" or so wide. The new cartridge was half the size, half the price and double the speed of the reel type magnetic tape. The new cartridge could store 18,000 characters in one inch of magnetic tape. So over a 30-year period, you went from 100 characters per inch to over 18,000 in one inch. This is a growth factor of 1800 percent.

Then in the early 1990's they learned how to double the capacity of the 18,000 BPI cartridge file to 36,000 BPI. So now in the 50 years computers have been around, we have increased tape storage capacity 3600 times. This is phenomenal growth! Of course, all this tape development has been followed by gigabytes of disk storage for insignificant costs as compared with the original computer storage costs.

The above discussion has been about the capacity of storing information on magnetic tape. I have not discussed the speed of reading information off of magnetic tape which again got faster and faster as the information was being read off the magnetic tape faster and faster.

I can't even guess what the next growth in magnetic tape will be. It's just phenomenal what has happened and what will happen with these tape devices.

It all started on the mainframe computer and is now available on the PC.

Mailing lists

The 1950's
Fifty years ago, almost all mailing lists were on what were called Addressograph plates or Speedomac plates. These were one inch by three inch steel or tin plates where the name and address of the recipient was embossed. There was insignificant flexibility. You could mail to that address, and that is about all you could have done. Few, if any, mailing lists were available for rent. Nobody would give out his or her mailing list for rent for someone else to use.

The 1960's
Even in the 1960's the computer was probably too expensive to use for maintaining a mailing list.

There were some punch cards around, and the Social Security Administration had their 130 million Social Security payees on punch cards. They could not afford magnetic tape or mainframe computers. They used punch cards. However, in 1963 the Post Office announced the requirement for a site; a two-position zone on all mailing addresses. That meant that the industry had to recreate all the addressograph plates to add a zone. Or, was it time to go to magnetic tape?

Most companies chose, in the 1960's to join the computers and use magnetic tape for the maintenance of their mailing lists. In 1963 Printronic was started and the first computer generated letters appeared on the scene. They increased the response by over 400% from just a label on an envelope.

The 1970's
These were the years when magnetic tape became affordable for mailing lists. Mailing lists were maintained on mainframes that cost millions of dollars. Small companies with mailing lists of 1,000 to 5,000 names could generally not afford a computer, and if anything, they would go to a service bureau to have the work done.

The 1980's
High-speed printers were introduced in the 1980's so that labels were printed very rapidly. Computer letters could contain all types of personalization. It was the beginning of a new wave of direct mail as we begin to know it today. What really made the 1980's was that computer hard disks were now able to store gigabytes of data, and it wasn't necessary to have magnetic tape in tens of thousands of reels. You could store most information on disks and use magnetic tape only to back-up the disk, not to do the work from.

The 1990's
The PC became the forefront of most processing for direct mail. Mailing lists became available to everybody. The PC economics were available to everybody, and most files were now being transformed to the PC, which now talked had gigabytes of storage on a hard disk. Of course during the 1990's the Internet began making a tremendous impact on all industries, not just the direct mail industry. All of this development seemed very interesting because it laid the groundwork for where we are going right now.

The year 2000 - Today
The year 2000 brings on the PC, the Internet, electronic printing presses, digitalization in the Post Office, and a whole new way of doing business with the Post Office.

Hardware Installation

At the computer:

Turn off the power to the PC

Open your PC to expose an empty expansion slot.

Remove the expansion slot cover from the rear panel.

Plug the controller card in the slot.   Replace the screw.

Plug the cable into the mating connector on the rear of the card.   It can go on only one way.

Replace the cover on the PC.

At the tape drive:

Turn off the power at the tape drive.

Plug the cable onto one connector on the drive. There are four edge connectors. Pick any two, one labeled P1, the other P2. If you purchased your drive from Electrovalue, choose the connectors just below the colored dots and match up the colored dots. Attach the ground wire on the cable to the screw labeled ‘frame ground’ wire. This will ground the tape drive to the PC to reduce electrical noise and for safety’s sake.

Hook connector J1 to P1, and J2 to P2. The remaining 2 connectors are for daisy chaining the drive and are not used.

Placing peripherals on the drive

The top of the tape drive is a convenient place to put a printer, monitor, or other piece of equipment. Sometimes, this will cause problems, other times it won’t. Some peripherals radiate electromagnetic energy. If too much of this energy gets to the tape drive’s head, it will cause the drive to continuously move tape forward without reading or writing. We have run into this problem and so have some of our customers. The good news is that any problem caused by placing equipment on the tape drive is temporary. Remove the equipment from the tape drive and the problem will disappear.   No harm is done. We don’t recommend placing any equipment on top of your tape drive.

Software Installation

Note:

- To run this software under DOS requires at least Dos 3.3 or higher with 640k of memory.
- This software will also run under Windows, Windows 98, and Windows Millennium Edition (ME). This software does not support long file names.

- This software will NOT run under Windows for Workgroups, Windows NT or Windows 2000.
- Remember that cartridges in this manual refers to 3480, 3490, and 3490e cartridges unless otherwise noted.

General: Make a directory or folder on your hard drive. Copy the files on the disk to the directory you just created. None of the programs depend on the presence of any other files.

Important: Do not use the distribution diskettes as your working diskettes. Copy the contents to your hard disk, make an extra insurance copy of the original diskettes and store them in a different place.

Example: Copy all the files from the distribution disk in the A: drive to the TAPE directory.

```
        MD C:\TAPE

        CD C:\TAPE

COPY A:*.*
```

Or, you may use the Windows explorer to copy the files to a folder (preferably in the root directory) on your hard drive.

If you wish, you may edit the PATH command in your AUTOEXEC.BAT file to include the "TAPE" subdirectory. Then you can access any of the tape drive software from any drive or directory. Otherwise, you will have to change to the tape software directory or precede the program names with the complete path to execute any of the programs.

Configuring The Software

This software is DOS based and will run under Windows 95 and Windows 98. Because the software is DOS based, it does not support long file names.

TACO is used to configure the Electrovalue tape software settings so the software can access the tape drive.

The TACO Program

PURPOSE: TACO (TApe COnfiguration) allows the user to easily reconfigure the default Hardware specific values used by the Electrovalue tape drive software.

FEATURES**:** TACO lets you set the way the tape drive software uses certain hardware resources. You can:

Setting the I/O port .

The controller card is configured for I/O port 220. Normally, this will work fine. If you must change this you have to set the jumpers on the card and run TACO to configure the software.

Setting the drive ID number.

This is the ID of your drive. It is normally set for zero. It may be changed through the front panel or through switches in the drive. Usually, you should not change this.

This discussion assumes that you already copied all the tape drive programs into a single subdirectory on your hard drive. Note that TACO will process only the programs that actually access the tape. Thus FXED and LRL need not be present although there's no harm if they are. TACO looks for tape drive control programs ('exe' files) in the current directory.

Using TACO:

- go to your tape drive subdirectory or folder.
- Type in <TACO>
- Answer the questions TACO asks giving the controller card I/O setting and the tape drive numbers
- Hitting <ENTER> alone will result in using the displayed default parameter.

TACO will inspect each selected EXE file to see if it contains a valid parameter configuration block. The parameters of each qualifying program will be updated and the name of the program will be listed on the screen. When all selected EXE files have been exhausted, TACO will return to the operating system.

Whenever any tape drive program (FX, KEYS, DITTO, D2T, CUBS) is executed, it displays the hardware dependent parameters on the top line of the screen. Thus, if you like, you can simply execute one of these programs to confirm the changes.

TACO doesn't require that the programs it modifies have any particular names.

If no filename is specified on the command line, all the EXE files in the current subdirectory are inspected for a valid 'Electrovalue Tape Software' parameter block. Normally, you should run TACO without specifying any optional filenames so as to configure all the software in the subdirectory.

Because the software checks each .EXE file for a configuration block, the configuration will run faster if you don't have a bunch of non-Electrovalue .EXE files in the subdirectory.

If you do reconfigure the software, you, of course, have to make corresponding physical changes on the controller card and/or the tape drive unit. Reconfiguration simply changes the default values the program will use. Reconfiguring does not, of course, change the values on the drive or controller card.

Although TACO actually re-writes all modified disk files, the original dates and times are preserved as an aid in revision control.

Taco can optionally configure only selected files. You can do this if you wish to change only selected programs.

TACO [optional list of filenames] <ENTER>

E.g.: TACO CUBS FX <ENTER>

If one or more filenames has been specified on the command line (as in the example above), TACO will process only the named file(s). If no filenames are given, TACO will process all the valid tape drive programs it can find in the current directory.

Testing The Software

We will use the CUBS program to create, write, read, and compare some test data. Load a scratch tape in the drive. Warning: This test will overwrite any data on the tape. Use a blank tape, or a tape with data you no longer need. With the drive loaded and online, go to the tape drive software directory. Type in "CUBS B M=250 32K H ". This will call the CUBS program and tell it to write 250 blocks of data at a block size of up to 32K. The program will then read in the data checking for positioning information and block size. CUBS will display progress and error status if any. If the program writes and then reads 250 blocks of data with no errors you are set. If CUBS reports "I/O transfer errors" or "Hard errors" (HER) you will have to change the jumper settings on the controller card and run TACO to configure the software.

REMEMBER: An improperly configured drive will seem to be broken. This is because there is another device sharing the memory address space with the drive software. This corrupts the data to the tape drive.

Errors can occur because our card is not "plug and play". When two cards try to use the same I/O address space, the cards interfere with each other. This can cause problems with the tape controller card and the offending card. These problems show up in Electrovalue software as: "data transfer errors", or messages by the software saying "Rewinding" when the tape is not rewinding, or other general problems.

Often an I/O conflict will make it seem like the tape drive is broken. The tape drive is not broken. The controller card is getting bad (corrupt) data and is operating erratically.

Problem Resolution

Remember:

- The software must be configured with the correct controller card I/O setting and drive numbers.

- If the programs say, "Tape drive not ready or not online" even though the drive is online. Make sure the controller card I/O software setting matches the jumper settings on the card.

- Remember, the jumpers on the controller card must match the settings for the software. Otherwise, the programs will not see the controller card.

- An I/O conflict will make it seem like the tape drive is broken. Change the I/O jumpers and run TACO to set the software.

Jumper settings on the controller card

The default I/O base address is 220H.

If you need to change the I/O address of the card because the CUBS B M=250 32K H test does not run correctly, you should adjust the jumper settings as corresponding to the chart below and run TACO to set the software settings to conform to the jumper settings.

Note that a 1 means the jumper is in place. A 0 means the jumper is off.

Use the settings in the table below

Powers of 2				HEX	
5	6	7	8	ADDRESS	
1	1	1	1	200	
0	1	1	1	220	🡐 Factory setting
1	0	1	1	240	
0	0	1	1	260	
1	1	0	1	280	
0	1	0	1	2A0	🡐 Recommended second choice
1	0	0	1	2C0	
0	0	0	1	2E0	
1	1	1	0	300	
0	1	1	0	320	
1	0	1	0	340	
0	0	1	0	360	
1	1	0	0	380	
0	1	0	0	3A0	
1	0	0	0	3C0	
0	0	0	0	3E0	

Cleaning The Heads And The Tape Path

Clean the tape path (especially the head) following the procedures outlined here.

Magnetic tape sheds oxide during normal tape operations. Debris from the tape should be removed by periodic cleaning. Electrovalue recommends the use of lint free swabs or cloth and 91% ISOPROPYL ALCOHOL to periodically clean the read/write heads and tape path components. If you have a cartridge machine, you should use a cleaning cartridge as recommend by the manufacturer. How often you should perform these cleaning procedures depends on usage, operation environment and tape quality. You can usually get 91% Isopropyl alcohol at your local drugstore. We do not recommend the use of alcohol mixtures containing less than 91% alcohol. The rest is water, which extends drying time. Make certain that you let the heads after cleaning them. Loading a tape with wet heads will rip oxide from them making the heads dirty.

In most cases you should perform these cleaning procedures every eight hours of tape drive usage. If error messages (e.g. FX's Read Error Corrected by drive message) begin to occur regularly, you should clean the drive head and tape path more frequently. If frequent cleaning does not improve reliability, check your tapes. Are the tapes old, worn or kept in a dirty or humid area? Old and or worn tapes should be copied and then discarded.

The definitions in this chart should help you develop an appropriate cleaning schedule:

Cleaning schedule

Usage	**Cleaning Schedule**
MINIMUM	Clean the tape path thoroughly after every eight hours of use if: • Less than ten reels of tape are used in eight hours. • You see no particles on the tape head after each reel of tape. • You do not suspect abnormal amounts of dust in the drive's environment from increased traffic or vacuuming.
NORMAL	Clean the tape path thoroughly after every one or two hours of continuous running if: • More than ten reels of tape are used in eight hours. • You see no particles on the tape head after each reel of tape. • You do not suspect abnormal amounts of dust in the drive's environment from increased traffic or vacuuming.
HEAVY	Clean the tape path thoroughly after each reel of tape if: • Particles appear on the tape head after each reel of tape.

	• You are reading interchange tapes from an unknown tape drive. • You are using new or little-used tapes (new tapes often contain debris from the slitting process during their manufacture).
SPECIAL	Clean the tape path if you suspect abnormal amounts of dust in the drive's environment because of custodial activity, equipment moves, supply deliveries, or if the drive has not been used for several days.

NOTE: Some sources suggest that you clean the heads with "FREON TF". Don't bother trying to find it. The EPA made it disappear. Isopropyl alcohol is fine.

9 track drive cleaning procedure

Pour a small amount of 91% ISOPROPYL ALCOHOL into a clean container, such as a small UNWAXED paper cup. Alcohol dissolves wax. If you use a waxed cup, the wax transfers to the tape path. Don't use 70% isopropyl, the other 30% is water and will extend the drying time. **The worst thing you could do is load a tape with the heads still damp. This would immediately rip oxide from the tape, making the heads dirtier than before you started.**

Dab all cloths and swabs into the container as needed.

Applying gentle but firm pressure in one direction, use a lint-free swab or cloth to clean the following surfaces.

Clean the read/write/erase head. Start with a CLEAN swab or cleaning cloth. Use one or more until no more dirt is removed. This is the most important assembly to clean. Unlike the head on a VCR, the read/write head is not fragile. Use firm pressure and a VERTICAL motion. Too much horizontal pressure could misalign the drive's skew adjustment, making it less compatible in reading and writing tapes for exchange with other drives.

Facing the drive from the front, to the right of the read/write head, locate the tape cleaner block, usually made of clear plastic set in metal. Use a vertical motion to clean the scraping surfaces and debris trap. The job of the tape cleaner block is to remove tape debris before it reaches the read/write heads. It will normally be the dirtiest surface you clean.

Periodically check and wipe off the rubber gripping fingers on the supply reel hub. Do NOT use alcohol on the supply reel hub gripping fingers.

Brush out the entire tape drive bed with a lint-free cloth to remove any remaining debris. Let the tape drive dry for several minutes before loading a tape. Remember loading a tape with any of the surfaces still damp will defeat the purpose of cleaning and may remove oxide from the tape, making it unreadable.

Cartridge drive cleaning procedure

Some drives have a ‘cleaning required’ light or an error code which makes itself known when it requires cleaning. Most drives do not tell you when they need to be cleaned.

If the manufacturer recommends a cleaning schedule, it would be best to follow that schedule. If the manufacturer does not recommend a cleaning schedule, use the schedule shown in the above table. In general, cleaning cartridges are only good for a set number of uses. Replace the cleaning cartridge after it has been used the specified number stated by the manufacturer.

Short Program Descriptions

FX	- is used to read and write tape files in industry standard format so that the data may be exchanged between your PC and minicomputers/mainframes. FX , along with the features added by FXED, is the most important program.
FXED	- is used "off-line" to create FXED control files. FXED control files enhance FX's capabilities with advanced features useful when processing fixed-length mainframe data records. These same powerful, field- processing and record-selection features are also available from within FXED in "DISK-TO-DISK" mode.
CUBS	- is very useful in the analysis of labeled and unlabeled tapes of unknown content. You'll want to use CUBS's "L" mode on each mystery tape you receive for processing.
DITTO	- is invaluable in analyzing the FIELD STRUCTURE of files containing fixed length records. Input is from tape or disk. The data is displayed below column numbers generated by DITTO that show the location of each character in the record.
D2T	- creates mainframe friendly tape files and can optionally generate layout documentation directly from dBASE and dBASE compatible internal-format .DBF files. This means you don't need any free space on your disk! Without this program, you'd need free space the size of the .DBF file.
KEYS	- is useful in quickly determining the block size, code (ASCII or EBCDIC) and record length used on mystery tapes using the “R”, “A”, “E”, “D”, “N”, and “C” commands. Additional KEYS features are useful in hardware testing, erasing tapes, and general tape analysis.
LRL	- does not access tape directly. LRL generates possible record lengths from block sizes you input. LRL is helpful in determining unknown logical record lengths. See the chapter "Dealing With Mystery Tapes".
U	- does nothing more than send an unload command to the tape drive. If the tape is not at the load point (BOT), U sends a rewind command prior to the unload command.

Quick Start & Overview

This note is only meant to summarize what we are trying to accomplish.

From tape to PC

Our objective is to read in the data from the tape or cartridge to process it on the PC. We will use the FX program to transfer data from tape to the PC. If we have a simple file transfer, we can simply start FX and follow the prompts. If we want to select out fields, do record level manipulation, selectively import records, or other record level conversion, we will use the FXED program to create an additional control file that tells FX what we want to do.

Read the chapter on FX to learn about transferring files from tape to disk.

Writing data to tape.

The data should usually be in a mainframe-friendly format. This means having each data field length fixed and having each record contain the same number of records. Thus, you will have a fixed length, fixed field file.

D2T will put dBASE compatible files in a fixed length mainframe-friendly format and write the file to tape in one step. In addition, D2T can optionally print a file layout and information report to accompany the tape.

FX can transfer other types of files to tape. You can use your database program to put the data in a mainframe-friendly fixed length format, and then use FX to transfer the file to tape truncating the carriage return and line feed ending each record (if the carriage return and linefeed exist)

In summary, it's easy to use FX to transfer files from tape to disk or from disk to tape.

Read the chapters on FX and D2T to learn about transferring files from disk to tape.

Running this software under Windows 95/98

This software will run under a DOS window under Windows 95/98 in addition to running under MS-DOS.

There are more than just a few ways to run a DOS program on a Windows 95 machine. We have tried eight of them and the list is still growing. They are as follows:

1. Boot your PC. Hit the "F4" key as soon as you see "Starting Windows 95..." Note that this runs your "OLD" version of DOS, i.e. the DOS that was active on your PC when you installed Windows 95/98. So, this isn't really running under Windows 95/98, it's a way to run our software on a machine that has Windows 95/98 installed. If your PC came with Windows 95 pre-installed [or you allowed Window 95's installation program to remove your old DOS when you installed it] you won't have a previous version of DOS and hitting F4 will have no effect. In this case, skip ahead to number 3.

2. Boot your PC. Hit the "F8" key as soon as you see "Starting Windows 95..." You'll be presented with a menu at this point. Choose "Previous Version of MS-DOS." Choosing this is equivalent to number 1 above thus this choice won't be present in every situation.

3. Boot your PC. Hit the "F8" key as soon as you see "Starting Windows 95..." You'll be presented with a menu at this point. Choose "Command Prompt Only".

4. Click START, Select Programs, click MS-DOS PROMPT. This runs Windows 95's DOS in a Windows 95 window.

5. Click START, Click RUN, hit <DEL> to clear the box and type in the Disk drive, path etc. e.g.: **C:\TAPE\FX <ENTER>.** This runs the program under Windows 95's DOS.

6. Click START, select PROGRAMS, click on Windows Explorer, find the program you want to run and run it. This is technically equivalent to number 5 above.

7. Click START, select "Restart the Computer in MS DOS Mode", click Yes. This puts you in Windows 95's DOS. .

8. Double click the "MY COMPUTER" icon on the desktop, double click on the icon of the disk drive which contains the tape drive software, then select the "folder" (directory) where the tape drive software resides, then select the program you want to run,   This runs the program under Windows 95's DOS.

The FX Program

PURPOSE: FX transfers data.

FROM TAPE TO..........DISK, SCREEN OR PRINTER.

FROM DISK TO..........TAPE.

FEATURES: Dozens are available to make the data "friendly" to the target computer. A basic set of features is available within FX itself; a number of advanced, data-import features are available through use of the FX control file editor "FXED".

Although the most common use of FX is to exchange text files between the PC and other computers there are no restrictions whatsoever concerning the type of files processed by FX. Binary files may be read and written without restriction. The maximum physical tape block length is 65535

USE: To run FX simply type: FX <ENTER>

The following menu will be displayed:

```
D         Transfer from Tape to Disk
P         Transfer from Tape to Printer
S         Transfer from Tape to Screen

T         Transfer from Disk to Tape

C         Display Disk Directory (Catalog)
M         Toggle Single/Multi Block Modes (Tape to Disk)
V         Toggle Low/High Velocity mode
Q/U       Quit FX program.   Unload tape, then quit FX
*         Toggle Log to Line Printer mode on/off
F1        Aborts any operation in progress.
```

Make your selection by hitting the appropriate single letter. The mnemonic significance of the choices that transfer data is that they represent the first letter of the "TO" device. Answering any subsequent questions is fairly straightforward. The rest of the topics presented below elaborate on the features and operation of FX and should be read if you would like to take full advantage of its capabilities and stay out of trouble.

Block size on tape

You are asked to specify the block size in bytes when writing tape. The maximum block size that can be written is 65535 bytes (64K -1). When reading tape, the block size need not be known but it is also limited to 65535 bytes. If you select any of FX's internal features which add characters to the data being read from tape the maximum block size is 32767 bytes (32K -1); see discussion on the next page. This restriction does not apply if you are using an FXED control file (control files are produced by FXED - see the FXED chapter for additional details).

Answering questions

Questions that require only a single character answer do not require you to hit <ENTER>. Hesitate after typing the character if unsure. Otherwise the <ENTER> key could be confused as the answer to the following question.

Tape positioning

By convention the first file on tape is called file 0. Normally (see next paragraph for the exception), when first executed, FX rewinds the tape to load point. When performing any subsequent tape operations, FX keeps track of where the tape is positioned so that it can move directly to any requested file with maximum speed. The user should keep this in mind and not manually position tape without exiting from FX and running it from the beginning.

"NR" No Rewind option: If you specify "NR" on the command line (e.g. "FX NR <ENTER>"), FX will not rewind the tape on the way in, nor on the way out. This helps in situations where FX is called from another program.

"Enter tape file number [the first tape file is file 0]:

Simply enter the number of the tape file you want to read, then hit <ENTER>. Remember, the first file on a tape is, by convention, file # 0. See the next page for information on transferring multiple tape files to disk.

Users who want to start processing at some point other than the first block of a file may make use of FX's block positioning feature.

Reseting 8th bit

ASCII character set consists of 128 seven-bit characters. Thus, when reading an "ASCII" tape, one would expect that the 8th or high-order bit of every byte would be 0. Sometimes, "ASCII" tape data has this bit set. The PC's interpretation of these characters would be bizarre. Answering **Y** to the above question forces an immediate reset of the 8th bit of every character from tape, **before** any further processing (e.g. by an FXED control file). If you're not having problems you should answer this question with **N**. Note that EBCDIC is an 8-bit code. If you answer **Y** to this question, FX will skip the "*Convert EBCDIC to ASCII ?*" question. It's up to you to avoid selecting this incompatible option when processing an EBCDIC tape with an FXED control file. If your data is in EBCDIC, you must answer N to this question.

Using an FXED control file

"If input data processing criteria are defined in an FXED control file, input the <filename>, or press <ENTER> alone for no control file:"

FXED control files, produced by the FXED program, allow FX to perform additional, advanced features while transferring fixed-length data records from tape to disk. If you specify a control file name, the following questions (except for "Enter disk file name:") will be skipped. See the FXED chapter for full information.

Change all control characters to spaces

Some tape data contains control characters. The FX option “Change all control characters (00-31 dec.) to Spaces (32 dec.)?(Y/N) ” lets you change troublesome control characters to harmless spaces. Control characters are characters with a decimal value from 0 to 31 inclusive. In most cases, when processing data on the PC, you don't want control characters. The most troublesome control character users run into is CTRL-Z (dec. 26). Unless you're trying to import data containing control characters on purpose, (e.g. material destined for a typesetter or formatted printer) answer "Y" to this question. Converting control characters to SPACEs is performed after any other conversions (EBCDIC to ASCII, Reset 8th bit). If you've specified the use of an FXED control file, the above question is skipped. See FXED's "Secondary Translation" option for FXED's similar function.

Change all CR's and/or LF's into the CR-LF pair ?

ASCII text files on the PC normally end each line with the character pair "CARRIAGE RETURN-LINE FEED". However, some computers end each line with just a single CR or a single LF. When engaged, this feature of FX looks at every character coming in from tape and changes each CR or LF to the pair of characters

"CR-LF". If the data from tape (e.g. tape to printer) is overprinting itself you may want to experiment with this feature and the "Add CR-LF after how many char?" feature described below to find the one best suited to your situation. Selecting this option will cause FX to skip the "Add CR-LF after how many characters ?" question described below - you can't have both options in effect at once. See the "BLOCK SIZE LIMITS ..." paragraph below for additional information. This question is skipped when CHOPing.

Convert EBCDIC to ASCII ?

If you want to convert from EBCDIC answer Y otherwise the data will not be translated.

Add CR & LF after how many characters?

For convenience in processing tape files containing records that do not end in CARRIAGE RETURN, FX contains a feature that can automatically add a CARRIAGE RETURN and a LINE FEED after every "N" characters. This facilitates the orderly printing of files containing logical records of a constant size, importing files for processing with database programs, etc. If you answer the question with just the <ENTER> key the automatic CR & LF feature will be turned off.

Block size limits when "CR & LF" options are in effect:

When either CR & LF option has been selected, FX cuts the input buffer size in half to provide a block of memory into which the text coming from tape can be copied while the selected characters are being added. This limits the physical size of the tape block that can be processed to 32,767 bytes (32K -1). Reading a larger block will result in an error message and the transfer will be aborted. To work around this limit, you can re-block the data by first reading the file to disk without any options, then write it back out in smaller blocks [be sure the block size is an integer multiple of your record length]. Alternatively, you can use an FXED control file to add the needed character(s). See the FXED index page, "INSERTING CR/LF" and "INSERTING LITERAL TEXT".

If you enter a decimal number "N" (followed by <ENTER>) the pair of characters "CR & LF" will automatically be added after every "N" characters. If, for example, the tape contained 80-column punched card images end-to-end you would enter "80". If you were transferring from tape to disk, due to the addition of the two characters at the end of each line, each resulting "record" would be 82 characters long - the file size would grow by 2.5% This option can not be selected if you have already selected CHOPing (see page FX5) or the "Change CRs and/or LFs into CR-LF " option - these combinations of options are mutually exclusive. See block size limitations in following paragraph.

```
Enter a single disk <filename>,
or <filename>/A to append to an existing disk file.
or <filename>/D to append to an existing .DBF file.
Or hit <F1><ENTER> for main menu:
```

Enter the file name (precede with pathname if needed) just as you would with any DOS command, then hit <ENTER>.

```
Examples:                CENSUS.DAT <ENTER>
                         C:\NAMES\GIRLS\BLONDES.TXT <ENTER>
```

Transfer of multiple tape files to disk

NOTE: This is an advanced feature to be used only when you know you have multiple files to transfer from tape that all have the same data characteristics.

If you have a lot of tape files to transfer to disk it could be a tedious job to sit at the computer and answer all the "tape-to- disk" option questions over and over again. FX provides a way around this. Do not use this feature unless and you are sure that the data in each tape file has the same characteristics (e.g. same code (EBCDIC or ASCII), same record length (for CR/LF insertion etc.). If this is the case and you're willing to accept one of the disk file name choices (described below), you can have FX transfer a large number of tape files to disk for you automatically.

CALLING THIS FEATURE INTO PLAY: NOTE: To avoid confusing the inexperienced user by suggesting a seldom selected feature, the following question DOES NOT invite him to answer with more than one file number, BUT YOU CAN:

"Enter tape file number [the first tape file is file 0]:"

Simply answering with more than one tape file number will put FX in multi-file mode. File numbers are either separated by a SPACE (to indicate individual files) or by a DASH (to indicate ranges of files). The following examples will serve to illustrate how you can specify tape file numbers:

```
0                 Only tape file 0 will be transferred.  Single-file mode.
1 3 5-8 11        Files 1 3 5 6 7 8 and 11 will be transferred.  Multi-file mode.
0-15 22 109       Files 0 thru 15 and 22 and 109 will be transferred.  Multi-file
mode.
```

WHAT IF I SPECIFY A FILE NUMBER THAT IS TOO LARGE? When multiple files are specified, FX asks you what to do if a "Double filemark" is encountered. (Two adjacent filemarks are the standard way of indicating "End-of-Data" on tape.) The choices are (P)ause, (R)ewind and (U)nload. If two adjacent filemarks are hit and you've selected (P), you'll be informed of the situation and be given the option to (C)ontinue or (A)bort.

WHAT ABOUT DISK FILE NAMES? THIS IS HOW YOU CREATE EITHER ONE OR MANY DISK FILES. If you have specified more than one tape file to be transferred to disk FX will ask for the disk file name in a special way:

Enter a single disk <filename>, to append all tape files into that one disk file, or a disk <filename> extension alone [e.g. .TXT] to generate individual disk files, one per tape file, with that extension.

Or hit <F1><ENTER> for main menu: D:\TEMP <ENTER>

If you enter a single disk file name all the tape files specified will be appended into this one disk file. If you enter only a filename extension (a dot followed by 1-3 characters), each tape file will be transferred into a separate disk file. They will all have the specified filename extension with the left part of the filename consisting of the letters "TF" (stands for Tape File) followed by the tape file number. If, for example, you select tape files 1 3 5-8 and 11 for transfer and enter ".ABC" as a file name, you'll get disk files named TF1.ABC, TF3.ABC, TF5.ABC, TF6.ABC, TF7.ABC, TF8.ABC and TF11.ABC.

Transfer of large tape files to disk

A) "DISK FULL" SITUATION: When a tape-to-disk operation is started, FX takes note of the amount of free storage remaining on the disk. FX keeps track of the amount of data written to disk and will stop the transfer just short of filling the disk. If this happens you will be asked to insert a new formatted disk. It's a good idea to keep formatted floppies on hand in anticipation of this situation. If you don't have any formatted disks on hand, all is not lost. You can abort the transfer by hitting <F1><ENTER>. See the discussion of positioning by blocks and display of block numbers on page FX12. Use of these features will allow you to continue the job later without having to re-run the part already transferred.

At any rate, after you've inserted a new floppy, you'll be asked for the filename into which the transfer is to continue. The amount of data you can transfer from tape to disk is limited only by the number of diskettes you have on hand.

B) DELIBERATE READING OF ONLY PART OF A TAPE FILE (Block Positioning): In normal use, FX transfers complete tape files to the disk, screen or printer. A block positioning feature (discussed also on page FX12) allows you to start (and stop) the transfer on particular blocks within a file. Here's how this feature is called into play: Instead of just entering a tape file number, you enter f(a-z) where " f " is the file number, " a " is the first block to be processed and " z " is the last block to be processed. The following examples will serve to illustrate this feature:

0(1-1) Only the first block of file 0 will be transferred. This is an easy way to get a small sample of data from tape, for example, to test an FXED control file in FXED's disk-to-disk mode.

7(34-) The first 33 blocks of file 7 will be skipped, processing will start at block 34 and proceed through the end of the file.

7(1-1000) Transfer will start with the first block of file 7 and stop after block number 1000 has been transferred.

7(34-1000) The first 33 blocks of file 7 will be skipped, blocks 34 through and including block 1000 will be transferred.

After an "ending block" has been processed, and no more blocks remain to be read, the tape is spaced in reverse to the beginning of the file.

Appending tape data to an existing disk file

During the Tape-To-Disk dialogue, FX asks you the name of the disk file into which you want to transfer the data from tape. You simply enter a disk file name. If the specified disk file already exists, FX will ask you to confirm your desire to overwrite it. Alternately, FX allows you to APPEND data from tape to the end of an existing disk file. Just add "/A" after the disk file name e.g.

```
Enter a single disk <filename>
or <filename>/A to append to an existing disk file
or <filename>/D to ...
or <F1><ENTER> for main menu:   ABC.TXT/A    (FX will complain if the disk file does not already
exist.)
```

Chop off the beginning of a block

This is a rarely used option not suggested by the dialogue. Inspection of some tapes (e.g. with the KEYS and DITTO programs) shows that the first 'n' bytes of each physical tape block contain superfluous characters. An example of this is with vestigial variable-length-block data, each block starts with an extra 4 bytes, even when the following data is fixed-length. To facilitate processing this type of data, FX provides a "CHOP" option to discard the first 'n' bytes of each physical tape block. Just add /C after the tape file number, then FX will ask an extra question, e.g.:

```
Enter tape file number [the first tape file is 0]:   2/C <ENTER>
Chop the first how many characters of each block ?    4 <ENTER>
```

NOTE: The CHOP feature is incompatible with the "Expand single CR or LF into CR/LF" and "Add CR/LF

every 'n' characters" features. The console dialogue doesn't invite the user to select them when CHOPing is in effect.

"FAST" MULTIPLE-BLOCK-MODE (the default mode) when reading tape

FX can read from tape in "single-block-mode" and "multiple-block-mode". Multiple-block mode (the default) is usually significantly faster, especially if you are reading small tape blocks. The data is read from tape at full tape speed in streaming mode without stopping the tape between blocks until a large memory buffer is filled - then the entire buffer is processed. For tape-to-disk reads, the total number of bytes in the aggregate multi-block read is displayed on the screen.

Whenever you are at the main menu or reading from tape, a reverse video panel in the lower right hand corner of the screen shows either "SBM" or "MBM" as a reminder of the mode currently in effect.

CAUTION: If you are reading a "variable-length-block" tape (infrequently encountered), you should turn off multiple-block-mode by hitting "M" at the main menu. A data loss will occur if a large block, preceded by a smaller block, crosses the PC's DMA page boundary at the end of the buffer. FX will prevent full multiple-block style reads, whenever it can detect a variable-length-block tape - but the ultimate responsibility is yours when reading these non- standard tapes.

Transfer of binary files to disk

There are no restrictions on the type of data you may transfer from tape to disk. By binary data we mean that each byte may assume any of the possible 256 permutations of 8 bits. You may find, however, that some programs running on the PC will misbehave when encountering certain characters. We went around in circles for a while with two different word processors both of which insisted on stopping right in the middle of a giant text file we had imported from a UNIX word processor via tape. We could see that the file size was right when we did a DIR, we could see all the text using DEBUG to read through the disk file but we couldn't get past a certain spot. Then, analyzing embedded control characters (which the UNIX system used for font and format control), we found the offending character to be CTRL-Z, which many MS-DOS programs use as an End-Of-File. We changed it to a space and all was well.

Display Of Physical Tape Block Size: When reading tape without the use of an FXED Control File, FX displays information about the size of the tape blocks being transferred. These displays deal with the physical tape block length, NOT the logical record length.

Reading tape with an FXED Control File- When using an FXED Control File, FX knows the logical record length, the number of records input from tape and the number of output records, which may be less if Conditional Record Acceptance is in effect. You'll see a constantly updated screen display of the cumulative number of bytes read from tape, followed by the input and output record statistics.

A large number of corrected errors usually indicates either dirty heads or tape drive incompatibility. Try cleaning the heads per the instructions in your tape drive manual. If this doesn't reduce the frequency of corrected error messages then there's either a tape drive incompatibility problem or a hardware problem. If a drive with truly clean heads pulls errors (hard or corrected) writing and reading its own tapes, there's a hardware problem. If the errors are only in reading tapes from another source, there's an incompatibility problem.

Tape read error handling

FX will try to re-read a bad tape block a number of times before it gives up and outputs a tape read error message. Should this occur it means that the tape drive hardware has detected an error. It also means that it is very likely that one or more bytes of the block just read is in error. You'll see a message of the format:

Reading tape without an FXED control file

When reading tape you probably don't need to know the size of each tape block but in some applications the block size may be useful information. In single-block-mode you'll see the size of each individual tape block on the screen, one after another. For tape-to-disk transfers in multiple-block mode, the block size displayed is the total number of bytes accumulated by each multiple-block streaming read.

DISPLAY OF FILE NUMBER AND BLOCK NUMBER: For an explanation of these numbers, displayed on the bottom of the screen, as well as some suggestions as to their usefulness, see the paragraphs entitled "DISPLAY OF FILE #, BLOCK # ETC" and "BLOCK POSITIONING FEATURES"

READ ERRORS CORRECTED BY FORMATTED TAPE DRIVES: Due to redundancy in data written on tape, modern formatted tape drives (e.g. the Cipher M995, hp 88780, 7980S, Fujitsu 2481,etc) can reliably correct certain tape read errors. This is called a "Corrected Error", and is reported by the message "READ ERROR CORRECTED BY THE DRIVE". When corrected errors are reported, the multiple-block-read is terminated early (no data is lost) so don't be alarmed if you see a smaller 'block-size' reported on the screen. It's not unusual to get a few 'corrected errors' when reading tapes from various sources. Don't worry about receiving this message occasionally. However, note that not all drives return corrected error status to the calling program.

```
   Unrecoverable Read Error(s) in block 150
   Bad block's length = 2587
   (C)ontinue, (S)kip  or (A)bort ?
```

It's up to you to decide, based on the type of data you're reading and the use it is to be put to, what you want to do. If you pick Continue, and the bad block has a bad length, it will mess up the alignment of any following fixed length records. Picking Skip (skip the bad block, then continue) might make more sense in this situation. If you continue, the possibly corrupted data will become part of the file. If you abort, then the good data already read will be processed (e.g. saved to disk if you're doing a tape-to-disk transfer) but the rest of the tape file will be ignored and the tape positioned back to the beginning of the file.

Dialogue when writing to tape

"Enter disk <filename> or names [e.g. ABC.TXT   LILY   *.BAS   XYZ.?]

To transfer a single file from disk to tape enter the file name, with or without drive and directory specifiers, just as you would with any DOS command, then hit <ENTER>. For Example:

```
CENSUS.DAT <ENTER>
C:\NAMES\GIRLS\BLONDES.TXT <ENTER>
```

FX allows you to transfer more than one file to tape without going through the Disk-To-Tape dialogue for each file. See "MULTIPLE FILE TRANSFERS" on the next page .

`"Do you want to TRUNCATE or EXPAND the records you're transferring to tape?"`
Answer "Y" to access these powerful features which allow you to delete <CR><LF>'s, block out variable length records to fixed length and create fixed length records on tape which are larger or smaller than the fixed length records on disk. Details on page FX9.

`"Enter starting tape file # [0-9999] or <ENTER> for END-OF-DATA"`
Simply enter the number of the tape file you want to write to, then hit <ENTER>. Remember, the first file on a tape is, by convention, file # 0.

END-OF-DATA: It is common practice for minicomputers and mainframes to use two adjacent filemarks (tape marks) to indicate the end of all the data on the tape. FX follows this convention. If you hit <ENTER> alone instead of a tape file number FX will space forward past the two filemarks, then backspace over the second filemark, write the requested disk file(s), followed by a new set of two filemarks.

When writing to a tape for the first time (i.e. a new or bulk erased tape) the only acceptable answer to this question is file 0. If, for example, you answer file 3, the tape drive controller will attempt to move the tape forward counting the filemarks at the end of files 0, 1 and 2 - then stop, ready to write file 3. This will work only if files 0-2 already exist on tape. If it were a virgin tape, the drive would slew tape (attempting to count past 3 file marks) until the EOT (end-of-tape) marker was hit.

Astute users, who want to start writing at some point other than the first block of a file, may make use of FX's block positioning feature. Use of this feature is fully explained in the section entitled "BLOCK POSITIONING FEATURES" on page FX12.

"Append disk file(s) to an EXISTING tape file? (Y/N)"

FX allows the user to merge any number of disk files into a single tape file. The answer to this question determines whether FX will create a new tape file or append the disk data to the end of an existing file. To create a new tape file, answer with "N". To append to the end of an existing tape file answer with "Y". In this case, FX will position the tape just past the last data block of the file, ready to overwrite the old filemark with one or more new data blocks followed by a new filemark.

Notes about appending

When you write the first part of the tape file, to which you will append more disk files later, you must answer "N" to the "APPEND TO FILE?" question. Then answer "Y" for each subsequent transfer to the end of the same tape file.

As described below, FX normally pads the last block with spaces as necessary to fill it out to size. Keep in mind that if you allow these pads, they will become "internal" parts of your file during an append operation. The pad characters can be avoided by use of the "/S" option when the block size is specified or, of course, by planning ahead and judiciously specifying the block size.

FORMAT OF DISPLAYED BLOCK NUMBER WHEN APPENDING TO A TAPE FILE (+NNNNN):

If the user selects the APPEND feature when writing to tape, the controller does a high-speed search for the filemarks while ignoring data blocks. Since the number of blocks passed over is not available, FX can not display the actual block number of each block it appends to the file. Instead, the relative number of each newly written block is displayed, preceded by a "+" sign as a reminder. For example, if file 5 has 100 blocks and you append 15 more the display at the bottom of the screen will read "FILE # 5 BLOCK # +15" when processing is completed.

Padding last block with space characters

When writing a tape file, FX pads the end of the last block with SPACE characters as necessary to maintain the user- specified block size. This feature may be circumvented by using the /S option when the block size is specified - see below.

"Enter BLOCK-LENGTH alone for last block padded with SPACES,

or BLOCK-LENGTH/S to allow short last block: "

Here is where you specify the length (in bytes) of each physical block to be written on tape. In deciding on what block length to use there are several things to be considered. First, if the tape is going to be read elsewhere, you should ask the intended recipient what size blocks he would like.

If someone else has not specified the block size, keep the following in mind: ANSI standards specify block sizes of less than 30 bytes are to be avoided. Standards also specify that the maximum size block for interchange is 2048 bytes, unless larger by mutual agreement - and larger blocks are a common exception to the rule. The Electrovalue software can read and write blocks up to 65535 bytes (64K - 1). If you request a block size larger than 64K-1, FX will remind you of the maximum size. Of course, your hardware may limit your block size too.

Short last block

Normally, FX will write each tape block of a file the same length, filling out the last block with ASCII or EBCDIC SPACE characters as necessary to maintain the specified block size. This feature can be disabled by following the block size with /S, in which case the last block will contain only the last characters, with no padding characters.

EXAMPLES OF BLOCK LENGTH ENTRY:

```
                Enter BLOCK-LENGTH...: 12345 <ENTER>
                Enter BLOCK-LENGTH...: 56789/S <ENTER>
```

ASCII -to-EBCDIC conversion

"Convert ASCII to EBCDIC ?"   If you answer "N", the disk data will be written to tape EXACTLY as it is on disk, bit- for-bit, byte-for-byte; i.e. what is called a BINARY transfer of data.

END-OF-TAPE MARKER (Tape Full):

The end of the physical tape is indicated by a reflective marker called End-Of-Tape or EOT for short. FX allows the transfer of large disk files to multiple reels of tape. If EOT is hit, you'll be prompted to mount a new tape so the disk to tape file transfer can continue.

Multiple file transfer

You can answer the "Enter disk <filename>" question with more than one file name (put a space between each file name and hit <ENTER> at the end of the list) and you can specify wildcard characters (* or ?), or a combination of both. Any answer except a single file name results in the following question:

"Use a separate tape file for each disk file ?"   If you answer "N", all the disk files will be appended into one tape file. Answering "Y" will put each disk file in a separate tape file. As the transfers take place you'll see the details (file names, tape file numbers etc.) on the screen. Here's a sample session:

```
Enter disk <filename>, or names [e.g. ABC.TXT  LILY  *.BAS  XYZ.?]: *.TXT
Use a separate tape file for each disk file? Y
Enter starting tape file # [0-9999] or <ENTER> for END-OF-DATA 0
Convert ASCII to EBCDIC? Y
Enter BLOCK-LENGTH alone for last block padded with SPACEs, or BLOCK-LENGTH/S to allow short last
block:  65535/S
Transferring disk file FXPN.TXT to tape file # 0.   34374 bytes in disk file.
Transferring disk file SUPN.TXT to tape file # 1.   28395 bytes in disk file.
File Transfer(s) Complete.   2 File(s) transferred from disk to tape.   Press any key to continue
```

Truncation & expansion

DISK TO TAPE - TRUNCATION AND/OR EXPANSION FEATURES

These features provide you with a helpful means of creating mainframe friendly fixed length logical records on tape from certain types of data. As mentioned previously, during the disk to tape dialogue you'll be asked the question:

"Do you want to TRUNCATE or EXPAND the records you're transferring to tape?"

If you answer "Y" for yes, FX will ask additional questions that will let you define exactly what you want to do. Before the sample dialogue is presented, we'll discuss the four possible combinations of features and explain how each operates.

(1) FIXED LENGTH Logical Records on Disk   If your disk file contains FIXED LENGTH LOGICAL RECORDS you can change the length as the records are being written to tape.

TRUNCATION:

You can chop the end off of each disk record, e.g. to delete trailing <CR><LF>'s. It's likely that your PC database program creates "FLAT ASCII" (SDF) data as records ending in <CR><LF>. It's just as likely that the mainframe does not want to see records ending in <CR><LF>. Suppose your records on disk are 82 bytes long, the last 2 characters being <CR><LF>. You can specify an input record size of 82 and an output record size of 80 to chop off the <CR><LF>'s.

EXPANSION:

You can add SPACE's or NUL's (ZERO's) to the end of each fixed length disk record if you want to pad each record out to a larger size.

(2) VARIABLE LENGTH Records on Disk: If your disk file contains VARIABLE LENGTH RECORDS ending in <CR><LF> you can change them to fixed length logical records as they are being written to tape.

EXPANSION: FX will search through the text from the disk file until it finds <CR><LF>, delete the <CR><LF>, then pad out the record with SPACE's or NUL's (you choose which) to whatever output record size you have specified.

TRUNCATION: If you've told FX that the records on disk are variable in length and FX finds <CR><LF> at a byte position greater than the output record size you've specified, you'll loose (on tape) all the bytes in the disk record after the output record size up through and including the <CR><LF>. We don't have a suggested use for this mode of operation but that's how it works and it's documented here for the sake of completeness. If, at some point in the disk file, there are characters NEVER followed by <CR><LF>, they won't be written to tape. If it should happen that there are no <CR><LF>'s at all in the disk file, you'll get an empty file on tape, i.e. just a filemark.

SAMPLE DIALOGUE:

```
Do you want to TRUNCATE or EXPAND the records you're transferring to tape? Y
Type input (disk) record length or 'V' for variable; hit <ENTER>: V
Records from disk are variable length ending in <CR><LF>. Type output (tape) logical record length and
hit <ENTER>: 130
Pad with (S)paces or (Z)eros? (S/Z) S
Convert ASCII to EBCDIC? (Y/N) N
Treat CTRL-Z in a disk file as End-Of-File ? (Y/N) Y
Enter block length: 1300/S
Enter starting tape file # [0-9999] or <ENTER> for end of data: 0
```

NOTES: The blocking factor must be an integer. That means that the block length on tape must be the output logical record length times an integer. FX will not accept a non-integral blocking factor. The output logical record length must not exceed 32,000.

DISPLAY OF PHYSICAL TAPE BLOCK SIZE WHEN WRITING TAPE: FX increases the throughput of formatted tape drives during disk-to-tape transfers by filling up to 64K of RAM with one disk read, and then writing the individual tape blocks in one 'streaming' burst. In this case, the screen display shows the INDIVIDUAL BLOCK SIZE times THE NUMBER OF BLOCKS, once per I/O buffer written to tape (e.g. 7980 x 8 7980 x 8 7980 x 8 1596

Log to line printer

TAPE to DISK and DISK to TAPE ONLY

This feature is turned ON/OFF by hitting the * key (<shift><8>) while you're in the main menu or during the initial dialogue of Tape to Disk or Disk to Tape. While log mode is active, you'll see a highlighted "L" in the lower right hand corner of the screen.

Don't enable the log mode if you don't have a printer or if the printer is turned off. This will cause FX to hang up the PC and eventually do a timeout abort when it's time to print.

If you log to the printer you'll get a printout containing all the important details of the job. Store it with the resulting disk or tape to avoid making sticky labels.

Partial sample logs follow, preceded by the console dialogue that caused them.

DISK TO TAPE:

```
Enter disk <filename>, or names [e.g. ABC.TXT  LILY  *.BAS  XYZ.?]:  *.TXT
Do you want to TRUNCATE or EXPAND the records you're transferring to tape ? N
Use a separate tape file for each disk file?  Y
Enter starting tape file # [0-9999] or <ENTER> for END-OF-DATA 0
Convert ASCII to EBCDIC?  Y
Treat CTRL-Z in a disk file as End-Of-File ? Y
Enter BLOCK-LENGTH alone for last block padded with SPACES,
 or BLOCK-LENGTH/S to allow short last block: 44444/S
Transferring disk file FXPN.TXT to tape file # 0  36138 bytes in disk file
   ...   Transferring disk file XYZ.TXT    ...   ...etc.
Transferring disk file INSTPN.TXT to tape file # 5  12299 bytes in disk file
File Transfer(s) Complete.    6 File(s) transferred from disk to tape.
```

TAPE TO DISK:

Note that the user asks for multiple tape files to be transferred into a single disk file. This causes FX to append all the tape files into the disk file, "\TEMP".

```
Enter tape file number [the first file on tape is file 0]: 0-5
Reset 8th bit ? N
If Input Data processing criteria are defined in an FXED control file,
input <filename>, else press <ENTER> for no control file:  <ENTER>
Replace all control characters (00-31 dec.) with SPACEs (32 dec.) ? N
Change all CR's and/or LF's into the CR-LF pair ?   N
Convert EBCDIC to ASCII?  Y
Add CR/LF after how many char? (<ENTER> alone = none)  <ENTER>
Enter a single disk <filename>,
 or <filename>/A to append to an existing disk file,
 or <filename>/D to append to an existing .DBF file,
 or hit <F1><ENTER> for main menu:  \TEMP
Transferring tape file # 0 to disk file \TEMP   ...   Appending tape file # 1 to
disk file \TEMP   ...   ...etc.
Appending tape file # 5 to disk file \TEMP.   File Transfer(s) Complete
```

ASCII-EBCDIC and EBCDIC-ASCII CODE CONVERSION:

A- When transfer is from tape (and you have not picked the "Reset 8th bit" option) the following question will be asked:

Convert EBCDIC to ASCII? (Y/N)

If the data on tape is written in pure EBCDIC (doesn't contain PACKED DECIMAL nor ZONED DECIMAL - see "4/" below) you will surely want to answer with Y for Yes. This will cause the EBCDIC data to be translated to ASCII (on a character- to-character basis) before it continues on its way to the PC's screen, printer or disk. Otherwise answer N.

NOTES: {In the following discussion codes are shown as (decimal, hex)}.

FX handles three special situations in EBCDIC-ASCII conversion ...

EBCDIC lower case numbers (176,B0) to (185,B9) are translated to the corresponding ASCII numbers (48,30) to (57,39).

There are undefined gaps in the 256 possible 8 bit EBCDIC codes. If an illegal EBCDIC code (e.g. 203,CB) is read from tape, an ASCII UPARROW (94,5E) is substituted.

If a legal EBCDIC code which has no reasonable ASCII equivalent (e.g. 26,1A =CC) is read, an ASCII BACKSLASH (92,5C) is substituted.

FX itself makes no special provision for this 4th case, but Electrovalue's FXED program allows you to handle it effectively: "EBCDIC" tapes (and sometimes ASCII tapes) may contain numerical data encoded as Packed Decimal, Zoned Decimal and/or Packed HEX/BCD. FX can translate these to ASCII after you identify them (field-by-field) using the FXED program. See the "Translation Definition Phase" of the FXED chapter of this book for full details.

B- When transferring data to tape the following question will be asked:

Convert ASCII to EBCDIC ? If you answer N, the disk data will be written to tape without any changes, i.e. bit-for-bit, byte-for-byte (this is called a BINARY transfer). Answer Y to produce an EBCDIC tape.

1. There are six ASCII characters that have no EBCDIC equivalent. These are translated into an EBCDIC BACKSLASH (224,E0). They are:

ENQ (5,05) DC3 (19,13) GS (29,1D) US (31,1F) ` (96,60) | (124,7C)

2. ASCII is a 7-bit code, defined by an official standards group. ASCII characters can assume decimal values from 0 - 127. The PC allows what some call "Extended ASCII" where the 8th bit is set, creating characters with values from 128 - 255 decimal. There are no standards with respect to these characters, neither do they have EBCDIC equivalents. If you ask FX to convert them from ASCII to EBCDIC, the eighth bit will first be masked off, then the translation will be performed. This has the effect of subtracting 128 from the value of the "Extended ASCII" character before translation to EBCDIC. If you need the ability to define your own custom translation table to handle these codes, see FXED's D (Disk-To-Disk) and S (Secondary Translation) modes.

Tape to screen and tape to printer

Special "CHARACTER PROCESSING": To create friendly displays, Tape-To-Screen and Tape-To-Printer pre- process the data. The printable characters (decimal 33-126, hex 21-7E) are displayed directly. Carriage Return and Line Feed are passed through. All other characters are displayed as decimal points, NOT as their screen or printer graphics. If you want to pass tape data through to a printer without this character processing, see "LASER PRINTERS, HIGH SPEED" on the index page.

Testing an fxed control file

You can use one of these modes to help fine-tune an FXED control file but don't overlook FXED's 'Disk-To-Disk' mode which does a more efficient job. Just import a small sample of raw tape data with FX (don't select any options which would change the tape data). Use the FX option to read only part of a tape file to keep your sample small (e.g. answer the tape file question with 0(1-10) <ENTER> .

Block positioning

DISPLAY OF FILE #, BLOCK # ETC.

The block positioning features of FX (described below) allow transfers to or from tape to start at any block within a file.

Whenever reading from or writing to tape, FX displays a file number and block number in a protected area at the bottom of the screen. File numbers on tape start with 0; block numbers within a file start with 1. The block number is incremented after the block has been successfully read or written.

BLOCK POSITIONING FEATURES (IMPORT ONLY A RANGE OF BLOCKS FROM TAPE):

The block positioning feature may be used whenever FX requests a tape file number. The normal response to a file number request is for the user to enter a file number and hit <ENTER>. The block positioning feature is automatically invoked if the file number entry is in the format " f(a-z) " where " f " is the file number, " a " is the block with which to start the transfer, and " z " is the block with which to stop the transfer. For example, entering the file number 0(22-33) would transfer file 0, blocks 22 through 33 inclusive. This is a nice way to capture a small sample of data for test purposes before running a large job.

When transferring from tape to disk you may input multiple file numbers on one line. For example, "0 2 4 6-9 11 12(123- 456) 13(333-444)" is a valid input. Block positioning file specifications may not be immediately preceded or followed by a "-"; e.g. "0(11-22) - 6(7-11)" and "2 4 5 - 12(23-34)" are not valid inputs. Specify "0(11-22) 1-5 6(7-11)" and "2 4 5-11 12(23-34)" respectively to work around the invalid examples respectively.

ENTERING A LARGER NUMBER OF BLOCKS THEN A FILE CONTAINS:

When doing tape positioning, the tape drive first does a high speed search either forward or reverse to the beginning of the file requested, then searches forward while counting blocks. Any tape errors are ignored and hitting a filemark causes spacing by blocks to halt. If you erroneously enter a block number which is larger than the number of blocks in the file you intend to read or write, processing may begin in some other file.

Ctrl-Z

CTRL-Z THE PC's "END-OF-FILE" CHARACTER

HOW IT MIGHT CAUSE TROUBLE, HOW TO FIND IT AND WHAT TO DO ABOUT IT:

DOS allows the use of the ASCII character "SUB" to mark the end of a text file. This is not an industry-standard convention, it is an IBM-PC convention. SUB is also called CTRL-Z because it can be generated from the keyboard by hitting CTRL-Z. SUB is a control character defined in both the ASCII (26 dec., 1A hex) and EBCDIC (63 dec., 3F hex) character sets. Many PC programs (e.g. word processors and database programs) throw in one (or many) CTRL-Z's at the end of a text file without even telling you. To make matters worse, you can't see the CTRL-Z if you TYPE or PRINT the file since these commands automatically terminate when the CTRL-Z is encountered without showing the CTRL-Z on the screen or printer. FX's mission is to read and write industry-standard magnetic tapes, and FX can transfer ALL the characters of a file between disk and tape without prejudice. If you send a tape to a mainframe shop and are told it has one or more garbage characters at the end of the tape file they are likely to be CTRL-Z's.

SUGGESTION: CHOOSE TO STOP ON CTRL-Z WHENEVER TRANSFERRING TEXT FILES TO TAPE

Normally, when exchanging data with other computers, you'll be sending ASCII or EBCDIC text files. In this case, it's usually the best choice to stop on any CTRL-Z that FX finds. It's true that lots of files on your disk contain CTRL-Z characters (or binary characters which are the same code) which are not intended to be interpreted as "end-of-file". These include most program files and most graphics files. But it's unlikely that you'll want to send binary data to another computer using FX.

The question is asked: "Treat CTRL-Z in a disk file as END-OF-FILE ?

If you select this option (by hitting "Y" for Yes), any CTRL-Z in the disk file will be interpreted as "End-Of-File", neither it nor any subsequent characters in the disk file will be written to tape. "Yes" is usually the best choice when writing TEXT files to tape. Don't answer "Yes" if you are writing binary files to tape.

The DOS COPY command provides an option that allows you to see CTRL-Z's on the screen. We'll leave it to you to read your DOS Reference Manual to understand the COPY options. The following example shows how to see CTRL-Z characters on the screen (displayed as a right-pointing arrow):

```
COPY  SOURCE.TXT/B  CON  <ENTER>
```

Tape to dBASE .DBF disk files

Using FX to directly import tape data into dBASE ".DBF" files

STRUCTURE OF DATA ON TAPE: Typical mainframe tape data, provided for data interchange between computers, consists of fixed length data records, end-to-end, without any extra "record separator" characters. Each fixed-length-record is built up from individual, fixed length fields. Physical blocks of data on tape contain an integer number of records - i.e. logical records do not span physical blocks.

APPENDING DIRECTLY FROM TAPE TO A ".DBF" FILE: FX has an option which allows you to append "fixed-length- record" tape data (as described above) directly to a .DBF database disk file.

The most popular database internal format in the MS-DOS world is the .DBF file. .DBF files are used by all versions of dBASE, FoxBASE, FoxPRO, Clipper, dBXL, Quicksilver, AlphaWorks and many workalikes and compatible utilities.

Generating the .DBF file): FX itself is not used to Create the .DBF file. You do that with your database program or, optionally, with the FXED program's "G" command (see the FXED chapter).

FX has the ability to add tape data to any existing .DBF file. The .DBF file may be an empty 'STRUCTURE' or a .DBF file that already contains data records. FX processes fixed-length tape fields into a .DBF compatible format as well as doing all the necessary "bookkeeping" to update the modification date and other control information in the .DBF file. FX supports dBASE's Character, Numeric, Date and Logical field types. However, there is no direct way for FX to fill Memo fields and their associated ".DBT" files.

Successful transfers depend on a perfect match between the order of the fields and the field lengths coming in from tape (after any optional processing by an FXED produced control file) and the fields defined for your database. When using FX to transfer data directly into a .DBF file, DO NOT add CR/LF at the end of each record.

The 2 steps used in importing data from tape into dBASE ".DBF" files

Importing mainframe tape data directly to a .DBF database disk file consists of 2 steps:

STEP 1. Define the database structure using your database software:

Use the 'CREATE' command and fill in the Field Names, Field Types and Field Widths.

To keep things simple, define all the fields as Character type fields. Let's call our example database DEMO.DBF. See below for information on Numeric, Date and Logical field types.

To keep things simple, the Field Widths you select in dBASE must be exactly the same as the field lengths in the corresponding fields on tape. The field lengths on tape should have been provided to you on paper, along with the tape. See below for hints in dealing with tape field lengths that are different than the dBASE Field Widths.

To keep things simple, the order of the fields you create in dBASE must be the same order that they are on tape. See below for pointers on how to re-order the fields as they are transferred in from tape.

To keep things simple, you must define a field in your database structure for each field in a tape record, whether you want all the tape fields in your database or not. See below to learn how to import only those fields of interest.

After you have created a dBASE structure you can list it by hitting <F5> in dBASE or using either the 'DISPLAY STRUCTURE' or 'LIST STRUCTURE' commands. An example of some of what you'll see is:

```
  Field   Field Name      Type           Width
      1   FNAME           Character      15
      2   LNAME           Character      20
      3   CNAME           Character      32
      4   STREET          Character      36
      5   CITY            Character      25
      6   STATE           Character       2      Important Note:
      7   ZIP             Character       8      The ** Total ** displayed by dBASE (234 below) is one
      8   PHONE           Character      16      more than the sum of the number of characters in all
the
      9   DATE            Character       8      fields.  The record lengths we'll deal with are one
less
     10   NOTES           Character      50      than this number; i.e. the logical record lengths
     11   SAW_AD_IN       Character      15      on tape.
     12   CUST_NMBR       Character       6
** Total **                              234
```

STEP 2. Use the FX program to append the data from tape to your .DBF disk file:

Select "Tape-To-Disk" mode from the main menu. Answer all the questions but the last (disk file name) in the normal fashion. Select EBCDIC to ASCII translation or invoke an FXED produced control file etc. when appropriate. DO NOT choose to add a CR/LF after every record. The last question FX asks during the Tape-To-Disk dialogue includes:

"Enter...<filename>/D to append to an existing .DBF file"

Type the name of your database disk file, followed by the 2 characters /D The .DBF extension may be omitted, e.g.:

DEMO.DBF/D <ENTER>   demo.dbf/d <ENTER>   DEMO/D <ENTER>   demo/d <ENTER>

All 4 of the above examples are equivalent and assume the prior existence of a disk file named "DEMO.DBF".

The /D can be adjacent to the filename or separated by SPACE(s). If neither a filename extension nor a DOT (.) is present, the filename extension .DBF is assumed and automatically added by FX.

As soon as the <ENTER> key is hit, the Tape-To-Disk data transfer will commence. Statistics about the .DBF file are shown on the screen before and after the append operation.

NOTES: This section mentions some of the advanced features made available to the FX file exchange program by Electrovalue's FXED program. This powerful program need not be used, nor understood, if you don't want to take advantage of these (and other) advanced features [which include the automatic generation of an empty .DBF structure].

Changing field lengths:

Using FXED, in the Output Definition Phase, you can create the exact field widths you want to match an existing database structure. You can truncate tape fields if necessary by not defining some of the (probably rightmost) bytes for output. You can expand fields by using the TAB function or by inserting SPACES (or other characters of your choice) as filler characters.

Changing the order of fields:

Using FXED, in the Output Definition Phase, you can scramble the fields into any order you like to match an existing database structure. They will end up in your .DBF file in whatever order you define them for output.

Discarding unwanted fields:

Using FXED, during the Output Definition Phase, you define the tape fields that (at run time) FX will transfer to the disk. Any and all fields not defined for output will be ignored by FX. Thus you can import only those fields that interest you and not clog up your database (and hard disk) with extraneous data.

dBASE numeric, date and logical data types:

.DBF Numeric Fields: Internally (from left-to-right), DBF numeric fields consist of an optional minus sign ("-"), the digits to the left of the decimal point, the decimal point itself and the digits to the right of the decimal point. The maximum number of positive digits and the decimal point add up to the Field Width shown by your database program's "DISPLAY STRUCTURE" command. The Numeric Field 1234.56 (Field Width 7) in a .DBF file is normally stored on tape as 123456, with the tape field definition paperwork telling you where the mainframe assumes the decimal point. To import this string properly, use FXED's Output Definition Phase, literal text insertion feature, to change the number on tape into 3 adjacent fields: the first 4 digits, a decimal point literally inserted, then the last 2 digits.

.DBF Date Fields are 8 characters long. When displayed (e.g. by the 'EDIT' command), the extra two characters are / (slash) characters, used as separators. Date fields stored internally in the .DBF file are still 8 characters long, but the slashes are absent - instead the most significant digits of the year are present. For example, the 8-character .DBF date (as shown by the 'EDIT' command) 11/22/88 is represented internally as the 8 characters 19881122. Usually, mainframe tapes represent dates as MMDDYY. Use the Electrovalue KEYS program to make a quick inspection if the paperwork that came with the tape doesn't make it clear. You can use the Output Definition Phase of FXED, along with its ability to define literal text insertions (e. g. the missing "19") to instruct FX to change the order of things and insert the "19" when the Tape-to-Disk job is run.

.DBF Logical Fields are one byte long and contain either the character T (for True) or F (for False). It's not likely that you'll find T's and F's on mainframe data tapes but you can easily add them as "place-holders" using the INSERT function during FXED's Output Definition Phase.

Index files associated with a .DBF file:

Appending tape data to a .DBF file has no effect on any INDEX (.NDX, .IDX etc.) files which may be associated with the .DBF file. If your application requires index files, be sure to rebuild them after adding tape data. Updating index files is sometimes transparent to the user who is adding data via the PC's keyboard.

Tape to .dbf error & warning messages

RECORD LENGTH PROBLEMS

Remember, the record length from tape, after possible processing by an FXED produced control file, must match the record length used by the .DBF file. Without a control file, the blocking factor on tape must be an integer - i.e. the .DBF record size must divide evenly into the physical block length. To prevent bizarre results, whenever it has enough information to do so, FX checks the reasonability of the situation before proceeding. You'll see one of the following messages whenever FX detects something that seems like it would cause harm and/or nonsense results:

Record length in DEMO.DBF 'header' is 478 but control file CF would produce records of length 500 - processing can not proceed.

Try again. Enter a disk <filename>....or <F1><ENTER> for main menu.

The .DBF record length must divide evenly into the tape block size.

It doesn't. The current .DBF record length = 478 and the tape block

length = 5090. Hit <ESC> to try again.

DEMO.DBF does not appear to be a valid .DBF file. Based on the data in

the 'header', DEMO.DBF should be 456 bytes long.

DEMO.DBF is actually 656 bytes long. Length difference = 200.

If the .DBF file is longer than FX thinks it should be, this message is considered a warning and you'll be given the option to continue anyway. Some dBASE workalikes produce .DBF files with an extra wob of garbage at the end. FX will add records at the 'logical' end of the DBF file, overlaying any garbage. If the .DBF file is shorter than it ought be, there's no hope - you'll get the following message:

Try again. Enter a disk <filename>....or <F1><ENTER> alone for main menu.

If you are running without a control file, in tape to .DBF mode, and pick one of the FX options which adds CR/LF or changes single CR's or LF's into CR/LF you'll get the following error message (records in .DBF files aren't supposed to end in CR/LF).

Above requested CR/LF option not valid when transferring to a .DBF file.

Press any key to continue.

OTHER FX ERROR MESSAGES WHICH NEED EXPLAINING

FX is written in Microsoft QuickBasic and assembly language. For unknown reasons, QuickBasic takes over some of the (already cryptic) DOS error messages and obfuscates them even more. You could see the following messages, here's what they mean:

If you try to write to an open file on a network, or an open (shared) file on a computer, or if you try to write to a disk file which DOS or Windows can not allow, you'll get:

Permission Denied in line nnnn of module FX at address ABCD:EGFF

Hit any key to return to system.

This can result when you attempt to transfer from tape to: (1) A Write-Protected diskette; (2) A Read-Only disk file; (3) A file which is in use {open already} on a network; (4) A <filename> which is a directory name off the current or specified directory.

FX requires a good deal of free memory. If you have much less than 640 Kb of ram or if you have many tsr's running or if you are shelling out from another program, you may get this message. If there's enough memory for the tape I/O buffers, but not enough memory for internal QuickBasic buffers you'll see:

Out of string space in line nnnn of module FX at address ABCD:EF00

Hit any key to return to system.

If you try to transfer a bad disk file (disk directory or file is unreadable by DOS) to tape you'll get:

Device I/O Error in line nnnn of module FX at address ABCD:EF00

Hit any key to return to system.

Batch mode-recording and playback of FX user input:

Would you like to easily run the same FX job again and again? Would you like to set up FX jobs to be run from beginning to end by inexperienced personnel with a single command line entry? You can automatically record all the keystrokes you enter during an FX "session" and play them back at any time. Here's how it works:

To record all your input in a file simply enter, at the command line:

FX R:filename <ENTER> e.g. "FX R:DAILYRUN <ENTER>"

The job will actually be run and all the keystrokes you enter will be recorded in the file you have specified on the command line.

To playback (run the same job again), simply enter, at the command line:

FX P:filename <ENTER> e.g. "FX P:DAILYRUN <ENTER>"

NOTES: You can edit the command file with your favorite pure ASCII word processor. The file will contain all your keystrokes, with added carriage returns for ease in reading. If you've hit the * key to force line printer logging of the job, it will show up in the command file as a diamond (decimal 04).

Tape to printer mode

Part of FX's "TAPE-TO-PRINTER" routine is fast enough to keep normal laser printers busy, but may not be fast enough to feed fast, high-speed commercial laser printers (e.g. Xerox 9700) at their rated throughput. Furthermore, "TAPE-TO-PRINTER" does some special 'character processing' to replace control characters with decimal points. If you want to pass control characters through, or if your application requires driving a Xerox 9700 class laser printer at maximum speed, go to the main FX menu and select "TAPE-TO-DISK" mode (written in assembly language). Then, specify PRN as the name of the output 'disk file'. PRN is DOS's reserved name for the line printer.

Below is a sample dialog for Tape to Disk transfer. User input is shown in **bold.** Comments in *italics*.

```
TAPE TO DISK     <enter> for main menu
Enter tape file number [the first file on tape is file 0]:   0
Reset 8th bit? N
If Input Data processing criteria are defined in an FXED control file,
input <filename>, else press <ENTER> alone for no control file: <ENTER>
Replace all control characters (00-31 dec.) with SPACEs (32 dec.)?(Y/N) Y
Change all CR's and/or LF's into the CR-LF pair? (Y/N) N
Convert EBCDIC to ASCII? (Y/N)Y
Add CR/LF after how many char? (<ENTER> alone = none)  80 <ENTER>
Enter a single disk <filename>,
 or <filename>/A to append to an existing disk file,
 or <filename>/D to append to an existing .DBF file,
 or hit <F1><ENTER> for main menu:  ABC.TXT <ENTER>
Transferring tape file # 0 to disk file ABC.TXT
 19680 19680 19680 19680 19680 19680 19680 19680 19680 19680 320
File Transfer(s) Complete
Press any key to continue
```

Below is a sample dialog for Disk to Tape transfer. User input is shown in **bold.** Comments in *italics*.

DISK TO TAPE **<ENTER>** *for Main Menu*

```
Enter disk <filename>, or names [e.g. ABC.TXT  LILY  *.BAS  XYZ.?]:. \GAMES\*.BAS
Use a separate tape file for each disk file?(Y/N) Y
Do you want to TRUNCATE or EXPAND the records you're transferring to tape ? (Y/N)N
Convert ASCII to EBCDIC? (Y/N) N
Append disk file(s) to the END of an EXISTING tape file? (Y/N)  N
Treat CTRL-Z in a disk file as END-OF-FILE? (Y/N) Y
Enter BLOCK-LENGTH alone for last block padded with SPACES,
 or BLOCK-LENGTH/S to allow short last block: 9999/S
Enter starting tape file # [0-9999] or <ENTER> alone for END-OF-DATA: 11 <ENTER>
Transferring disk file \GAMES\ART.BAS to tape file # 11  1879  bytes in disk file
 1879
Transferring disk file \GAMES\BALL.BAS to tape file # 12  1966  bytes in disk file
 1966
Transferring disk file \GAMES\MUSICA.BAS to tape file # 19  13431  bytes in disk
file
 9999 3432

Transferring disk file \GAMES\PIECHART.BAS to tape file # 20  2184  bytes in disk
file
 2184
Transferring disk file \GAMES\PYRAMID.BAS to tape file # 25  6945  bytes in disk
file
 6945
File Transfer(s) Complete.
 15 File(s) transferred from disk to tape.
```

The FXED Program

```
Press any key to continue
```

PURPOSE: The FX Editor (FXED) is used to create (and modify) FXED control files. FXED control files provide field-oriented runtime instructions to any of several Electrovalue programs to facilitate advanced, extensive processing of fixed-length (mainframe) data records. The powerful FXED control file features may be invoked during transfers from Tape-to-Disk (FX Program), Disk to Disk (internal to FXED) and Tape-to-Tape (optional Tape Copy Program).

LESS ADVANCED IMPORTING OF TAPE DATA WITH FX DOES NOT REQUIRE THE USE OF FXED.

Use FXED to take advantage of one or more of the following features:

- Delete unwanted fields from each input record to save space on your disk.

- Print mailing labels without phone numbers, etc.

- Conditionally accept or reject entire input records based on content.
- Import only the data you're interested in, eg, a range of zip codes for a mailing list.
- Break up large tape files into multiple disk files for separate processing.
- Translate ASCII from zoned decimal, packed decimal, packed hex, and / or packed BCD
- Stop processing the tape when a data record meets a condition you've defined. We call this feature "logical end of file". It's a big time saver when you know the order of the data on the tape.
- Insert literal text strings anywhere in each output record.
- Add the field delimiters required by your application program. Add decimal points to currency fields.
- Define single-key macros for frequently inserted text.
- Add extra blank fields to match a particular database structure.
- Move input fields to different positions in the output record.
- Match tape data fields to existing database structures.
- Print labels "4-up" directly from tape.
- Delete trailing space characters from selected input fields to save space on your disk. Condese the data being copied to floppies.
- Delete leading spaces, leading zeros and trawling spaces from selected input fields.
- Extract the significant part of numeric fields for cleaner indexing.
- Force a secondary translation to your custom character set:
- Change double quotes coming in from tape to single quotes so your application program doesn't think they are character field delimiters.
- Force upper case and / or lower case output from selected input fields.
- Force lower case output except for the first character of each word.
- "Tab" (back-fill with spaces) to particular output byte positions.

FXED is not needed for tape-to-disk transfers where the features you need to process your input data are record- oriented rather than field-oriented. You don't need FXED if you job is limited to straight EBCDIC to ASCII translation, adding <CR><LF> at the end of each logical record, converting all control characters to spaces etc. These and other record-oriented and file-oriented features are invoked by answering questions in the dialogue when you run the FX tape/disk file exchange program.

INTRODUCTION: Whenever you run FX, if you specify that you are transferring data from tape (to disk, screen or printer), you'll be asked if the input data processing criteria are defined in an FXED control file. You either input the name of the control file (which you produced with FXED previously) or hit the <ENTER> key alone to continue with the FX console dialogue which will let you select FX's built-in, less powerful, input data processing features.

The control file which FXED produces tells FX how to process (translate) each byte in each fixed length record on tape, which input data fields will become part of the output records and in which order, where to insert optional literal text in each output record, whether entire records from tape are to be conditionally accepted or rejected, whether to stop processing based on encountering specific data from tape and whether special secondary translation is to take place.

It is assumed that you have, on paper, a description of the logical record layout of your tape data. You figure out, before running FXED, the starting and ending byte numbers (or starting byte and length) of each field in an input record and what type of translation (if any) you will want FX to perform on each field from tape. You plan the order in which input fields are to appear in the output records, any literal text you want to insert between fields, any conditional record acceptance criteria and whether you want to process to the end of the input file or stop when the data meets a particular criterion.

FXED WAS DESIGNED TO BE AS EASY AS POSSIBLE TO USE.

If you have a clear understanding of the data format on your tape and what you want to see for output, you'll love FXED for its straightforward, clear and quick approach to letting you define the job to be done.

"Disk-To-Disk" Mode: If things don't work just right the first time, you'll enjoy the ease with which you can use FXED's "INPUT DATA FROM DISK" mode to view the results on the screen or printer (without leaving FXED), go back and make changes to the existing control file and try again until you're satisfied. Independent of its use in the magnetic tape environment, disk-to-disk mode makes FXED a powerful, general-purpose data-formatting program. See pages 20-22.

Using FXED to generate an FXED control file consists of several distinct phases:

Tell FXED what name to give to the control file (the filename extension ".CF" is assumed and need not be input) and how many bytes are in each fixed length logical record from tape (i.e. the input record length).

Specify what type of translation (or no translation) FX is to perform on each input record field.

"Output Field Definition:" You specify which fields from the input records are to be transferred to the output file or device and in what order, and you define literal text (if any) to be inserted between the fields of "translated" data from tape. The order in which you make these specifications is the order in which the input fields and any literal insertions will appear in the output. Input fields that are not defined in this phase will simply not become part of the output. You don't specify the output record length, FXED will add up all the output field definitions and tell you what it is. Comments are an optional part of each output field definition. Besides their value in documenting the output field definitions, the comments serve a purpose in the optional, automatic generation of an empty dBASE .DBF structure.

"Record Selection": You may optionally define a logical expression which will cause FX to conditionally accept or reject each input record after (a) comparing the contents of specified fields within the record with literal text or with each other or (b) looking for the field in a "KEY LIST" disk file. You specify that one or more fields, after translation, must meet your criteria or the entire input record is skipped. The comparison operators are = (equal), <> (not equal), >= (greater than or equal), <= (less than or equal), > (greater than), < (less than), [ ] (contains),] [(does not contain), $ (can be found in) and -$ (can not be found in). The [ ] and] [operators allow you to ask whether a literal text string can (or can not) be found ANYWHERE within a field (range of bytes) in each input record – a great feature to compensate for sloppy keypunching. Multiple comparisons may be joined by the logical conjunctions

"AND" and "OR". Records that do not meet the acceptance criteria are ignored and thus do not become part of the output; (i.e. records not accepted are rejected).

"Logical End-Of-File": You may optionally define a logical expression that will cause processing of the input file to stop (simulate hitting the 'end-of-file') if the specified condition becomes true. The syntax of the logical expression is the same as in paragraph 2 above.

"Secondary Translation:" You may optionally define secondary translation to be performed on data read from tape after it has been processed by the primary translation phase. This feature allows you to define an entire character set translation if you like. A more common use is to re-define one or two characters that would otherwise offend the application program that will be processing the imported data.

GENERAL:

FIELDS: As you define the processing which FX is to perform, you'll be entering numbers that specify "fields". For example, in the translation definition phase a field is a range of bytes all of which are to receive the same type of translation. Whenever you get a tape from one of your sources it should be accompanied by paperwork that specifies

the contents of the various fields which make up each logical record. Some sources specify that a field begins on a particular byte and ends on another byte. Another way you'll see fields specified is by starting byte number and length. For your convenience, FXED allows you to specify fields as either a range of byte numbers (e.g. 10-20), as a starting byte and a length (e.g. 23L6) or as a length starting at the next available byte (e.g. L8). Use whichever method you prefer, or mix all three. Internally, FXED maintains field definitions as ranges of bytes; this is the format you'll see if you ask for a display of currently defined fields.

THE BLOCKING FACTOR on tape (ratio of physical tape block length to logical record length) must be an integer, (i.e. logical records must not span tape blocks). If they do, you can transfer the tape data to disk and write it back out with a proper blocking factor or process the disk file directly with FXED's "Disk-to-Disk" mode without restriction.

PHYSICAL TAPE BLOCK LENGTH ... OUTPUT LENGTH When running FX with a control file, the maximum length of a block of data from tape can normally be up to 65,535 bytes (64K-1). There's a smaller limit in cases where you're creating output records that are larger than the input records (e.g. due to text insertion, packed decimal expansion etc.). When all of the logical records in an input block are expanded for output, the result must not exceed 64K.

INPUT EDITING: Use <BACKSPACE> to edit the current line in any FXED definition phase. Completed lines may be easily replaced by new lines.

TO RUN FXED:

Simply type: "FXED <ENTER>"

Or: "FXED filename <ENTER>"

Where "filename" is the name of an existing or new FXED control file. If the filename you specify does not contain a period, a filename extension of .CF is assumed.

If you don't enter a filename on the command line, FXED will ask for one.

If the control file already exists, you'll see a summary of statistics about it, e.g.:

```
Editing an existing FXED Control file: DEMO.CF
```

```
Which includes the following:
Input record length  = 111
Output record length = 63
Conditional processing criteria exist.
Secondary Translation Table exists.
Press <ESC> for Main Menu:
Here you hit <ESC> and see the main menu (less page references and notes):
Editing FXED control file: DEMO.CF
Main Menu
 T - Define Types of Translation.                      REQUIRED
 O - Define Output Record Layout.                      REQUIRED
 C - Define Conditional Record Processing.             OPTIONAL
 E - Define Logical end of file                        OPTIONAL
 S - Define Secondary Translation Table.               OPTIONAL
 V - View all definitions.                             OPTIONAL
 P - Print all definitions.                            OPTIONAL
 D - Disk File Processing                         <- read this, it's a powerful
feature
 G - Generate .DBF File                                OPTIONAL
 L - Change Input Record Length.                       OPTIONAL
 F - File Maintenance.                                 OPTIONAL
 Q - Quit, Saving results on disk.                     ONE OF THESE IS
 A - Abort, Don't save results on disk.                REQUIRED TO EXIT FXED
ESC - Return to this menu from any phase.

REMEMBER: You should select "T" first because you won't be allowed to operate on
undefined fields by "O", "C" or "E".
"O" is required eventually or you won't have any output data when you run with this
control file.
```

GENERAL INFORMATION - ENTERING RANGES OF BYTES

During the Translation Definition phase and the Output Definition phase much of your input will be ranges of bytes (field definitions), specifically ranges of bytes within the input record structure. FXED has several powerful features to facilitate the entry of typically required field definitions.

In general, throughout FXED, byte ranges may be entered in 2 ways:

<STARTING BYTE> - <ENDING BYTE>   e.g.: 11-22

This specifies bytes 11 thru 22 (12 bytes) from the input record.

<STARTING BYTE> L <LENGTH>   e.g.: 11L12

This specifies a field starting at byte 11 with a length of 12. It has the same effect as specifying "11-22".

Both of the above methods are valid throughout FXED whenever you need to specify a range of bytes.

There is a 3rd method, relative definitions, valid only in the initial Translation and Output definition phases:

<IMPLIED STARTING BYTE> L <LENGTH>   e.g.: L6

This specifies a field of length 6, starting at the next available byte from the input record. Where it starts depends on the most recent previous field definition. You'll find this to be the easiest way to enter fields when you're taking them in order from documentation that shows field lengths.

Note that this form of field definition is extremely powerful when it is used within a "MACRO" (explained next).

Macros

NOTE: Macros are wonderful work-savers in the right circumstances. They are most useful in situations where the user is creating lengthy control files which require numerous, repetitive sequences of keystrokes. First-time users in a hurry to get a small job done can skip to the next page if they promise to come back and learn about macros later.

What is a macro ? A macro is simply a string of characters (keystrokes) that is saved internally by FXED associated with a particular function key. After a macro has been defined, its contents will be played back, just as if you had manually typed them, whenever the associated function key is hit. Use macros if you need to make the same keyboard entries repeatedly and would like to save on keystrokes and errors.

FXED allows a maximum of 8 macro definitions. The keys <F1> thru <F4> are the macro keys. Four macros may be defined in the Translation Definition phase and a different four macros may be defined in the Output Definition phase. Note that this means that any particular MACRO key, e.g. <F1>, will generally have two distinct meanings, depending on what phase you're in when you hit it. The 8 macros are saved within the control file which FXED creates, thus when re-editing a control file they need not be re-defined.

To define a macro (while in the Translation Definition Phase or Output Definition phase) simply hit one of the keys <F1>, <F2>, <F3> or <F4> which has not previously been defined for that phase. Type in any series of keystrokes you want associated with that macro, then hit the same function key again. Macro definitions may not be nested, i.e. you can not include the function key associated with one macro within another.

To use a macro (while in the Translation Definition Phase or Output Definition phase) simply hit one of the keys <F1>, <F2>, <F3> or <F4> which has previously been defined. All of the keystrokes contained in the macro will be "played back", just as if you had manually typed them in.

To delete a macro (while in the Translation Definition Phase or Output Definition phase) hit the <DEL> key. FXED will ask you which macro you want to delete. Then it may be re-defined if you like.

To display your macros (while in the Translation Definition Phase or Output Definition phase) use the "M" command. The contents of all 4 macros associated with that particular phase will be displayed on the screen.

THE TRANSLATION DEFINITION PHASE (Select "T" from the main menu)

THIS PHASE MUST BE USED. You must define a translation type for each input byte which: (a) will become part of the output, or (b) will be evaluated in a logical expression (i.e. conditional acceptance or logical End of File).

Even if the input data is an ASCII file: If the data is already in ASCII and you don't want to change (translate) it in any way, simply define the entire record length for no translation with the "=" command.

FORMAL PRESENTATION OF TRANSLATION COMMANDS:

```
<range>E       Translate the bytes in <range> from EBCDIC to ASCII.
<range>e       EBCDIC to ASCII but force lower case ASCII output.
<range>CTRL-E  EBCDIC to ASCII but force UPPER case ASCII output.
<range>ALT-e   Same as "e" but not forcing LOWER case at beginning of words.
<range>P       Translate the bytes in <range> from Packed decimal to ASCII.
<range>U       Undefines the bytes in <range>. (UNDEFINE PREVIOUS DEFINITIONS TO
SAVE PROCESSING TIME)
<range>Z       Translate the bytes in <range> from Zoned decimal to ASCII.
```

```
<range>=        Defines the bytes in <range> as needing no translation.
<range>A        Translate the bytes in <range> from ASCII to EBCDIC.
<range>a        No translation but force lower case ASCII output.
<range>CTRL-A   No translation but force UPPER case ASCII output.
<range>ALT-a    Same as "a" but not forcing LOWER case at beginning of words.
<range>H        Translate each byte in <range> from packed HEX to 2 ASCII chars
<F1>...<F4>     Define/Invoke <F1>-<F4> macro.
       M        Show macros <F1>-<F4>.
   <DEL>        Delete a macro.  FXED will ask which one.
       V        View the translation definitions.
       ?        Displays this list of commands.
   <ESC>        Return to Main Menu.
```

THE TRANSLATION DEFINITION COMMANDS BY EXAMPLE:

Examples of the translation definition commands with explanations follow. Although there some logical flow as one example follows another, the collection of examples as a whole does NOT represent a typical translation-definition session, it contains a mixture of commands useful with "EBCDIC" input data (e.g. E and Z) and others which pertain to ASCII and BINARY data (e.g. =).

WHAT YOU DON'T HAVE TO DO: In the translation definition phase, you don't have to make entries in any particular order. You don't have to specify a translation type for every byte in your input record - only for those bytes which will either become part of the output or which will be evaluated in a conditional expression. You may define a translation type for bytes you won't be using, but by skipping them you will save a small amount of processing time when the job is run. Another thing you don't have to do in this phase is hit the <ENTER> key. The character that defines the type of translation also terminates the line.

3-10= Process bytes 3 thru 10 of each input record without change.

3L8= Starting with byte 3, process 8 bytes of each input record without modification. This is the same as entering "3-10=".

45-50E Translate bytes 45 thru 50 from EBCDIC to ASCII. UPPER case EBCDIC to UPPER case ASCII, lower case EBCDIC to lower case ASCII.

45E Translate byte 45 from EBCDIC to ASCII. UPPER case EBCDIC to UPPER case ASCII, lower case EBCDIC to lower case ASCII.

45^E Translate byte 45 from EBCDIC to ASCII. Force UPPER case ASCII.

46-50e Translate bytes 46 thru 50 from EBCDIC to ASCII.

Force lower case ASCII output.

45^A Don't change byte 45 at all unless it is a valid ASCII lower-case alphabetic character, then change it to the equivalent upper case ASCII character.

45a Don't change byte 45 at all unless it is a valid ASCII UPPER-case alphabetic character, then change it to the equivalent lower-case ASCII character.

1-108A Translate bytes 1-108 from ASCII to EBCDIC. Translate ASCII to EBCDIC when transferring data to the PC ?? Why would you want to do that ?? The customer who requested this feature is using his PC as a front-end to a mainframe. He gets ASCII tapes from minicomputers and his goal is to move the data thru the PC to an IBM (EBCDIC happy) mainframe.

NOTES ABOUT FORCING UPPER CASE & lower case:

Don't worry about non-alphabetic characters when using one of the translation types which force the output to either UPPER or lower case. Only alpha characters (A-Z and a-z) are considered for case changes.

The EBCDIC character set allows lower case numbers (ASCII does not). Lower and UPPER case EBCDIC numbers are always translated to normal ASCII numbers

ALT-a and ALT-e TRANSLATION TYPES

In both cases, the resulting output text is forced to lower case ASCII except for characters at the "beginning of words". The beginning of a word is defined to mean any of the following:

```
The first character of an OUTPUT FIELD or a character preceded by any of the following:
          SPACE          - HYPHEN            / FORWARD SLASH       " DOUBLE QUOTE
        _ UNDERLINE      . PERIOD            ( LEFT PARENTHESIS

98-108P           Process bytes 98 thru 108 of each input record as Packed Decimal. Byte
                  108 specifies the least significant digit and the sign of the whole
                  packed decimal field.  The sign (SPACE or -) will be copied to the
                  output just before the digits derived from byte 98. In this example, 11
98L11P            bytes of input data will result in 22 bytes of output data.

51-55Z            Process bytes 51 thru 55 of the input records as Zoned Decimal.  Byte
                  55 specifies the least significant digit and the sign of the whole
                  zoned  decimal field.  The sign (SPACE or -) will be copied to the
                  output just before the digit derived from byte 51.  Thus the 5 bytes
51L5Z             of input data will create 6 bytes of output data.
```

NOTES ABOUT "P" and "Z":

INVALID SIGN: Only certain codes (see the final page of this chapter) are valid in specifying the sign in the least significant byte of Packed Decimal and Zoned Decimal fields. If an invalid code is found for the sign, the character "?" will be placed in the sign position of the output.

ADJACENT PACKED DECIMAL or ZONED DECIMAL FIELDS: The software needs to know where each Packed Decimal or Zoned Decimal field ends so it can look for, and extract, the sign information. If you have 2 or more adjacent Zoned or Packed fields you have to define each field separately in the Translation phase.

1-7H Translate bytes 1 thru 7 from Packed HEX/BCD to ASCII. Each 8-bit byte will be translated to 2 ASCII characters, 0-9 or A-F.

V Enter "V" at any time to View what's been defined so far. You see a sorted list of the fields you've defined and their translation type. Mixed in you'll be shown all fields which are currently undefined.

? Enter a question mark if you'd like to see a list of the valid FXED commands for this phase.

<ESC> Hit <ESC> to return to the main menu at any time. You may re-enter the translation definition phase from the main menu as often as you like.

MACROS IN THE TRANSLATION DEFINITION PHASE

Frequently the translation definition phase is trivial to fill out. This is the case with input records that are all EBCDIC or all ASCII. We'll start with an example of this. Macros are of no help in the first example.

For the duration of these examples we'll assume an input record length of 600 bytes.

If all 600 bytes are EBCDIC, the following three keystrokes take care of the translation definition phase, starting at the main menu:

T ; to get into the translation definition phase

1-600E ; to say that all 600 bytes get EBCDIC to ASCII translation

<ESC> ; to go back to the main menu

If the input records were all ASCII, line 3 would be: "1-600="

Sometimes tapes will contain records where the type of translation needed alternates, in a regular pattern, between EBCDIC-to-ASCII and PACKED DECIMAL-to-ASCII. You might see this, for example, in a price list tape where the fields are:

01-20 ; description of item in EBCDIC

21-25 ; vendor's part number in PACKED DECIMAL

26-30 ; our internal part number in PACKED DECIMAL

31-50 ; same as 01-20

51-55 ; same as 21-25

56-60 ; same as 26-30

etc. etc. (the pattern repeats over and over)

The translation definitions to handle this would be entered as follows -

(the SPACEs are not to be entered, they are shown for clarity only):

1-20E L5P L5P

L20E L5P L5P

L20E L5P L5P

L20E L5P L5P

L20E L5P L5P etc. etc.

Note the repeating pattern after the first line.

Here's where a macro would be a big help. You define a macro by hitting the keystrokes shown below -

(the SPACES are not part of the macro, they are inserted in the example for clarity):

<F1> L20E L5P L5P <F1>

The macro consists of all the characters you type in between the 2 <F1>'s. Once a macro is defined, hitting the associated key (<F1> in this example) "plays it back" - just as if you had entered all the keystrokes it represents. This will greatly simplify entering the rest of the translation definitions. Note that if you had repeated the pattern two times within the MACRO it would cut your work in half again.

THE OUTPUT DEFINITION PHASE (Select "O" from the main menu)

THIS PHASE MUST BE USED.

Here's where you specify which fields from the input records will go to the output file or device and in what order.

Fields that aren't specified aren't written to the output - that's how you delete fields.

This is also where you get to insert literal text. Literal text is constant (unchanging) text you want to add to each and every output record. It doesn't come from the input records you specify it here. You use the INSERT key (shown as <INS>) to define the start and end of a literal text string.

The order in which you specify the output fields is important; it's the order in which they'll be written. That's how you move fields around to positions in the output that are different from their positions in the input.

You don't specify the output record length, FXED will add up all the output field definitions and tell you what it is.

If you change your mind or make a mistake in the order, you can make deletions, insertions or replacements at any time.

FORMAL PRESENTATION OF OUTPUT PHASE COMMANDS:

```
<range><ENTER>      Add the input bytes specified by <range> to the output.
<range>\            Same as <ENTER> except that trailing SPACEs are deleted.
<range>X            Same as \ but also removes leading SPACEs and 0's.
       S            Adds a SPACE to the output.
<INS>text<INS>      Adds the text specified to output at current position.
<INS><INS>text<INS> Adds the text specified to output at current position+1.
      xN            Add x digits of the Input Record Number to the output.
  x<TAB>            Tab to column x of the output.
      xU            Undefine (delete) output definition line x.
      xI            Start Inserting output definition lines after line x.
      xR            Start Replacing output definition lines at line x.
      xC            Start editing Comments at line x.
<F1>...<F4>         Define/Invoke F1-F4 macro.
       M            Show macros F1-F4.
   <DEL>            Undefine macro.
  (a-b)V            View lines a-b of the output definitions.
       ?            Displays this list of commands.
   <ESC>            Return to Main Menu.
```

THE OUTPUT COMMANDS, BY EXAMPLE: Each pair of numbers below, in parentheses is created by FXED. The first number is a "line number" (for editing purposes), the second is the "next available character position" in the output record you're defining. You'll reference the line numbers whenever you want to edit the control file to make deletions, insertions, replacements etc. Remember, the numbers you input are with respect to the layout of the INPUT RECORDS. FXED will show you the resulting range of bytes and field length for the OUTPUT records. They will often be different.

(01:001) 70-80 <ENTER>

Bytes 70-80 of each input record, after any translation, will occupy the first 11 bytes of the output.

(02:012) 70-80\

Same as above except that trailing SPACE characters will be deleted from the output. Note that the "\" and "X" commands will cause variable length output records. HINT: Using \ instead of <ENTER> to terminate output field definitions can save a lot of space on your disk in certain situations, depending on the data you're importing. If you choose to use \, remember to import the data into a database program as DELIMITED data, not SDF.

(03:023) <INS> - <INS>

The next 3 bytes of the output (12 thru 14) will be "SPACE-SPACE".

(04:026) 45-55 <ENTER>

Bytes 45-55 of each input record, after any translation, will occupy the next position of the output (bytes 15 thru 26).

(05:037) S

"S" is a shorthand way of inserting a single SPACE into the output. Same as entering <INS> <INS>.

(06:038) V

As in the translation definition phase, entering "V" allows you to View the contents of the output definition list at any time.

"V" is especially useful as the first step if you're re-editing an existing control file.

Here's what "V" would display for the entries on the previous page:

```
  Line    Range of   Range of  Output    Type of
  Ref     Input      Output    Field     Output
  #       Bytes      Bytes     Length    Field           Optional Comment
 01: I = 070-080  O = 001-011  L = 011  FIELD         ; Output bytes 70 thru 80
 02: I = 070-080  O = 012-022  L = 011  TRUN FLD      ; Now delete trailing SPACEs
 03:              O = 023-025  L = 003  " - "         ; Insert SPACE - SPACE
 04: I = 045-055  O = 026-036  L = 011  FIELD         ; Next, bytes 45-55 go out.
 05:              O = 037-037  L = 001  " "           ; Insert a single SPACE
Output Record Length = 37 bytes or less (1 or more variable length fields
present).
(06:038) 5N
```

At run-time, the Input Record Number (starting with 0) will be added to the output, as a 5-digit number, left-filled with 0's. If the input record number is wider than 5 digits, only the 5 least significant digits will be added to the output. The "N" feature is useful when, by multiple passes, using different conditional acceptance criteria, the input file is separated into a number of different disk files. It allows unequivocal determination of the original order of the input records.

(07:043) 50<TAB>

"TAB" to output column 50. In effect, at run-time, before input data and literal data are moved to the output, the output buffer is pre-filled with ASCII SPACEs. The <TAB> command lets you select any arbitrary position as the position where the NEXT output character will go. Tabbing in any direction (backwards or forwards) is allowed. Positions that are never overwritten stay as SPACEs. The <TAB> key on some PC's does not say 'TAB'; it has arrows pointing left and right.

(08:050) 5U

"Undefine" line 5. Deletes line 5 of the output definitions. What was line 6 now becomes line 5, line 7 becomes line 6 etc. If you want to "U" several lines, work from the highest numbered line to the lowest numbered line - because each "U" causes the following lines to "move up" in the list, changing their line numbers as things progress.

(07:050) 4I

Start inserting lines, after line 4. Enter a line. It will become line 5. Hit <ESC> to exit from inserting.

I(05:037) 1-5 <ENTER>

We enter the new line 5. The "I" reminds us we're inserting.

I(06:042) 3R

Start replacing lines at line 3. Equivalent to the 2-steps: "3U" followed by "2I". Hit <ESC> to stop replacing.

R (03:023) 45-56 <ENTER>

Here's the new line 3. We decided to change the range.

?

You enter a question mark if you'd like to see a list of the valid FXED commands for this phase.

<ESC>

Hit <ESC> to return to the main menu at any time. You may re-enter the output definition phase from the main menu as often as you like to continue and/or modify your definitions.

NOTES ABOUT LITERAL TEXT INSERTION:

(1) To insert <CR><LF> USE THE <ENTER> key, e.g. <INS><ENTER><INS>.

(2) To insert only CR, use the backspace key (left pointing arrow) to delete the LF i.e.

<INS><ENTER><BS><INS>.

(3) LITERAL TEXT can be any of the ASCII characters (000-127) and most of the non-ASCII codes above the ASCII character set (129-255). Only decimal 128 (hex 80) is restricted. Call Electrovalue if restricting this character causes you a problem.

(4) To insert NUL (decimal 000) use the key combination "CTRL-SHIFT-@"; DOS won't let you create a NUL using ALT-0 from the numeric keypad.

COMMENTS IN THE OUTPUT DEFINITION PHASE

The ability to comment each line in the output definition phase is a big help in coordinating the control file both with the original input record's field definitions and with your desired output record format. The comments become a permanent part of the control file and are shown every time the definitions are displayed or printed. Comments may be up to 40 characters in length. Only the first 24 characters fit a V screen display without wraparound. For convenience, comment lines may be terminated with several of the keys that can terminate a field entry: (<ENTER>, "\" and <TAB>).

Comments have an important second use. The first word is used as an xBASE 'field name' whenever you use FXED's "G" function to automatically generate an empty .DBF structure.

Editing Comments: Comment fields can be edited by using the "nC" command where "n" is the line number at which you want to start adding or editing comments. Comment edit mode is useful in adding comments to old FXED control files and in fine tuning comments used in automatic .DBF structure generation. When editing comments, the <BS>, left and right arrow and <DEL> keys have their normal DOS functions. <INS> toggles between the INSERT and OVERWRITE modes, CTRL-Y clears from the cursor to the end of the line, <ESC> leaves comment edit mode without any changes to the current comment. Hit <ENTER> to make permanent changes to the current line and move the editing cursor to the beginning of the next field-definition's comment.

MACROS IN THE OUTPUT DEFINITION PHASE

Suppose you're going to transfer a mailing list from tape to a number of floppy diskettes. You'll send the floppies to your customer who will import the data to his database.

Typical mailing list tapes contain from 40% to 60% SPACE characters in the form of trailing blanks that follow the actual data in each field. FXED allows you to output fields from the input record less any trailing blanks. You use the "\" terminator instead of <ENTER> to end the field definition. Using this method on each field from the mailing list tape will cut down the number of floppies you'll need to approximately half.

If you do choose to delete trailing blanks, you'll have to delimit the data so you don't loose the field definitions. The most common delimited data consists of character strings enclosed in 'double-quote' characters. Commas separate individual fields. Each logical record ends in <CR><LF>. Here's a sample:

"ABC","DEFGHI","JKLMNOPQ","RSTUV"<CR><LF>

When you choose to delete trailing blanks from each input field and delimit the output fields, you'll find yourself hitting the following keystrokes at the end of each field definition except the first and last:

\ \ <INS> " , " <INS> <ENTER>

NOTE: None of the spaces in these examples (above and below) belong there; they were inserted for clarity. Note that the double \\ and the <INS><ENTER> imply that you're not going to enter any comments in the optional comment area.

The example above, including shifting up and down for the double-quote characters, contains 12 keystrokes. Define it as a macro and you cut that down to ONE keystroke. The definition is:

<F1> \ \ <INS> " , " <INS> <ENTER> <F1>

Here's a sample listing of a typical output definition, the creation of which would be facilitated by the above macro.

```
Line    Range of         Range of       Output       Type of
Ref     Input            Output         Field        Output
#       Bytes            Bytes          Length       Field              Optional Comment

 01:              O = 001-001  L = 001  '"'
 02: I = 001-020  O = 002-021  L = 020  TRUN FLD
 03:              O = 022-024  L = 003  '","'
 04: I = 021-040  O = 025-044  L = 020  TRUN FLD
 05:              O = 045-047  L = 003  '","'
 06: I = 041-060  O = 048-067  L = 020  TRUN FLD
 07:              O = 068-070  L = 003  '","'
 08: I = 061-080  O = 071-090  L = 020  TRUN FLD
 09:              O = 091-093  L = 003  '","'
 10: I = 081-100  O = 094-113  L = 020  TRUN FLD
 11:              O = 114-116  L = 003  '"<CR><LF>'
```

NOTE: Macros can not start within a comment field nor after the <INS> which signals a literal text insertion. To use the macro feature in a comment, simply start the macro definition with the character that terminates the previous field definition.

Output definition phase-forward insertions

If you want to insert literal text between 2 fields from tape, you normally create a literal text insertion field, e.g.:

<INS>this is the insertion text<INS>

Numeric fields on so-called "EBCDIC" tapes are often encoded as 'Packed Decimal'. Packed Decimal data is also found (infrequently) on so-called pure ASCII tapes.

Packed Decimal data is easily converted to an ASCII character string by the Electrovalue software, all you have to do is identify it as such (P) to FXED in the translation phase.

When Packed Decimal input data is translated to ASCII, you get two ASCII characters (two bytes) from each single byte on tape. See the last page of this chapter for a definition of Packed Decimal data, with examples.

It's fairly common to want to unpack a number and insert a decimal point (for dollars and cents) or slashes (in a date field) as the data is being transferred from tape to disk. When the tape data is Packed Decimal, you're likely to find that you need to make an insertion between two bytes that are derived from the same byte on tape. In this case you need to use "Forward Insertion".

Forward Insertion inserts the specified literal text string after the first character of the NEXT field. A Forward Insertion field definition starts with <INS><INS> and is terminated with a single <INS>, e.g.:

<INS><INS>.<INS>

When displayed with V or P (see 3/ below - line 3), forward insertions will show an adjusted output range and "->".

HERE'S AN EXAMPLE:

According to the paperwork that came with the tape, the data on tape consists of three fields:

```
BYTES    DESCRIPTION     NOTES
 1-5     Part Number     EBCDIC
 6-10    Cost of Part    Packed Decimal, Dollars and cents.
11-15    Vendor Code     EBCDIC
```

Analysis shows that a part costing $123,456.78 would be encoded on tape as:

```
BYTE            06   07   08   09   10
CONTENTS        01   23   45   67   8F
```

In this example, we want to insert a decimal point between the 6 and 7, i.e. between two bytes which will be derived from the single tape byte in position 09.

THE FOLLOWING SCREEN DUMPS SHOW:

1. The FXED Translation definitions,

```
       1-5E
      6-10P
     11-15E
```

2. The FXED Output Definition phase dialogue,

```
 (01:001) 1-5 <ENTER>         ; PART NUMBER <ENTER>
 (02:006) 6-8 <ENTER>         ; COST, MOST SIGNIFICANT <ENTER>
 (03:013) <INS><INS>.<INS>    ; INSERT THE DECIMAL POINT <ENTER>
 (04:015) 9-10 <ENTER>        ; COST, LEAST SIGNIFICANT <ENTER>
 (05:017) 11-15 <ENTER>       ; VENDOR CODE <ENTER>
```

3/ The FXED output definitions, as displayed by V or P,

```
01: I = 001-005  O = 001-005  L = 005  FIELD         ; PART NUMBER
02: I = 006-008  O = 006-012  L = 007  FIELD         ; COST, MOST SIGNIFICANT
03:              O = 014-014  L = 001  "."->         ; INSERT THE DECIMAL POINT
04: I = 009-010  O = 013-016  L = 004  FIELD         ; COST, LEAST SIGNIFICANT
05: I = 011-015  O = 017-021  L = 005  FIELD         ; VENDOR CODE
```

Output definition phase – discussion & examples

A lot of the power of an FXED control file stems from the features resident in the Output Definition Phase. A clear understanding of these features will let you use your tape drive to maximum advantage.

Besides the "commands" which control the FXED editor itself, the Output Phase "commands" can be grouped into 2 categories:

1) Commands which select fields from the input stream for addition to the output.

<range><ENTER>   e.g. 25-35<ENTER>   25L11<ENTER>   L9<ENTER>

The examples above will cause the run-time program to take the specified range of bytes from each input record, perform any translations specified for that range in the Translation Definition Phase and, without any further processing, add the resulting character string (starting at the next available spot) to the output record. If the bytes are derived from either a Packed Decimal, Zoned Decimal or Packed HEX input field, the output string will be longer than the input string. Otherwise, the input and output strings will be the same length.

<range>\   e.g. 25-35\   25L11\   L9\

Think of the backslash as pushing away the trailing (right-hand) SPACE's. The examples above will cause the run-time program to take the specified range of bytes from each input record, perform any translations specified in the Translation Definition Phase, then strip all trailing SPACE's from the character string and add the resulting character string (starting at the next available spot) to the output record. The resulting string can be as small as 0 characters.

<range>X   e.g. 25-35X   25L11X   L9X

Think of the "X" as standing for eXtract. The examples above will cause the run-time program to take the specified range of bytes from each input record, perform any translations specified in the Translation Definition Phase, then strip all leading (left-hand) ZERO's and SPACE's and all trailing SPACE's from the character string and add the resulting character string (starting at the next available spot) to the output record. The resulting string can be as small as 0 characters.

For example:

' 00012345   '   becomes   '12345'   The quotes aren't part of the string - they're to show SPACE's.

2) Commands which add literal (constant) text to the output.

<INS>any text string<INS>   S   <INS><INS>text string<INS>   n<TAB>

Four different commands are shown on the above line.

The <INS> command, by far the most common, lets you insert any string of characters into the output record, starting at the next available spot. Use it to add field delimiters, to add <CR><LF> at the end of each record, to insert "/" characters in date fields, to insert "-" characters in telephone numbers etc.

The S command is simply a shorthand way of inserting a single SPACE character.

The <INS><INS>text string<INS> Forward Insertion command, is explained with examples on the previous page.

The "n<TAB>" command inserts multiple SPACE characters up to, but not including, output record position "n". The <TAB> feature is of particular use following one or more "\" or "X" commands which have been used to extract and combine the significant characters of several fields. Examples are in creating mailing labels, and in creating lookup fields in real estate data from lot and block numbers.

The conditional processing definition phase

This phase in the creation of an FXED control file is optional.

Conditional record processing provides the means by which each input record may be accepted or rejected based on the contents of one or more of its fields. Records that meet all of the criteria are accepted; the others are rejected.

Comparison operators are used in comparing the contents of a field with (a) literal text or (b) the contents a second field. Additionally, the operators $ and -$ can look for a field in a disk file called a "KEY LIST".

The comparison operators are:

```
=       equal              <>      not equal
<       less than          <=      less than or equal
>       greater than       >=      greater than or equal
[ ]       contains          ] [       does NOT contain
 $  CAN be found in "KEY LIST" disk file
-$  CAN NOT be found in "KEY LIST" disk file
```

Comparisons may be combined using the logical conjunctions "AND" and "OR".

When you specify a range of bytes it refers to byte positions in the input record. Remember this when you're comparing to zoned decimal or any of the packed numeric formats.

Comparisons are made after any translation(s) have been performed.

Literal text strings must be bracketed with either single or double quote characters. The same type of quote must be used at both ends of any particular literal text string. Literal text is NOT allowed on the left side of a comparison.

The expression may come from the keyboard (maximum length 256) or from a disk file (2048 characters maximum).

Let's assume we've purchased a tape that lists all of the computer and electronics stores in the USA. We want to do a mailing to RADIO SHACK stores in Wyoming and Utah. The pertinent field definitions (from the mailing list supplier) are:

9-32 NAME OF STORE, 71-72 STATE ABBREVIATION, 81-85 ZIP CODE

To define the acceptance criteria, we select "D" from the sub-menu. We are invited to enter a logical expression by the phrase "Accept if ". We input the line below. Be careful that the comparison text is entered in the right case, we're assuming that this tape is all in UPPER CASE or that we've forced UPPER CASE output in the Translation Phase.

Accept if 9L11 = 'RADIO SHACK' AND 71-72 = "WY" OR 71-72 = 'UT' <ENTER>

Parentheses may be used to specify or clarify logical groupings, the above could be entered as:

Accept if   9L11 = "RADIO SHACK" and (71-72 = 'WY' OR 71-72 = "UT") <ENTER>

The two entries above are equivalent. Use parentheses if you want to be sure how FXED will evaluate your conditional expression. Whether or not you have used parentheses during input of the expression, when you use the "V" command to view your conditional expression, FXED will display it with parentheses to show how it's been evaluated.

Another example. We want to do a mailing to all stores in ZIP codes 07900 thru 07999, except RADIO SHACK stores. The expression we enter is:

Accept if   9L11 <> 'RADIO SHACK' AND 81-85 >= '07900' AND 81-85 <= '07999'

MORE EXAMPLES, WHICH ILLUSTRATE THE [ ] and] [operators.

We want to send catalogs to all beachfront and coastal stores. The expression we enter is:

Accept if   9-32 [] 'BEACHFRONT' or 9-32 [] 'COASTAL'

We want to do a mailing to all stores in ZIP codes 12345 thru 23456, except stores with SHACK as any part of their names. The expression we enter is:

Accept if   9-32]['SHACK' and 81-85 >= '12345' and 81-85 <= '23456'

"KEY LIST" disk files in CONDITIONAL ACCEPTANCE expressions. The $ and -$ operators.

The $ and -$ operators allow the creation of simple Conditional Acceptance Expressions which will import only those tape records which contain a field (any range of bytes) which can be found ($ operator) or not found (-$ operator) within a disk file. Each disk file, called a "KEY LIST" file, may contain up to 64K-1 (65,535) keys. If you need more, multiple KEY LIST files may be specified using the logical AND & OR operators. It is suggested that you name KEY LIST files with a .KL extension but this isn't enforced by the software.

Think of the $ operator as meaning CAN BE FOUND IN and the -$ operator as meaning CAN NOT BE FOUND IN. The rest of this section will present these features in terms of the $ operator; please remember that -$ is also available and it works just the opposite.

In some ways, the FXED $ operator is similar in function to the dBASE $ operator. The important difference is that the Electrovalue software doesn't look to see if it can find <STRING1> within a single example of <STRING2>, it looks at each and every (fixed-length) string in the KEY LIST file until it finds one which matches <STRING1>. If the string from tape is found in the KEY LIST file, the expression is TRUE, otherwise it is FALSE.

The "keys" in the KEY LIST file consist of fixed-length strings of ASCII characters terminated in any 2 characters. The terminating characters (usually <CR><LF> but any 2 characters may be used) are not considered to be part of the string being matched. The length of a key is from 1 to 510 characters plus the 2 terminating characters.

IMPORTANT: THE KEY LIST FILE MUST BE SORTED IN ASCENDING ORDER. You can use your DBMS program to create and sort the key list file.

AN EXAMPLE: Suppose you're in the business of selling government surplus airplane parts. From time to time, you receive a tape listing hundreds of thousands of parts that are going to be auctioned off at various sites around the USA.

Each record contains fields for QUANTITY, PART NUMBER, CITY OF AUCTION, STATE OF AUCTION, CONDITION and DATE OF AUCTION. The PART NUMBER field is bytes 11-22 of each record and the STATE is bytes 40-41.

Your firm is only interested in 5,000 particular helicopter parts and you can only participate in auctions in southern states. You'd like to make one pass on the tape and transfer to disk only those records which meet these two criteria.

You create (and maintain) a KEY LIST containing only the part numbers which interest you. Terminate each part number with <CR><LF>. You create another KEY LIST file with a list of states you can attend auctions in.

The file PART-NO.KL will have entries like this:

```
1134-2222-99
1134-3434-65
1244-4566-32  etc.
```

The file STATES.KL will have entries like this:

AL
AZ

```
GA
SC
TX     etc.
```

Your Conditional Acceptance Criteria will look like this:

Accept If   11-22 $ 'PART-NO.KL' and 40-41 $ 'STATES.KL'

NOTES:

The KEY LIST filename must be surrounded by quotes, single or double, just like any other string used in FXED logical expressions. A KEY LIST filename may contain a disk designator and pathname.

Processing involving large KEY LIST files will proceed much faster if you use a RAM DISK. If the KEY LIST is larger than 8K bytes, a RAM DISK is strongly suggested.

The $ and -$ operators may be used in "Logical End-Of-File" expressions as well.

Control files using this feature may be used with FX (Tape-to-Disk), FXED (Disk-to-Disk) and TAPE COPY (Tape-to-Tape).

Taking the conditional expression from a disk file

When you select "D" from the sub-menu to define a logical expression, FXED prompts "Accept if " (or "Stop if ", see below) then pauses for input. As previously shown, you can proceed by typing in a logical expression.

Alternatively, you can input the character "@" followed by the name of a disk file, e.g.:

@FILENAME <ENTER>

in which case the logical expression will be taken from the disk file FILENAME.

FXED will import the contents of FILENAME, add parenthesis, and display the result.

The maximum logical expression length, when taken from a disk file, is 2048 characters, not counting <CR><LF> characters. The disk file may contain as many <CR><LF> characters as you like in the interests of clarity, or none at all. FXED strips them away when importing the contents of the disk file, e.g.

```
1-5 = "ABCDE" or

11-15 >= '70000' and

99-100 = "NJ"
```

THE "LOGICAL END-OF-FILE" DEFINITION PHASE   (Select "E" from the main menu)

This phase in the creation of an FXED control file is optional.

The Logical End-Of-File (EOF) feature allows processing to stop, based on encountering data that meets a condition you have defined, before the entire input file has been read.

The instant the Logical End of File condition becomes TRUE, processing of the input file stops, just as if the record that matched the condition were actually an End of File. The data from the record that satisfies the End of File criteria does not become part of the output file.

This can be a tremendous time saver in situations where you know something about the order of the data on the tape (e.g. it's in ZIP code order).

The rules concerning creation of the Logical End of file conditional statement are the same as those for Conditional Record Processing (fully described in the previous section).

When you select "D" from the Logical End of File (EOF) sub-menu, you are prompted with "Stop if ". A sample line, showing the prompt and a response, might be:

Stop if 81-85 > '19999' or 1-99 [] 'XYZZY'

The valid commands in either of the conditional definition phases are:

D Define the conditional expression. Input your conditional expression as in the examples above. Terminate keyboard input with the <ENTER> key. Let the screen display wrap around if your expression is larger than 80 characters -

hit <ENTER> only to terminate the input.

U Undefine an existing conditional expression. "U" deletes the current conditional. If you exit without creating a new one you won't have any conditionals.

V View the current conditional expression. FXED shows you the current conditional expression with parentheses inserted to show how logical expression evaluation has been grouped.

? List the commands which may be used in this phase.

<ESC> Return to main menu.

The secondary translation definition phase

This phase in the creation of an FXED control file is optional. SECONDARY TRANSLATION allows you to specify that input data, after any translation specified in the FXED Translation Definition Phase, is to be run through one more "CHARACTER-TO-CHARACTER" translation process.

If need be, you can use secondary translation to define a complete, custom translation for each of the 256 possible characters which can be represented in 8 bits.

A typical use of this feature is to re-define troublesome characters that would cause problems with your application software if they were passed thru without change.

Secondary translation is performed:

- On all input data (after any primary translation - e.g. EBCDIC to ASCII) before being transferred to the output record, and

- On input data (after primary translation) before it is used in a comparison within a conditional logical expression.

Secondary translation is NOT performed on literal data you have defined with FXED, e.g. in the output field definitions and in logical comparisons.

To get to the secondary translation menu, select S from the main menu. HERE'S WHAT YOU'LL SEE:

```
FXED - Secondary Translation Commands
       T          Enter Translation Data.
       C          Change all control characters (00-31 dec) to
SPACEs.
       V          View active Translations.

       ?          Displays this list of commands.
     ESC          Return to Main Menu.
```

To enter one or more translations hit T.   You'll see, e.g.:

Enter "from" character, then "to" character.

Characters may be entered in 3 ways:

By hitting the corresponding keyboard key(s).

In HEX. Press <F6> 1-2 HEX digits <F6>

In DECIMAL. Press <F10> 1-3 DECIMAL digits <F10>

Hit <END> to return to Secondary Translation Menu.

SAMPLE DIALOGUE:

Translate from $ to #

Translate from @ to *

Translate from <END>

To DELETE secondary translation definitions, simply re-define the character to translate to itself.

To view the existing translations, hit V, you'll see, e.g.:

```
Format: Character(DEC/HEX)

$(036/24) -> #(035/23)

@(064/40) -> *(042/2A)
```

Converting all control characters to spaces

Selecting C from the Secondary Translation menu lets you re-define all control characters (0-31 decimal) to SPACEs, in one fell swoop.   This is for convenience, you could make these re-definitions one at a time, but that takes lots of individual swoops.

Generate a .dbf file

OVERVIEW: FXED can automatically create an empty dBASE compatible .DBF structure from the information found in an FXED control file. The DBF structure is ready to receive data directly from tape via FX's fast, unique "Tape-to-DBF" feature. You're guaranteed a perfect match because the same output field definitions which will process the tape data are used to define the .DBF structure. Note that FXED's "Disk-to-Disk" mode can not append a disk data file to a .DBF file.

ADDITIONAL FEATURES: Secondary commands in the G phase can also be used to View and/or Print the structure of any .DBF file, whether created by FXED or not. FXED displays more information about a .DBF structure than dBASE and workalikes - the range of bytes in each field is also shown.

HOW .DBF FIELDS ARE NAMED

The .DBF fields are named, whenever possible, from the first word in the FXED control file's output definition comment field. Only the first ten characters of the first word of each comment are considered in creating the .DBF field names.

If a valid field name can not be constructed from the first word in the comment field (e.g. no comment, first word invalid as a DBF field name, field name already used), FXED constructs a name consisting of a single alpha character plus the control file's output definition line number, e.g. "A010".

FIELD LENGTHS & FIELD TYPES

By using certain leading characters in the control file's comment fields you can force other-than-default field lengths and field types in the DBF file being produced. Leading SPACEs are ignored.

The defaults, when no special leading characters are encountered, are CHARACTER fields, ONE per control file output field, the same length as the output field.

Special leading characters are required if the DBF field is to be NUMERIC, DATE, LOGICAL or to include more than one FXED output field.

The special leading characters that will have an effect on how FXED processes the control file, are:

```
 +  NUMERIC
 -  NUMERIC
 /  DATE
 ?  LOGICAL
 ^  Include this field as an extension of the DBF field started above.
```

If you want a field in the DBF structure to consist of more than one FXED output field, start the next comment(s) with an up-arrow (^) character. Think of the up-arrow as pointing up to the previous field definition. It has the effect of causing the length and type of the previously started DBF field to be continued to include the current field definition.

FIELD LENGTH RULES: Date fields must add up to being 8 characters long. Logical fields must be 1 character in length. FXED will refuse to generate wrong-length fields.

MORE .DBF INFORMATION: Detailed information on the internal format of .DBF files and typical tape files is presented in the separate FX and D2T chapters. We suggest that you become familiar with this information.

PLUS and MINUS: There is no difference between using + or - to create a NUMERIC field type, other than the comment itself possibly being more meaningful to someone reading it.

DECIMAL POINT POSITIONS are automatically created by the "G" command whenever it is processing the continuation of a numeric field definition and the purpose of the line is to insert a literal decimal point. This is demonstrated in the following examples (lines 05, parts 2 & 3).

.DBF file generation

1. The following table is the record layout information that came with a tape. We're going to extract only the fields we want, in the order we want, and automatically generate an empty .DBF file, ready to receive the extracted data from tape. It is implied, that after we're finished with the work shown in this example, we'll exit the FXED program and run the FX program to process the tape. We'll tell FX to use the FXED control file (EMP.CF) to extract and rearrange the fields and to transfer the data into the .DBF file we've created here (EMP.DBF).

FIELD #	CONTENTS	STARTING POSITION	LENGTH
1	LAST NAME	1	15
2	FIRST NAME	16	10
3	MIDDLE INITIAL	26	1
4	SOCIAL SECURITY NUMBER	27	9
5	DEPARTMENT	36	4
6	SALARY (DDDDDDCC)	40	8
7	BENEFIT CODES	48	12
8	DATE OF HIRE (MMDDYY)	60	6
9	AMOUNT OF LAST INCREASE (DDDD)	66	4
10	EMPLOYEE NUMBER	70	6
11	BURDEN RATE	76	4
12	SECURITY CLEARANCE	80	5
13	MINORITY MEMBER (Y/N)	85	3

2. This is a commented screen dump of you entering the data in the O phase. (After picking O from the Main Menu):

(FXED stuff) THE STUFF BELOW HERE (except the semi-colons) IS YOUR ACTUAL INPUT. COMMENTs terminated with <ENTER>.

```
 (01:001)   1-15 <ENTER>        ; LAST_NAME
 (02:016)   L10 <ENTER>         ; FIRST_NAME
 (03:026)   L1 <ENTER>          ; MI (middle initial)
 (04:027)   40-45 <ENTER>       ; +SALARY (dollar part)
 (05:033)   <INS>.<INS>         ; ^ insert a decimal pt.
 (06:034)   L2 <ENTER>          ; ^ this is the CENTS
 (07:036)   <INS>19<INS>        ; /HIRE_DATE (literal 19)
 (08:038)   64-65 <ENTER>       ; ^ the YY from tape
 (09:040)   60-63 <ENTER>       ; ^ the MMDD from tape
 (10:044)   70L6 <ENTER>        ; +EMP_NUM  (employee #)
 (11:050)   80L5 <ENTER>        ; SEC_CLEAR (clearance)
```

3. The following printout shows the above definitions as displayed by the V or P command.

```
 01: I = 001-015  O = 001-015  L = 015  FIELD     ; LAST_NAME
 02: I = 016-025  O = 016-025  L = 010  FIELD     ; FIRST_NAME
 03: I = 026-026  O = 026-026  L = 001  FIELD     ; MI (middle initial)
 04: I = 040-045  O = 027-032  L = 006  FIELD     ; +SALARY (dollar part)
 05:              O = 033-033  L = 001  "."        ; ^ insert a decimal pt.
 06: I = 046-047  O = 034-035  L = 002  FIELD      ; ^ this is the CENTS
 07:              O = 036-037  L = 002  "19"       ; /HIRE_DATE (literal 19)
 08: I = 064-065  O = 038-039  L = 002  FIELD      ; ^ the YY from tape
 09: I = 060-063  O = 040-043  L = 004  FIELD      ; ^ the MMDD from tape
 10: I = 070-075  O = 044-049  L = 006  FIELD      ; +EMP_NUM (employee #)
 11: I = 080-084  O = 050-054  L = 005  FIELD      ; SEC_CLEAR (clearance)

Output Record Length = 54 bytes.
```

Here you hit <ESC> and select G from the Main Menu.

.DBF FILES, Examples of "G" phase commands

4. This shows the screen dialogue used to GENERATE and empty .DBF file.

(After picking G from the main menu):

FXED - Generate .DBF Structure

Input name of .DBF file or <ENTER> alone for EMP.DBF: <ENTER>

```
FXED - Generate .DBF Structure Commands
      G       Generate .DBF structure in file EMP.DBF
      V       View .DBF structure of file EMP.DBF
      P       Print .DBF structure of file EMP.DBF
      ?       Displays this list of commands.
    ESC       Return to Main Menu.
Hit G,V,P,<ESC> or ? for HELP: G
Generating EMP.DBF from EMP.CF.    Done.
```

5. This is how FXED DISPLAYS or PRINTS the file EMP.DBF.

Note the extra information (Range) not provided by dBASE and friends.

```
Hit G,V,P,<ESC> or ? for HELP: V (view on screen)  or  P (print to LPT1)
Structure for database    EMP.DBF
Number of data records:   0
Date of last update:   01/06/97

Field  Field Name  Range    Type      Width  Dec

    1  LAST_NAME   001-015  Character    15
    2  FIRST_NAME  016-025  Character    10
    3  MI          026-026  Character     1
    4  SALARY      027-035  Numeric       9    2
    5  HIRE_DATE   036-043  Date          8
    6  EMP_NUM     044-049  Numeric       6
    7  SEC_CLEAR   050-054  Character     5
```

DISK FILE INPUT (Select D from the main menu)

OVERVIEW: Normally you use FXED to produce a control file that other programs will use in processing tape files. In "Disk Input" mode FXED itself performs the processing, taking the input data from a disk file. The control file you are currently editing provides the processing instructions.

Output is to any DOS file or device, you can select the screen or printer for output and see the results immediately. Note that FXED can't append a disk data file to a .DBF file. It's the FX program that can transfer tape data directly to a .DBF file.

PURPOSES:

Debug: To provide a quick means of debugging your FXED control file without requiring you to leave FXED, run a trial FX job, return to FXED, edit, return to FX etc. You can use disk-to-disk mode to create and fully debug your control file on any PC (no need for a tape drive) before running the production job on the PC that has the drive.

Transfers to diskettes: Disk-to-Disk mode facilitates the fast, quiet processing of large "Tape-to-Floppy" jobs with the added insurance of "check pointing" so that jobs interrupted for any reason can be continued without restarting from the beginning. Output records are never split between diskettes- i.e. partial output records are never written.

CREATING AN INPUT DISK FILE FOR USE IN DEBUGGING YOUR CONTROL FILE: Use FX to transfer several blocks of raw data from tape to disk. The data in the disk file must be an exact, unmodified copy of the tape data. Answer "N" for No to all the questions FX asks which would otherwise cause FX to perform processing of the input data. FX allows you to easily transfer only part of a tape file to disk - in case you've missed this feature in the FX chapter, here's an example:

To transfer only the first 12 physical blocks of tape file 0 to disk, when FX asks you to input the tape file number respond:

"0(1-12) <ENTER>"

RUNNING THE JOB: Select D from the main menu.

You're asked to input the source file name, then the target file name.

The target file may be:

Any valid disk 8.3 filename (with drive and pathname if you like),

or "CON" for screen output,

or "PRN" for printer output.

Then FXED says: "Processing......" and eventually "Done. Press <ESC> for main menu".

SAMPLE "DISK-TO-DISK" JOB

```
Disk File Processing.        Hit <CTRL-C> at any time to ABORT, <S> for STATUS.
Enter source file name: DATA.RAW
DATA.RAW contains 14166 142 byte records.
Enter target file name: 079XX.OUT
Skip how many input records? (<ENTER> alone = 0):<ENTER>
Processing DATA.RAW to 079XX.OUT with FXED Control File LIST20.CF.
14166 Input Records Processed.
3526 Output Records Created.          Done.     Press <ESC> to for main menu
```

EXTRA BENEFIT: Like FX, FXED allows you to transfer large (larger than a diskette) files to multiple diskettes. When a diskette is full you're asked to insert another formatted diskette, input a file name and hit <ENTER> (or hit <ENTER> alone to abort). The DOS "COPY" command doesn't provide such friendly handling of the "diskette full" situation. You can use FXED instead of DOS's "BACKUP" command to move a large hard disk file to multiple diskettes; define input and output record lengths of 1 and no translation.

DISK FILE INPUT MORE NIFTY FEATURES, (SKIPPING INPUT RECORDS, CHECKPOINTING & SILENCE):

Tape to Floppy Jobs

If you are running a "tape to diskette" job where you would use an FXED control file, and you have room on your hard disk (e.g., as determined by running "CUBS A"), you should consider transferring the RAW data to hard disk with FX (don't invoke any options) and creating the diskettes using FXED's Disk File Input mode.

You can turn off the tape drive (and any noise it makes) after transferring the data to hard disk. FXED writes "checkpoint" information on each output floppy. The checkpoint info, together with FXED's ability to start a disk-to-disk transfer after skipping a specified number of input records, allows you to continue a disk-to-diskette job without starting from scratch after being interrupted for any reason, e.g. long lunch, power failure, or running out of formatted diskettes.

Here's an example of a disk-to-diskette job. It processes a mailing list, extracting only certain ZIP codes (that's why we called the output files "07XXX"). It fills up the floppy in drive A and we continue with one in drive B.

```
Disk File Processing.        Hit <CTRL-C> at any time to ABORT, <S> for STATUS.
Enter source file name: DATA.RAW
DATA.RAW contains 14166 142 byte records.
Enter target file name: A:07XXX.OUT
Skip how many input records? (<ENTER> alone = 0):<ENTER>
Processing DATA.RAW to A:07XXX.OUT with FXED Control File LIST20.CF.
Current disk is full. 5071 input, 3273 output records so far.
Please insert a blank, formatted disk or hit <ENTER> alone to abort.
Enter disk file name: B:07XXX.OUT
14166 Input Records Processed.
3526 Output Records Created.     Done.
```

CHECKPOINT INFO ON EACH DISKETTE

```
FXED writes two zero-length files on each diskette.
The file names are of the format nnnn.IN  and  nnnn.OUT.
The name of the IN file is the cumulative number of INPUT records processed so
far.
The name of the OUT file is the number of OUTPUT records written on this diskette.
Checkpoint information is written to removable output media only - FXED won't clog
up your hard disk with useless files.
Checkpoint information doesn't waste any room on the floppies - DOS has already
reserved a block of bytes for a bunch of filenames.

Here's the directory from the diskette in drive A: from the above job:
07XXX    OUT   340392   1-17-96   9:51p
5071     IN         0   1-17-96   9:51p
3273     OUT        0   1-17-96   9:51p
```

RUNNING FXED ENTIRELY FROM THE COMMAND LINE IN "DISK-TO-DISK" MODE:

A number of customers use FXED's "Disk File Input" mode as a powerful, fixed-length-record manipulator. Some even "shell out" to FXED's Disk-To-Disk mode from running database programs. The command line arguments which facilitate this are DFI=, DFO= and RTS=. They let you define the Disk File (for) Input, Disk File (for) Output and (optional) Records To Skip. Remember, the first command line argument must be the name of the existing control file. If a control file, input file and output file are specified on the command line, FXED will just run the disk-to-disk job without stopping for any console dialogue, then return to DOS.

Formally: FXED <controlfile[.CF]> DFI=<inputfile> DFO=<outputfile> [RTS=<# of input records to skip>]

Example: FXED CTRL DFI=C:\TAPEDATA\ABC.RAW DFO=D:\OUTPUT\ABC.OUT <ENTER>

CHANGE INPUT RECORD LENGTH (Select L from the main menu)

The "L" command allows you to re-define the input record length for the FXED control file you are editing. The maximum input record length is 32,752.

FILE MAINTENANCE (Select F from the main menu)

The "F" command takes you to a sub-menu that allows you to:

Input a different control file from disk, or

To re-define the name which will be used when the control file is written to disk, or

To clear ALL definitions currently in effect for the control file being edited.

Here's the "F" sub-menu:

```
FXED - File Management Commands
Editing: DEMO
       N          Get (N)ew control file from disk.
       R          (R)ename file being edited.
       C          (C)lear all current definitions.
       ?          Displays this list of commands.
     ESC          Return to Main Menu.
```

PRINT ALL DEFINITIONS, IN DETAIL (Select P from the main menu)

Use this command from the main menu to print the FXED control file name, followed by all of the definitions, for all phases, in detail.

The printout you'll get is a good hardcopy record to save once you have a control file which performs as you intended.

If you're still "fine-tuning" the control file, it's a helpful working document to use in approaching the desired results.

If no data has been specified in the optional phases (Conditional Record Processing and/or Secondary Translation) there will be no output for these phases.

VIEW ALL DEFINITIONS

Similar to the "P" command above but the results are output to the screen.

See below for examples of output from the "V" and "P" commands......

EXAMPLE OF "V" and "P" OUTPUTS (With editorial comments.)

FXED Control File: DEMO.CF

Input record Length = 100 This is the length of the input records.

TRANSLATION DEFINITIONS:

```
      1-50E                          We've defined some bytes that won't be used
     51-55P                          in the output; this isn't necessary but it
     56-70E                          doesn't hurt.  Note the Packed and Zoned decimal
     71-74Z                          translations and see comments on them below in
    75-100E                          lines 08: and 10:
```

OUTPUT DEFINITIONS:

```
01:             O = 001-001  L = 001  '"'    Insert a double quote before first character field.
02: I = 001-012  O = 002-013  L = 012  FIELD  The word "FIELD" means a field from tape.
03:             O = 014-016  L = 003  '","'  Insert a double quote after first character field, then a
comma as
                                             field separator, then a double quote before the next
                                     char. field.
04: I = 013-030  O = 017-034  L = 018  FIELD
05:             O = 035-037  L = 003  '","'
06: I = 075-086  O = 038-049  L = 012  FIELD
07:             O = 050-051  L = 002  '",'   Insert a double quote to terminate previous
                                             character field, then a comma to delimit the
                                             following numeric field.

08: I = 051-055  O = 052-061  L = 010  FIELD  Bytes 51-55 of the input (which are Packed
                                             Decimal) become bytes 52-61 of the output.
                                             Note that 5 bytes of input become 10 bytes of the
                                             output - be sure to leave room in your database program's
                                             field definitions.

09:             O = 062-062  L = 001  ","    Insert a comma to separate 2 numeric fields.

10: I = 071-074  O = 063-067  L = 005  FIELD  Bytes 71-74 of the input (which are Zoned decimal) will
                                             become bytes 63-67 of the output - be sure to leave room
                                     for
                                             this in your database program's field definitions.
11:             O = 068-069  L = 002  ",""
12: I = 033-050  O = 070-087  L = 018  FIELD
13:             O = 088-090  L = 003  '","'

14: I = 087-100  O = 091-104  L = 014  TRUN FLD Bytes 87-100 of the input become >= 0
                                               bytes of the output (14 bytes maximum). Truncated
                                            output
                                               fields are those defined with the "\" character - this
                                            causes
                                               trailing SPACE characters to be deleted.
                                               This can save lots of room on disk.

 15:             O = 105-107  L = 003  '"<CR><LF>'  The <CR><LF> was input by hitting
                                                   <INS><ENTER><INS> This terminates each output
                                                     record with a <CR><LF>.
Output Record Length = 107 bytes or less (1 or more variable length fields present)
```

CONDITIONAL CRITERIA:

Accept if 56-60 >= "10000" and 56-60 <= "10999" Only input records which contain 10000 - 10999 in positions 56-60 will be output.

SECONDARY TRANSLATIONS:

Here we make sure that any double quotes that may be found on tape will be translated to single quotes so that the database program won't confuse them with character field delimiters.

Format: Character(DEC/HEX)

"(034/22) -> '(039/27)

NUMERIC DEFINITIONS: PACKED & ZONED DECIMAL, PACKED HEX/BCD, EBCDIC LOWER CASE

From a translation standpoint, so called EBCDIC tapes aren't always 100% EBCDIC - they often contain numeric fields that can not be translated on a one-to-one character basis from EBCDIC to ASCII. Numbers on EBCDIC tapes are frequently found in PACKED DECIMAL and ZONED DECIMAL, and less often in PACKED BCD/PACKED HEX. These special numeric codes are even (infrequently) found on otherwise pure ASCII tapes generated by COBOL programs.

The purpose of this section is to define the numeric codes which FXED can translate to ASCII (STANDARD EBCIDC NUMBERS, LOWER CASE EBCDIC NUMBERS, ZONED DECIMAL, PACKED DECIMAL and PACKED BCD/PACKED HEX) and to teach you to recognize them in case they have not been clearly identified as such in the documentation which came with your tape. Remember, if need be, you can import fields with NO TRANSLATION using the =.

TERMINOLOGY: BYTE = 8 bits. NIBBLE = 4 bits.

HIGH-ORDER = leftmost = most significant. LOW-ORDER = rightmost = least significant

STANDARD EBCDIC NUMBERS: The values 0, 1, 2 ... 9 are represented, one byte per digit, by the hex values F0, F1, F2 ... F9 respectively. This makes them easy to recognize in a hex dump and their value is apparent upon inspection.

LOWER-CASE EBCDIC NUMBERS: The values 0, 1, 2 ... 9 are represented, one byte per digit, by the hex values B0, B1, B2 ... B9 respectively. This makes them easy to recognize in a hex dump and their value is apparent upon inspection.

PACKED DECIMAL: In packed decimal, each byte (except the low order byte) represents 2 decimal numbers, one in the high-order nibble and the other in the low order nibble. The sign of the packed decimal string is contained in the low- order nibble of the low-order byte. Packed Decimal is sometimes called "COMPUTATIONAL 3" or "COMP 3" in the mainframe world.

Positive values have a hex C or F as the low-order nibble. Negative values have a D as the low-order nibble. Any other value in the sign nibble is illegal. The character "?" is output for an illegal sign specification.

```
VALUE          ON TAPE (HEX)   FX/FXED ASCII OUTPUT
+1234          01 23 4C        " 01234"
+1234          01 23 4F        " 01234"
-1234          01 23 4D        "-01234"
```

PACKED BCD/PACKED HEX looks like packed decimal but the least significant nibble is NOT treated as an indication of the sign of the data. Use FXED's H translation type to have it converted to ASCII. Each byte will yield 2 characters.

The relationship between the number of bytes input and the number of ASCII characters output for each packed decimal field is:

(NUMBER OF PACKED DECIMAL INPUT BYTES) x 2 = NUMBER OF ASCII OUTPUT BYTES

ZONED DECIMAL: Zoned decimal numbers are similar to standard EBCDIC numbers in that every digit occupies one byte. The sign of a zoned decimal field, however, replaces the high-order nibble of the least significant byte.

If the high-order nibble of the last byte is A, C, E, or F the value is positive. If the high-order nibble is B or D, the value is negative. Anything else (0-9) is illegal. Electrovalue software outputs the substitute sign character "?" when an illegal sign specification is detected.

```
VALUE          ON TAPE (HEX)   FX/FXED ASCII OUTPUT
+1234          F1 F2 F3 A4     " 1234"
+1234          F1 F2 F3 C4     " 1234"
+1234          F1 F2 F3 E4     " 1234"
+1234          F1 F2 F3 F4     " 1234"
-1234          F1 F2 F3 B4     "-1234"
-1234          F1 F2 F3 D4     "-1234"
```

NOTE: The double quotes are not part of the output. They are included in these examples to show the character placed at the left for positive values.

The number of output bytes (ASCII characters) for each zoned decimal field is one byte more than the number of input bytes.

LOOKING AT TAPE WITH THE KEYS and DITTO PROGRAMS:

Use KEYS' R, CTRL-N, D, N and E commands to find and easily identify the various numeric types on any tape.

See the KEYS and DITTO chapters. DITTO allows you to display records in HEX and as characters, preceded by a printout of byte position numbers.

The KEYS Program

"SINGLE-KEY-COMMAND" TAPE ANALYSIS UTILITY & HARDWARE TEST PROGRAM

Additional supported equipment: Line printer.

PURPOSES:

ANALYSIS OF TAPE STRUCTURE AND CODE   The user who is mainly interested in transferring data from another computer to the PC can use the R and D commands to easily and quickly determine the format, code and structure of unknown tapes. This user may also find the data display commands E, A and N to be useful. These 3 commands allow the optional insertion of a CR/LF every xx characters that is helpful in determining the logical record length of tape data. In addition, the C command checks for a possible labeled tape and displays any useful information gleaned from the label.

HARDWARE TEST AND TROUBLESHOOTING   KEYS allows step-by-step control of the tape drive from the keyboard. KEYS is used by computer technicians for test and diagnostic purposes. KEYS may also be used by an Electrovalue customer as instructed (e.g. on the telephone) to help diagnose operational problems.

CAUTION
KEYS CAN EASILY WRITE ON TAPE IF YOU HIT THE WRONG KEY. WRITE PROTECT THE TAPE FIRST BY REMOVING THE WRITE RING OR PUSHING THE THUMBWHEEL TO THE LOCKED POSITION.

A summary of keys commands

Type the command letter at the keys prompt (the Yen sign (¥)).

The effect is immediate, i.e. the <ENTER> key is not needed.

```
     @ - Print this list of commands in reverse order.
     F1 - Abort current operation.
      * - Toggle output to the printer ON or OFF (A, D, E, N, and C).
  (xx)A - Display buffer in ASCII adding CR/LF each (xx) characters.
  (xx)B - Space by xx Blocks. (xx is negative for back blocks.) [M]
      C - Check tape for an initial label & display information if found.
      D - Display buffer in NUMERIC (DECIMAL or HEX) - ASCII - EBCDIC
     -D - Same as "D" but displays end of buffer.
  (xx)E - Display buffer in EBCDIC (IBM code) adding CR/LF each xx chars.
  (xx)F - Space by xx Files.  (xx is negative for back files.) [M]
xCTRL-F - Fast forward x Files while checking for end of data.
      G - Write erase Gap. [M]
      H - Display Hardware Status. [M]
      I - Write a Filemark.  [M]
      J - Jam the buffer (of size set by S) full of copies of the BIOS.
 [-]xxK - Kick out the buffer with block lengths of xx. [M]
      L - Rewind tape to Load point.
  (xx)M - Multiple.  Repeat  [M] commands xx times.  M alone resets.
  (xx)N - Numeric (DECIMAL or HEX) buffer display, CR/LF each (xx) bytes.
     -N - Same as "N" but displays the end of buffer.
 CTRL-N - Toggle Numeric display mode between DECIMAL and HEX. (D, N).
      O - unload tape.
  (xx)P - Fill the write buffer with a Pattern where xx is 0-255.
      Q - Quit the keys program.
      R - Read a block from the tape. [M]
  (xx)S - Set size of block to be used by W, K & J.  S alone displays size.
      T - Triple.  Does W, -1B, R. [M]
      U – Duplicate last command again. [M]
      W - Write a block to tape.  Block size is set by S command. [M]
      Z - Zap tape.  Erase entire tape for security purposes.
      F5 - Decrease CR/LF 'N' by 1 and redisplay buffer. (A & E).
      F8 - Increase CR/LF 'N' by 1 and redisplay buffer. (A & E).
      ? - Print this list of commands.
```

Overview of keys operation

When KEYS is first executed it calls DOS to allocate a 64K I/O buffer. All read, write and buffer fill ("P" command) operations take place with respect to this same buffer.

Commands which display the contents of the buffer are terminated at the length of the last block read in from tape or by an "S" (set write buffer size) command if an S command has followed the last Read.

Each KEYS command consists of a single character. Some commands may be preceded by a decimal argument; sometimes this number is optional. It is never necessary to hit the <ENTER> key. KEYS commands are executed when the command key is hit.

DETAILED EXPLANATION OF INDIVIDUAL COMMANDS

F1 - Abort current operation

"F1" aborts any operation in progress. If hit during multiple mode execution "F1" returns to the prompt - multiple mode remains in effect. F1 also aborts the Z command before the tape is erased. F1 will not abort the tape while the actual erase operation is in effect.

* - Toggle print mode ON or OFF

Hit the * to toggle the print mode on or off. You'll get a print mode status message. When print mode is ON, the A, C, D, E and N commands send a copy of the screen output to the printer (LPT1). The printer must be on and ready to go otherwise an error will be generated and the program will abort.

? – Prints the list of the KEYS commands.

@ - Prints the list of KEYS commands in reverse order so the ones that passed by the screen in the ? command are visible.

(xx)A - Display buffer in **A**SCII adding CR/LF each (xx) characters.

This command is normally used after an 'R' (read in a block from tape).

The contents of the I/O buffer are displayed as ASCII characters on the screen, 80 bytes per line. To force a full, 80 character wide screen display CR and LF (if present in the buffer) are translated to decimal 227 and 236 respectively. On the screen CR will show up as the "Π" character, LF as "∞". If print mode is active (see the * command) a copy of the display will be sent to the printer port (LPT1). An IBM compatible printer will print the above CR and LF graphics, otherwise you'll get "c" for CR and "l" for LF. The printable characters (decimal

33-126, hex 21-7E) are displayed directly. All other characters are displayed as decimal points, NOT as their screen graphics. If 'n' is specified, e.g. 123U, KEYS will start a new display line (output an actual CR/LF) every n characters. This is helpful in determining the logical record length of tape data.

The displayed record length and the blocking factor are displayed at the bottom of the display.

Using 'A' to find the logical record length of ASCII data:

Read in a data block with the 'R'command.

Press <A>.

Judging from the displayed record you can guess a length and type in xxA where xx is the guessed length and hope the data comes close to lining up.

Caution: The records will also line up when the displayed record length is a multiple of the actual record length also. IE: If the actual record length is 150, displaying the buffer with a command such as 300E will cause the records to line up also. If you input a tape and find that you have half the records the sender says you should have, you probably picked the record length to be twice the actual record length.

If the record length looks smaller (in this case we will assume a possible length of 55) you can press the F5 key to decrease the displayed length and display the buffer with the new length again. Repeatedly pressing F5 will decrease and display the length again. As the displayed length approaches the actual length, the data will appear to line up. When the correct data length is hit, the data will appear in columns, the length will appear on the bottom of the display, and the blocking factor will also appear on the display.

If the record length is greater than 80, pressing F8 will increase the displayed length and display the buffer.

Important The F5 and F8 keys only have an effect immediately after pressing the "A"' or "E" keys if another key is pressed the F5 and F8 keys loose their effect. Issue another "A" or "E" command to reactivate the F5 and F8 keys. Of

xxB - Space xx **B**locks. (xx is negative for back blocks.) [M]

The "B" command spaces the tape forward or reverse by blocks. If a filemark is encountered spacing is halted immediately after the filemark and you are notified of how many blocks were spaced. For example, "7B" spaces forward by 7 blocks, "-3B" spaces back 3 blocks.

If you input 7B and a filemark is encountered after 2 blocks, spacing is halted after the filemark is passed. You will be notified that a filemark was encountered and spacing was halted after two blocks.

C - **C**heck tape for an initial label & display information if found.

This command checks the tape for a label. If there is an initial label the relevant information will be displayed. Note: This command only checks the first file on tape. It does not analyze the complete tape. If you want to analyze the whole tape you should use the CUBS program in the L mode to check labeled tapes.

(-)D - **D**isplay buffer in NUMERIC - ASCII - EBCDIC

The contents of the I/O buffer are displayed on the screen, 12 bytes per line, side-by-side, in NUMERIC, ASCII and EBCDIC (converted to ASCII). CR, LF and SPACE are all displayed as SPACE. The printable characters (decimal 33-126, hex 21-7E) are displayed directly. All other characters are displayed as decimal points, NOT as their screen graphics. If print mode is active (* command) a copy of the display will be sent to the printer port (LPT1).

This command is useful in checking out whether the data is in ASCII or EBCDIC. It is also helpful if you need a decimal or hex dump.

The numeric mode is DECIMAL or HEX as set by the CTRL-N command.

If "-D" is input only the last few lines of the buffer are displayed.

(xx)E - Display buffer in **E**BCDIC (IBM code) adding CR/LF each xx chars.

Same as “A” but displays data in EBCDIC. See the “A” command

xxF - Space xx by **F**iles. (xx is negative for back files.) [M]

The "F" command spaces the tape forward or reverse by file marks. For example, "7F" spaces forward by 7 file marks, "-3F" spaces back 3 file marks.

NOTE: When spacing in reverse by file marks the requested number of filemarks +1 are passed in the reverse direction, then the tape is spaced forward over one filemark. This assures that the tape drive read/write head is never left "behind" a filemark and provides a logical compatibility between spacing in the forward and reverse directions. For example, if the tape is sitting just past a filemark and you space forward "N" files, then backwards "N" files the tape will end up positioned just where it started.

xxCTRL-F - **F**ast forward x files while checking for End of Data

The tape moves forward at high speed until xx files have been passed or until "End- Of-Data" (2 filemarks in a row) is encountered. EOD detection in not a function of the hardware.

G - Write erase **G**ap. [M]

"G" writes 3 or 4 inches of blank tape. Some tape drives write 3 inches, others write 4 inches. This command could be used to erase a section of tape. For instance, you have one or more files on tape and you want to erase all data after this file. You could the KEYS Fast forward files spacing commands to get to the end of data. This will put you between the last two filemarks. Then space forward one block. You will hit the last filemark and stop just past it. Next hit 500M. Then hit G Keep hitting G until the drive EOT.

H - Display (H)ardware Status. [M]

"H" displays a snapshot of the tape drive status. F means the condition is false, T means it is true. Note that not all drives return BOT status to the controlling program. This means that H is not always able to detect BOT status at BOT.

I - Write a **fi**lemark. [M]

Each time 'I' is pressed a filemark is written to the tape.

J - **J**am the buffer (of size set by S) full of copies of the BIOS.

A quick way to fill the I/O buffer with something known. Otherwise, after a fresh boot, the buffer is likely to be filled with zeroes. The buffer size should be set using the 'S' command before this command is issued.

[-]xxK - **K**ick out buffer contents in block lengths of xx in streaming mode. [M]

The buffer is written out in streaming mode in blocks of length "xx". This command allows you to divide up the buffer into parts of length xx and write the blocks out to tape. If the buffer size is not evenly divisible by xx, the remainder is dropped. The optional "-" sign allows the remainder to be added on as a short block.

Example: The buffer size is set to 450. Giving the command "100K" will write out 4 blocks of 100 bytes. The command "-100K" will cause 4 blocks of 100 bytes and 1 short block of 50 bytes to be written.

L - Rewind tape to **L**oad point.

"L" sends a rewind command to the drive causing the drive to rewind to the load point (BOT reflective marker). This command tells you that the rewind command has been sent to the drive or, if the drive returns BOT status to the software and the drive is at BOT, that the drive is already at load point. Note: Some drives do not return BOT status to the drive when they are idle.

(xx)M - (M)ultiple. Repeat [M] commands xx times. M alone resets.

Some commands may be automatically repeated a multiple number of times. The commands which qualify for multiple execution are identified in the menu and in the KEYS chapter. The multiple counter is set by preceding the M command with a decimal number, e.g. "999M " sets the counter to 999. For example, if "999M" is followed by "W", KEYS will write 999 blocks on tape.

Some Typical Uses of "M"

Write Deskew: Set the buffer size to a large number, e.g. "55555S". Fill the buffer with all "1" bits, i.e. "255P". Set the multiple counter to a large number, e.g. "44444M". Then hit "W" to start the drive writing all 1's. Scope the appropriate point(s) while adjusting the write deskew pots until optimum write deskew is achieved.

Checking for an intermittent read or write problem in a cable or subassembly: Set the write buffer to a small size, e.g. "80S". Set the multiple counter to a large number and execute the "T" command (see below). While the test proceeds, abuse the suspected cable or component (wiggle, rap, heat, spray cool etc.) while listening for the beep that occurs each time a read error is detected. When you can make the beep come or go you've identified the perpetrator.

Scoping for whatever purpose: Multiple mode can be used to follow a signal with an oscilloscope. Set up to write small blocks a bunch of times or, if you want to follow a read signal, use the M mode to write the tape first, then rewind it and do a multiple "R".

Adjusting ramp times: Set the block size small and the multiple counter high and initiate a series of writes while checking and adjusting the start and stop ramp times with a scope.

STOPPING & RESETTING "M" MODE:

The F1 key will abort any operation in progress and take you back to the prompt.

When you're at the keys prompt, hitting M alone will reset multiple mode.

Reflective Markers: Hitting end of tape (EOT) or beginning of tape(BOT) markers during a multiple operation will abort the operation in progress and take you back to the prompt. Multiple mode will remain in effect.

(-)(xx)N - Display buffer in **N**umeric mode.

More economical and faster than "D" since you get 20 decimal numbers or 40 HEX numbers per line in this mode. You select decimal or HEX with the CTRL-N command. If print mode is active (see the * command) a copy of the display will be sent to the printer (LPT1). If 'xx' is specified, e.g. 123N, KEYS will force a new display line (output an actual CR/LF) every xx bytes.

If "-N" is input only the last few lines of the buffer are displayed. The '-' and 'xx' options should not be mixed.

CTRL-N - Toggle back and forth between DECIMAL and HEX modes. Affects "D", "N".

(xx)P - Fill write buffer with a **P**attern.

P has 2 modes.

"P" alone fills the buffer with 10 different groups of bytes, one group after the other. The groups consist of: 255, 0, 1, 2, 4, 8, 16, 32, 64 and 128. How many of each you get depends on the size of the write buffer as set with the "S" command. For example, "100S" followed by "P" would set the write block size to 100, then fill the buffer with 10 bytes of 255, followed by 10 bytes of 0, followed by 10 bytes of 1 ... ending with 10 bytes of 128. Try it, i.e. type "100SPN" to see the buffer contents on the screen. Writing and reading this pattern is useful in finding bits which are stuck on, off or shorted together.

If you precede the P by a decimal number the write buffer will be filled with that byte. Valid inputs range from "0P" through "255P". Example: 65P fills the buffer with the letter A. See the ASCII chart in this manual.

Q - **Q**uit the KEYS program.

R - **R**ead a block from tape. [M]

"R" reads the next block from tape into the I/O buffer. If a data block is read the number of bytes read are displayed. If a filemark is read, a message stating this is displayed. If End of tape is hit you'll get message saying so and a BEEP.

Although "R" does not perform any read retries on bad blocks your drive is probably configured to do read retries.

That's all "R" does unless one or more read-errors are detected. In that case, you'll hear a BEEP and see a list of errors.

Possible errors are:

Hard error.	The tape drive could not read a data block.
Corrected error	The tape drive used error correction in reading in data. Data is good.
Input output error	There was a problem transferring data from the tape drive.

Hard error and Corrected errors can be due to a bad tape (e.g. wrong density or poorly written), a dirty read head (clean it) or a hardware problem. Input output error usually indicates a hardware or configuration problem. Note that not all drives return a corrected error status.

After "R" you can display the contents of the buffer using any of the display commands (e.g. "A","E", "D", "N"). The amount of displayed data is set by the length of the block that was just read. You can use "F1" to interrupt the display.

(xx)S - **S**et size of block to be used by W, K & J. S alone displays size.

The default write block size is 120 bytes when KEYS is first executed.

Entering "S" alone will display the current block size.

To change the block size, precede the S with a decimal number. KEYS will accept entries in the range of 0 - 65535 minimum is 1 byte. These minimums are well below the ANSI standard minimum block sizes. Of course you should check with your drive specifications before attempting to write blocks less than 30 bytes or greater than 32767 bytes.

T - **T**riple. Does W, -1B, R. [M]

"T" is essentially a combination of three other KEYS commands, "W", "-1B" and "R" with one additional test; that the length of the block read back is the same as that of the block written. The data written is not compared to the data read back. "T" writes a block, backs up and reads the block. It's a quick check of the tape hardware's basic functions and is useful in multiple mode during service procedures. Any read errors detectable by the hardware, or a block length mismatch, will cause a BEEP and a display of the error details on the screen.

U – D**u**plicate last command again. [M]

This command lets you send the past command again to the drive. For example you enter 50B to space forward by 50 blocks. Enter U to space forward again by 50 blocks.

W - **W**rite a block to tape. Size is set by S command. [M]

"W" simply attempts to write the I/O buffer to the tape. The block size is that which was previously set by the "S" command and is shown on the screen after each "W".

Possible errors:

Hard error	The tape drive could not write a block
Input/ouput error	There was a problem transferring data to the tape drive.

Z - **Z**ap tape. Erase entire tape for security purposes.

By pressing 'Z' you'll encounter a dialog telling you to press F9 if to erase the entire tape or to press F1 or any other command key to cancel. While there is no technical reason to erase tapes before use, this command is included for those who want to erase a tape for security (secrecy) purposes or for those whose customers require that the tape be erased prior to use.

F5 - Decrease CR/LF 'N' by 1 and redisplay buffer. (A & E only).

Most useful when dealing with mystery tapes, this command decreases the displayed logical record length by one and redisplays it.

After reading in a block with the R command, display the block using A or E. A and E default to a display width of 80 characters. You can use F5 to decrease the displayed length of the data in the buffer to help find the actual record length.

For example if the actual record length on tape appears to be somewhere around 60, simple hold down F5 until the displayed length on the bottom of the screen goes to near 60. Obviously, because the data is fixed length, the data will line up when you get to the correct length.

If you pass the correct length, simply use F8 to increase the length. You can toggle between F5 and F8.

Caution: The records will also line up when the displayed record length is a multiple of the actual record length also. IE: If the actual record length is 150, displaying the buffer with a command such as 300E will cause the records to line up also.

If you input a tape and find that you have half the records the sender says you should have, you probably picked the record length to be twice the actual record length.

Important: Note that hitting any key other than F5 or F8 will take you out of the tweaking mode. You must then key in A or E to go back into tweaking mode. Of course, you can prefix A or E with the suspected block length. Ex: 130E displays data in EBCDIC with a carriage return and linefeed every 130 characters. Use F5 or F8 to tweak the record length.

F8 - Increase CR/LF 'N' by 1 and redisplay buffer. (A & E only).

See the description of F5 for this command.

The CUBS Program

TAPE ANALYSIS/HARDWARE TEST PROGRAM FOR FORMATTED TAPE DRIVES

PURPOSES:

CUBS provides high-speed, streaming, tape analysis capabilities. If you've received a tape to be transferred to disk (by the FX file exchange program) and would like to quickly determine the size of each tape file, CUBS' Analyze mode is the tool you'll use. The Analyze mode quickly displays the size of each tape file (in bytes) as well as the file and block structure of the entire tape. Any hard read errors and/or corrected read errors are reported on a file-by-file basis and accumulated for the entire tape. The additional information present in "Labeled Tapes" may optionally be displayed.

CUBS provides a means of life-testing tape drives while accumulating performance statistics.

Finally, CUBS provides the means of writing tapes of known structure for use in testing situations.

We recommend running CUBS in the L mode to analyze unknown tapes. The L mode will analyze any label information if found. If the tape is not labeled, the L mode will give you the same information as CUBS A mode.

CUBS HELP SCREEN:

```
Usage: CUBS mode [options] <ENTER>

Modes:   A - Analyze any tape.                W - Write CUBS compatible tape.
         A- - Analyze any tape, min. output.
         L - Analyze Tape and Display Labels.  R - Read CUBS compatible tape.
         E - Erase Entire Tape.               B - Both Write and Read.

 Options: H - Halt on any Hard Error           U - Unload upon termination
          L - Log to disk file \"CUBS.LOG\".    L:<filename>, log to <filename>
          \ - Repeat specified operation(s).    P - Log to Printer
          N - Display block length changes- (A & L Modes ONLY)
      (nnn - No output if <nnn bytes..      )nnn - No output if >nnn bytes.
         C - Allow Corrected Error Display     A - Auto Abort at End-Of-Data
       nnK - Maximum block size nnK (nn = 1-64).
         S – Slow speed operation.  High speed is default.
       M=n - Maximum Number of blocks to write in W or B modes.
       L=n - All blocks in W or B modes are fixed Length n, no random filemarks
       F=n - Fill Write buffer with n (0 - 255).  Use with 'W' only, don't 'R'
Run time commands:   S - Display cumulative status and continue.
                CTRL-A - Display cumulative status and Abort "
```

HOW TO RUN CUBS:

CUBS is controlled from the command line. When CUBS is run, the screen is cleared, and the program revision level and a copy of the command line are displayed. Then CUBS starts reading and/or writing tape depending on the "mode" specified on the command line.

Command line arguments consist of one "mode" (required) and zero or more "options", separated by blanks. Only one mode may be specified but the mode may be followed by as many options as you like, in any order.

If a valid mode is not specified, CUBS will remind the user that he can input "CUBS ? <ENTER>", to see a help screen containing a list of the valid CUBS command line arguments.

Note: M, the number of blocks specified, counts filemarks too. CUBS searches for 2 filemarks together to mark the end of data. Therefore, when in random mode, it is possible for CUBS to write a filemark directly before the two filemarks that mark the end of data. When reading the tape CUBS will read in one less file or “block” and not give an error. Ex: You tell CUBS to write 250 blocks. Every so often CUBS reads in 249 and reports no error. This is to be expected.

EXAMPLE COMMAND LINE: To Analyze a "mystery tape", type: CUBS L P A U <ENTER>

CUBS will analyze the entire tape checking for **L**abels, Print the results, Abort when 'End-Of-Data' is hit, Unload the tape and quit.

OPERATION WHEN NO OPTIONS ARE SPECIFIED

Normally, CUBS performs one pass on the tape at high speed, prints a summary line, rewinds, and quits. (See the \ and U command line options and the CTRL-A run-time command.)

No messages are output between BOT and EOT while CUBS is writing (W) or reading (A, L and R) unless a hard error is encountered. For hard errors, appropriate error messages are displayed. Corrected read errors are reported on a file-by- file basis only, unless the C option is in effect, then you'll see CER on a block-by-block basis for each read error that the tape drive was able to detect and correct. (Also see the L command line option and the S run-time command)

The analyze modes (A & L) display tape statistics on a file-by-file basis and halts when End-Of-Data is encountered (two adjacent filemarks), with an option to continue. (See the A and L command line options and the S run-time command)

The default is not to halt on error. (See the H option.)

The S command (Write tape at Slow speed) only has an effect if you have a dual speed drive such as a Cipher F880.

The minimum block size written is 12 bytes, the maximum is 65,536 (64K - 1) bytes. (See the K and L options.)

END-OF-DATA: Mainframe tapes, and tapes produced by all Electrovalue programs, write two filemarks at the end of all the data on the tape. This provides a convenient "End-Of-Data" mark. CUBS asks the user (C)ontinue or (A)bort ? when End-Of-Data is detected during a tape analysis. The A option may be used to avoid the question and automatically abort at End-Of-Data.

END-OF-TAPE: The End-Of-Tape reflective marker is ignored during "A" mode. Industry standard tapes, and tapes created by Electrovalue programs, will have an "End-Of-Data" double filemark written shortly after the EOT marker.

A mode

The A mode (Analyze) is used as a quick means of determining the block sizes, number of bytes, file structure and integrity, of ANY tape. As the tape is read, (in high-speed streaming mode if your tape drive supports this), any hard read errors, corrected read errors and read retries are detected.

Statistics are output to the screen on a file-by-file basis.

Here's a sample "CUBS A" screen output: (Use "CUBS A P <ENTER>" to get a printed output at the same time.)

```
     TAPE FILE 000:    22 BLOCKS,   89,277 BYTES.
       BLOCK    1,  1,934 BYTES
       BLOCK   22,    179 BYTES                          89,277 BYTES THIS TAPE.

     TAPE FILE 001:     27 BLOCKS,   79,522 BYTES.
       BLOCK    1,  2,302 BYTES
       BLOCK   27,    437 BYTES                         168,799 BYTES THIS TAPE.

     TAPE FILE 002:      3 BLOCKS,    9,564 BYTES.
       BLOCK    1,    186 BYTES
       BLOCK    3,  4,964 BYTES                         178,363 BYTES THIS TAPE.

       DOUBLE FILEMARKS ENCOUNTERED.    (C)ONTINUE OR (A)BORT?
```

Note that "CUBS A" reports:

- the number of blocks in each file,
- the size of each file in bytes,
- the size of the first block in each file,
- the size of the last block in each file,
- the cumulative number of bytes on the entire tape.

E mode

The E mode completely erases a tape, from beginning to end. A 10.5-inch, 2400 foot reel takes about 4 minutes on a Hewlett Packard 7980. There is no technical reason you would ever need to erase a 9-track tape cartridge. 9-track tape is just like audio tape or a videocassette; you just write on top of any old data. The E mode is provided for users who want to erase old data for security (secrecy) purposes. Using the U option will also unload the tape when done, e.g. CUBS E U <ENTER>

L mode

"L" mode is a special variation of the "A" analyze mode which displays additional information gleaned from the "LABEL" files on so-called "LABELED" tapes. Mainframe operating systems commonly create labeled tapes.

From our point of view (running the FX program to import tape data to the PC) a labeled tape contains extra, small files of minimal interest. We just have to remember to skip around them to access the data files we're interested in.

If you run CUBS in "L" mode on an unlabeled tape, you'll get exactly the same results as the "A" analyze mode would produce. By selecting L mode, you're asking CUBS to look for labels (80 byte blocks with certain ID words in certain positions). If labels are found, CUBS determines if the tape is written in ASCII or EBCDIC (and tells you), then outputs short, canned label-field-descriptions followed by the contents of the label fields.

A sample follows below:

```
ANALYZING TAPE....

TAPE RECORED IN EBCDIC
TAPE FILE 000:                         LABEL ID: VOL1
VOLUME ID: PEF001                      ACCESSIBILITY:
OWNER:                                 LABEL STANDARD VERSION:

TAPE FILE 000:                         LABEL ID: HDR1
FILE ID: PY.PROD.PEF.TAPE              FILE SET ID: PEF001
FILE SECTION NUMBER: 0001              SEQUENCE NUMBER:0001
GENERATION NUMBER:                     GENERATION VERSION NUMBER:
CREATION DATE (YYDDD):  89025          EXPIRATION DATE (YYDDD):  89055
ACCESS: 0                              BLOCK COUNT: 000000
SYSTEM CODE: IBM OS/VS 370
```

```
TAPE FILE 000:                        LABEL ID: HDR2
RECORD FORMAT: FIXED LENGTH
BLOCK LENGTH: 04050                   RECORD LENGTH: 00135
BUFFER-OFFSET LENGTH:

TAPE FILE 000: 003 BLOCKS, 240 BYTES.
  BLOCK   1; 080 BYTES
  BLOCK 003; 080 BYTES                240 BYTES THIS TAPE.

TAPE FILE 001: 103 BLOCKS, 415,530 BYTES.
  BLOCK   1; 4,050 BYTES
  BLOCK 103; 2,430 BYTES                415,770 BYTES THIS TAPE.

TAPE FILE 002:                        LABEL ID: EOF1
FILE ID: PY.PROD.PEF.TAPE             FILE SET ID: PEF001
FILE SECTION NUMBER: 0001             SEQUENCE NUMBER:0001
GENERATION NUMBER:                   GENERATION VERSION NUMBER:
CREATION DATE (YYDDD):  89025         EXPIRATION DATE (YYDDD):  89055
ACCESS: 0                             BLOCK COUNT: 000103
SYSTEM CODE: IBM OS/VS 370

TAPE FILE 002:                        LABEL ID: EOF2
RECORD FORMAT: FIXED LENGTH           BLOCK LENGTH: 04050
RECORD LENGTH: 00135                  BUFFER-OFFSET LENGTH:

TAPE FILE 002: 002 BLOCKS, 160 BYTES.
  BLOCK   1; 080 BYTES
  BLOCK 002; 080 BYTES                415,930 BYTES THIS TAPE.

  DOUBLE FILEMARKS ENCOUNTERED.  (C)ONTINUE OR (A)BORT? A
```

W mode

CUBS writes a full CUBS compatible test tape of random length blocks, with filemarks at random intervals, from the BOT marker to the EOT marker, outputs a summary line, rewinds the tape and quits.

NOTE: The normally random block size may be fixed by the "L=n" option. The maximum number of blocks may be limited by the "M=n" option. See page 5 for a full description of these options.

The "W" mode writes a tape suitable for reading by the "R" mode. This provides a means of extensively testing tape drive compatibility.

Tape blocks written by the "B" and "W" modes consist of pseudo-random data plus 12 bytes of meaningful data to allow more extensive error checking when the blocks are read by the "R" mode. The file number to which each block belongs, the block number and the block length are placed in 6 bytes at the beginning and at the end of each block.

Modern formatted drives do an automatic read after write test and can therefore detect, and attempt to correct, write errors. Uncorrected write errors and write retries are counted. Hard write errors are reported as they occur. Average write retries are reported on the summary line.

"W" ERRORS REPORTED:

WERR - Hard write error, after 10 write retries.

I/O INC – I/O incomplete. The input / output channel has not transferred all the requested bytes from a block of memory to tape. Most likely indicates a speed incompatibility between the computer's data transfer ability and the need to keep feeding the tape drive at its required rate. You may sometimes get this error when you enter "S" or "CTRL-A" while the tape is running at high speed; in these cases, the keyboard interrupt servicing routine delayed the computer's input / output channel too much.

AVG-WRR - Average write retries. Total write retries/total blocks written.

R mode

CUBS reads a full tape, from the BOT marker to the EOT marker, outputs a summary line, rewinds the tape and quits. For this test to be meaningful, the tape must be a CUBS compatible test tape. Use A mode (see next page) or L mode to read and analyze normal tapes.

"R" ERRORS REPORTED:

BLER (Block Length ERror) - the length reported by the hardware differs from that embedded in the header of the block.

BLOCK NUMBER/FILE NUMBER - the block number and/or file number embedded in the header differs from that expected by the program (CUBS keeps count of blocks and files (filemarks) it reads from tape).

DER (Data ERror) - the first 6 bytes of each block are supposed to be the same as the last 6 bytes. "DER" if not.

HER (Hard ERor) - when the hardware reports an uncorrectable read error that's a Hard ERror.

CER (Corrected ERor) - Total single-bit read errors (dead track errors) corrected by the tape drive electronics.

AVG-RDR - Average read retries. Total read retries/total number of blocks read.

B mode

The B mode is simply an automatic combination of a W, followed by an R. B stands for (B)oth [Both "W" and "R"].

CUBS writes a full CUBS compatible tape, from the BOT marker to the EOT marker, outputs a summary line and rewinds the tape. Then CUBS reads the full CUBS compatible tape just written, from the BOT marker to the EOT marker, outputs a summary line, rewinds the tape and quits.

The "B" mode provides a quick method of writing and reading a full tape with extensive checking.

If the "\" (repeat) option is used at the end of the command line, the "B" mode will run forever until aborted with "CTRL-A". This provides a means of life testing the tape drive subsystem over an extended period of time while accumulating statistics over a number of passes.

Command line options

H - HALT ON ERROR. Causes CUBS to halt on any error. Hit <ESC> to continue.

```
Examples:      CUBS  A  H  <ENTER>          CUBS  R  H  <ENTER>
  CUBS  W  H  <ENTER>
```

C - REPORT CORRECTED READ ERRORS ON A BLOCK-BY-BLOCK BASIS. Modern tape drives can detect a 'dead- track', and using the redundancy provided by the parity bit reconstruct the missing bit. The drive reports this to the computer as a 'CORRECTED ERROR'. CUBS reports these to the user, during A, L, or R modes, in the file summary, as CERs. If you want to have CERs reported as they occur, on a block-by-block basis, use the C option. Note that not all drives report these corrected errors to the host.

L - LOG RESULTS TO A DISK FILE. When "L" alone is input, the log file is named "CUBS.LOG". If "L:filename" is input, the log file is given the name "filename".

P - LOG RESULTS TO THE PRINTER. This is a shorthand equivalent to entering "L:PRN".

Examples: CUBS A L <ENTER> CUBS L P <ENTER>

N - Report block length changes (during A & L modes). Whenever the length of the block just read is different from the length of the previous block (New length), the file number, block number within that file and block length will be reported. This feature provides an easy way of detecting troublesome "variable-length-record" mainframe tapes or tapes with short blocks.

nnK - Limits the maximum block size written by B or W to nnK bytes where nn is a decimal number from 1 to 64. K = 1024 decimal. For example, entering "2K" would limit the maximum block size to 2048 decimal.

NOTE: You should specify 32K (or less) if you are not sure that your drive can handle 65535 byte blocks. If this option is not used, or if "64K" is used, the maximum block size is 64K - 1 (65535) bytes.

Example: CUBS W 9K <ENTER>

U - Unload tape when finished. Tells CUBS to issue an 'unload' command to the drive before exiting to DOS. This causes the drive to perform and unload function and turn off the blower. Of course the drive must be capable of executing an unload command.

Example: CUBS B U <ENTER>

A - Abort at "End-Of-Data" (two adjacent tapemarks). The A and L modes halt on End-Of-Data, and ask you if you want to continue with the analysis or abort. If you choose abort, then the tape is rewound. This option allows you to pre - answer that question so when you can leave the computer and then come back to find the tape rewound or unloaded, as you previously picked on the command line, e.g. CUBS A H A U <ENTER>

\ - Repeat the entire command line. Causes CUBS to go back and re-run things, again and again, until aborted with "CTRL-A", see below.

For example, if the command line is: "CUBS B U <ENTER>"

CUBS will write a full tape, read the full tape, print a summary, unload the tape and return to DOS.

If the command line is, however: "CUBS B U \ <ENTER>"

the "B" operation will be performed over and over again - this provides a simple means of life-testing the hardware. It will run all day until aborted with CTRL-A. Cumulative totals are kept for the entire run and the number of passes is output. You'll get a summary: at the end of each pass, whenever you input the (S)tatus run-time command and when you use CTRL-A to abort the endless run.

Use the (L)og option along with (\) so you don't loose any intermediate results off the top of the screen.

For example: CUBS B L 32K \ <ENTER>

Options to write test tapes with specified format:

L=n - (Write fixed Length blocks of length "n"). This option is valid in B and W modes. It is primarily useful in W mode in writing test tapes of specified structure. If there's an "L=n" on the command line, e.g.

```
CUBS W L=20000 <ENTER>
```

each block written will be "n" bytes long (20,000 bytes in the example above) and no random filemarks will be written. Unless limited by a "M=n" option (described next) a full tape will be written. In any case, the only filemarks will be the two adjacent filemarks written at the end of the string of fixed length data blocks.

M=n - (Limit number of blocks written to a Maximum of "n" blocks). Like the L option above, this option is valid in B and W modes. It is primarily useful in W mode in writing test tapes of specified structure. Normally CUBS continues to write blocks until EOT is reached or you type CTRL-A to abort, then two adjacent filemarks are written. The M option sets a Maximum on the number of blocks to be written. Unless EOT is hit first, you'll get exactly "n" blocks. For example, "M=1000" sets the maximum number of blocks to 1,000. When the specified number of blocks has been written, CUBS writes two adjacent filemarks and rewinds the tape.

F=n - (Fill each block with the bit pattern specified by "n"). This option is valid in W mode only. It is primarily useful in W mode in writing test tapes of specified structure. The decimal number 'n' (0-255) specifies the fixed pattern to be used to fill each block written. When 'F' is in effect CUBS does not stuff the block length, block and file numbers in the header and tail of the block. The entire block is filled with the pattern.

Run-time commands

S - Status display. One or more lines of cumulative status are output to the screen, then CUBS continues. Don't be bothered if using this command occasionally causes an Input / Output error.

CTRL-A - Abort. Gives a status summary, as with "S", then rewinds (or unloads if there's a "U" on the command line) and returns to DOS.

The DITTO Program

FIXED-LENGTH RECORD DISPLAY PROGRAM

ADDITIONAL HARDWARE SUPPORTED: line printer.

PURPOSE: DITTO helps the user to analyze the contents of tape or disk files comprised of fixed-length records. DITTO prints a user specified quantity of sequential records starting anywhere in the input file below a header of "column numbers" which indicate the byte position of each character. The DITTO display is useful in figuring out the field boundaries and data types on "mystery tapes".

FEATURES: DITTO contains logic to create and print up to 9999 column numbers and the associated data records, without overflowing the right margin of the screen or printer. Input may be from tape or disk. Output may be to printer, screen, or disk. EBCDIC to ASCII conversion is supported. In typical use, a bunch of records are displayed below each "header" of column numbers. The user chooses to display the records as character data, in HEX, or both. "Unprintable" characters are displayed as decimal points (for place keepers), not as their screen or printer graphics.

```
SAMPLE DITTO RUN
Type the Logical Record Length (maximum column number) and hit <ENTER>: 146
(T)ape or (D)isk input? T
Enter tape file number: 0
Convert EBCDIC to ASCII? Y
Enter range of records (e.g. 100-150) or <ENTER> to start with first: 201-209
Output as (C)haracters, (H)ex digits, or (B)oth? C
How many records should be displayed per header? 5
Output to (S)creen, (P)rinter, Printer with e(X)tra PrtSc or (D)isk File? P     NOTE:  The first 3 lines below are the column numbers
Type the Maximum number of columns/page (Page Width) and hit <ENTER>: 122       generated by DITTO.  Read them VERTICALLY.

00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111111111111111111
00000000011111111112222222222333333333344444444445555555555666666666677777777778888888888999999999900000000001111111111222
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012

401608949 86279 085A101 ADKINS         GEORGE ALTON        MWR   SUNBEAM TRAILER PARK  LOT 9           42754 0041645011586
401669562 86279 085A101 ADKINS         MARY E              FWD   SUNBEAM TRAILER CT    LOT 9           42754 0022046010176
394589230 88284 085A101 ALEXANDER      MARGIE HELENA       FWR   685 JOHNSON ST.                       42754 0100152010188
401981017 86087 085A101 ALLEN          DANIEL BRYANT       MWR   406 BRANDENBURG RD                    42754 0011768061070
403625119 79124 085A101 ALLEN          FRANCES E           FWR   209 CONKLIN DR                        42754 0041347000077

111111111111111111111111
222222233333333334444444
345678901234567890123456

100486 006007008009000000
012279 004005016007018010
090188 008009000001002000
012686 006017018119000000
040279 004015116007018010


00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111111111111111111
00000000011111111112222222222333333333344444444445555555555666666666677777777778888888888999999999900000000001111111111222
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012

403626900 73176 085A101AALLEN          JAMES MELVIN        MWR   406 BRANDENBURG RD                    42754 0101745101745
407085077 88097 085A101 ALLEN          JAMES TODD          MWR   406 BRANDENBURG RD                    42754 0021470061570
219489830 73222 085A101AALLEN          JUDY                FWR   406 BRANDENBURG RD                    42754 0071145060070
401980743 89038 085A101 ALLEN          THOMAS COLEMAN      MWR   406 BRANDENBURG RD                    42754 0082171082371

111111111111111111111111
222222233333333334444444
345678901234567890123456

032673 004115116117118110
021288 008019000001002000
062573 004015116017118110
020389 009000001002003000
```

How to run ditto

To run DITTO type: DITTO <ENTER> (conversational mode)

or

```
DITTO [?] [OPTIONS] <ENTER>   (command line mode)
```

As indicated in the above example, DITTO is most easily invoked in conversational (question and answer) mode. Alternately, after you gain experience, all the options may be specified on a single command line. Any command line options entered will result in the corresponding questions not being asked in the console dialogue. Conversely, if you forget to enter one or more command line options, DITTO will ask you one or more questions before starting processing.

Typing ? on the command line causes DITTO to display a short "help-list" of all valid command line options.

CONVERSATIONAL MODE CONSOLE DIALOGUE:

Type the Logical Record Length (maximum column number) and hit <ENTER>: Enter the logical record length. This is the highest column number DITTO will print. The column number will be reset to 001 every "n" characters where "n" is the logical record length entered here. If "LRL=n" appears on the command line, DITTO will not ask for the logical record length.

(T)ape or (D)isk input? Enter "T" or "D" to tell DITTO where to get input data.

This question is not asked if "DFI=filename" or "TFI=n" appears on the command line.

Enter input filename: This question is asked if the answer to the (T)ape or (D)isk input question was "D".

Answer with a disk filename, optionally preceded with a drive letter and/or path and optionally followed by a range of records within parenthesis, e.g. SAM.TXT or SAM.TXT(333-444) or D:\DATA\SAM.TXT

Enter tape file number: This question is asked if the answer to the (T)ape or (D)isk input question was "T".

Answer with a tape file number (the first file on tape is file 0), optionally followed by a range of records within parenthesis, e.g.: 0 or 0(345-567)

Convert EBCDIC to ASCII? Enter "Y" to have DITTO convert EBCDIC to ASCII for all "character" output. This question is asked only when input is from tape and neither "E" nor "A" appears on the command line. If you have EBCDIC data on your PC's disk, you'll have to enter "E" on the command line.

NOTE CONCERNING EBCDIC TO ASCII CONVERSIONS If you request EBCDIC to ASCII conversion, and the output type is Both HEX digits and characters, the HEX digits are the original EBCDIC, and the characters are converted to ASCII.

There are three special situations provided for in EBCDIC-ASCII conversion (codes are HEX).

EBCDIC lower case numbers (B0 to B9) are translated to the corresponding ASCII numbers (30 to 39).

There are undefined gaps in the 256 possible 8 bit EBCDIC codes. When an illegal EBCDIC code (e.g. CB) is processed, an ASCII UPARROW (5E) is output.

If a legal EBCDIC code which has no reasonable ASCII equivalent (e.g. 1A) is processed, an ASCII BACKSLASH (5C) is output.

Enter range of records (e.g. 100-150) or <ENTER> to start with first: This question isn't asked if you specified a range within parenthesis next to the disk filename or tape file number. If you didn't, it's a reminder that you don't have to start with the first record.

Type the Total Number of Records to Process (or <ENTER> alone for all): This question is skipped if you've already specified a range of records. If you see this question and you don't want to display the entire file, enter the desired number of records here. Just hit <ENTER> to process the whole file. If "TNR=n" appears on the command line, DITTO will not ask for the total number of records to process.

Output as (C)haracters, (H)ex digits, or (B)oth? Tells DITTO how you want the output to appear.

This question is skipped if "C" or "H" or "B" is specified on the command line.

NOTES ABOUT THE DISPLAY OF "CHARACTER" DATA: "Carriage Return" and "Line Feed" are displayed as the lower case letters "c" and "l" respectively. The printable characters (decimal 33-126, HEX 21-7E) are displayed directly. All other characters are displayed as decimal points, NOT as their screen or printer graphics.

How many records should be displayed per header? The number entered here will set the number of records displayed under each set of column numbers. If you want more than one record be displayed under each header (the usual case), this is where you specify that number. Hit <ENTER> alone to just display one record under each header. The above question is asked only when "TNR=n" does not appear on the command line. See the last page for information on exceptions.

Output to (S)creen, (P)rinter, Printer with e(X)tra PrtSc or (D)isk File? Enter "S" to send output to the screen, "P" to send output to the printer (PRN), "X" to first print the screen (to document the selected options) then send DITTO output to the printer or "D" to send the DITTO output to a disk file. Not asked if "DFO=filename" or "S" or "P" or "X" appear on the command line.

Enter Filename for Disk output: Enter the filename for output to be sent to. Asked if disk file output has been selected.

Type the maximum number of columns/page (Page Width) and hit <ENTER>: Enter the maximum number of columns you want to be printed across a page. During output, whenever this many columns have been printed, CR/LF's are inserted so that the display continues below and in the left margin. Asked only when "PW=n" is not specified on the command line. If the logical record length is less than the page width, the page width is internally set to the logical record length. Defaults to 80 for Screen output. If your screen is wider, use "PW=n" on the command line to override this.

SETTING PAGE WIDTH. After the number of columns specified as the Page Width are printed, a carriage-return/linefeed sequence is sent. Often, specifying one less than the maximum number of columns across a page produces better results. On many printers, if the actual maximum number of columns across a page is sent, the printer will "auto CR/LF" to the next line. Then, when the CR/LF is sent by DITTO, the text appears double-spaced.

DITTO STOPS & QUITS if a specified "Total Number of Records" has been output, or when a specified range of records has been output (whichever is less) or if you've attempted processing past the end of the input file - in which case you'll get a BEEP and the message: "END OF INPUT FILE". (Exception when TNR=0, see TNR=n below).

COMMAND LINE OPTIONS   Optional input is shown in square brackets -> [ ]

```
DFI=filename[(a-b)]   Get Disk File Input from "filename."  (a-b) is an optional range of records. E.g.
DFI=E:\SPOOGE\ABC.TXT(234-567)
DFO=filename     Disk File Output. Send output to disk file "filename". E.g. DFO=C:\SEPT29\FILE-1.DIT
TFI=n[(a-b)]   Tape File Input: Get input from tape file "n".   Note that File 0 designates the first
file on the tape. (a-b) is an optional range of records. e.g.   TFI=1(200-234)
```

LRL=n Set logical Record Length: The Logical Record Length is synonymous with the highest column number. After "n" columns of data are output, the column number is reset to 001 and the display continues with a fresh header.
TNR=n Set the Total Number of records to be output. If TNR=0, the entire file will be processed.
RPH=n Set the number of Records to be displayed Per Header.   See the last page for additional information.
PW=n Set the Page Width to "n" columns.
P - Printer output. Causes output to be sent to "PRN". A form-feed command is sent to the printer when printing is finished.
X - Same as P but the screen will be printed first (to capture the command line options or DITTO screen dialogue).
S - Screen output. Causes output to be sent to the Screen. Note that options P and S may not be used simultaneously.
U - Unload when done. Causes tape drive to rewind and unload tape upon program termination. Normally, tape is only rewound. Works only with tape drives which support this feature.
E - Convert input data from EBCDIC to ASCII. See note under CONSOLE DIALOGUE about EBCDIC to ASCII conversions.
A - Do NOT convert from EBCDIC to ASCII. Use when "TFI" input has been specified to discourage DITTO from explicitly asking if you want to convert EBCDIC to ASCII.
C or H or B Output as Characters or Hex or Both.   Choose only one.

Example of "B" (Both HEX & Character) output

NOTE: Tape data is EBCDIC, field 65-67 is Packed Decimal

```
0000000001111111111222222222233333333334444444444555555555566666666667777
1234567890123456789012345678901234567890123456789012345678901234567890123
DCECD4444444444DCCD4444444FFF4F6F4ECDC4EE44444DDDCCDCD44444444CC557ECFDCE
2925900000000001515000000021201120595502300000465414950000000009115C620818
KISER          JEAN       212 1/2 VINE ST     MONDAMIN        IA^^@WB0QAY

CDDDCDE44444444ECDDCE44444FFFF4FEC4EE4FF444444CDCCDCE444444444CD861DFDCE4
3633952000000003864120000018030938023014000000795535800000000003603C508190
COLLINS        THOMAS     1803 9TH ST 14      GREELEY         CO^^\N0QAZ

ECDDCCDE4444444DCCCCCD4444FFFF4D4DCDD444444444CDCCEEDDC4444444DD618DCFDCE
6933914200000004938153000071170507192000000000731423655000000004641C210819
WILLIAMS       MICHAEL    7117 N PARK         GLADSTONE       MO^.\KA0QAZ

DECDDC444444444DDECD444444FFFF4DCCDCDD4CD44444ECCDE4DDECDC4444DD656EFDC44
627664000000000968130000003506059325330490000021953016257800004640C608200
OSGOOD         ROYAL      3506 NICKELL DR     SAINT JOSEPH    MO^&%W0QB

CDECC4444444444CCDC4444444DDC4FFF4444444444444DCECDEEDDC444444EE214DCFDC4
268350000000000755500000007620671000000000000091555266640000006566C220820
BOYCE          GENE       POB 671             RAVENSWOOD      WV\.<KB0QB

DCCEC4444444444CDCCD444444D4F4CDE4FFFC44444444CD4CDDCCD4EDDCDCDD674DCFDC4
259380000000000134550000009020267026010000000053046914602799574644C230820
KEITH          ALDEN      R 2 BOX 260A        EL DORADO SPRINGMO^^<KC0QB

DCDCEEDD4444444CDDCDC44444FFFF4EE4FFEC44444444EDDCDC444444444DE664DCFDC4
315723650000000465134000001115026010380000000036752100000000002260C250820
LANGSTON       DONALD     1115 SW 10TH        TOPEKA          KS^-<KE0QB

CDECD4444444444CCDDE444444FFF4EDDCD4ECDDCE4444EDCCDCCEC4444444EC573DCFDC4
996950000000000819980000002920477590513358000027519692800000002478C260820
IRWIN          HARRY      292 UPPER VALLEY    SPEARFISH       SD^^.KF0QB
```

SUMMARY OF DITTO COMMAND LINE OPTIONS

This is an example of the list of options which are displayed if you type: DITTO ? <ENTER>

```
NOTE:   Optional input is enclosed within square brackets  ->  [   ]
       TFI=n[(a-b)]    - Tape File Input from file number n (0-999), [records (a-b)]
              LRL=n    - Logical Record Length is n bytes
              TNR=n    - Total Number of Records to process is n
              RPH=n    - number of Records displayed Per Header is n
               PW=n    - set Page Width to n columns
DFI=filename[(a-b)]    - Disk File Input from "filename", [records (a-b)]
       DFO=filename    - Disk File Output to "filename"
                  P    - Printer output
```

```
                X    - Printer output with Xtra PrtSc to document selected options
                S    - Screen output
                U    - Unload tape when done
                E    - Input file is EBCDIC.  Convert to ASCII
                A    - Input file is ASCII.  No conversion required.
C or H or B  Output as Characters or Hex or Both.   Choose only one.
```

ADDITIONAL INFORMATION, ERROR MESSAGES, EXCEPTIONS, HINTS

Records per Header   There are situations where the number of records displayed under a set of column numbers (header) is less than the number requested. Less records will be displayed if DITTO's internal buffer isn't large enough to accommodate a request. Additionally, if input is from tape AND the range of records requested spans tape blocks you'll see all the requested records but DITTO will generate an extra header under which will be less than the requested number of records per header. If this is a problem, use FX to transfer the records to a disk file and "DITTO" them from the disk file.

Note about Rewinding Tape When processing tape files, DITTO rewinds the tape before spacing to the selected file. The reel of tape is also rewound on exit to DOS.

"Blocking Factor not an integer: Block Length was nnn" error message DITTO will abort with the above message after processing the first block when input is from tape and the block length on tape is not an integer multiple of the specified logical record length. A common reason for getting this error message is that you have a labeled tape (the data is in file 1, not file 0) and you erroneously tell DITTO to take input from file 0. Run DITTO again and specify that the data in is tape file 1. If that doesn't help and you really have a tape where records span blocks, use the FX program to transfer the data to disk and run DITTO with the disk file as input. This is a very unusual situation, it's more likely that the record length is different than what you think it is. See the chapter "DEALING WITH MYSTERY TAPES" for hints on figuring out the logical record length.

The D2T Program

TRANSFER .DBF COMPATABLE FILES TO TAPE IN MAINFRAME FRIENDLY FORMAT

PURPOSE: D2T creates MAINFRAME FRIENDLY, fixed-length record tape files directly from the data contained in .DBF files. DBF files are the internal, working format used all versions of by dBASE, FoxBASE, FoxPro, Clipper, workalikes and compatibles. This means you don't need any free space on your disk! Without this program, you'd need free space the size of the .DBF file.

FEATURES: Tape data may be written in ASCII or EBCDIC. Numeric fields may be written in ASCII, EBCDIC, Packed Decimal and/or Zoned Decimal. D2T prints detailed Record Layout and File Structure documentation for you to send with the tape. You may specify any blocking factor that results in blocks smaller than 64K. Large .DBF files may be transferred to any number of tapes. D2T writes industry-standard unlabeled tapes. Numeric and date fields are automatically converted to common mainframe friendly formats. Experienced users may run D2T entirely from the command line. Alternately, D2T will ask questions in a user dialogue or any mixture of command line control and dialogue may be chosen. While running, D2T displays job statistics - see the example below.

SAMPLE D2T RUN   (conversational example)

CONSOLE DIALOGUE:

```
D2T <ENTER>
Name of .DBF File to Transfer to Tape: E:\VH\VH
Enter tape file number or 'EOD' for End Of Data [first tape file is 0]: 0
Convert ASCII to EBCDIC ? (Y/N) Y
Logical Record Length on Tape Will Be: 38
Enter Tape Blocking Factor [1-1680]: 125
(P)rint, (V)iew, (L)og or (N)o Record Layout Report?  P
Automatic Rewind ? (Y)es, (N)o, (U)nload   U
```

SAMPLE D2T RUN (command line example)

The following example shows how, with a single command line, to perform the exact same job as in the conversational mode example above. The order of the command line options doesn't matter.

COMMAND LINE ENTRY:

```
D2T   DFI=E:\VH\VH   TFO=0   E   BF=125  P  U  <ENTER>
```

SUBSEQUENT SCREEN DISPLAY WHILE D2T IS RUNNING:

```
         File:  E:\VH\VH.DBF
Tape file number:       0
Record    Length:      34
Blocking  Factor:     125
Block       Size:    4250
Writing Blocks:       15 - 28                  (block length = 4250 bytes)
Blocks Written:       28                       (119000 total bytes)
Records Processed     3500 of 57573            (0 skipped)    6 % complete
```

TO RUN D2T TYPE:

D2T <ENTER> (conversational mode)

or

D2T [?] [OPTIONS] <ENTER> (command line mode)

As indicated in the above example, D2T is most easily invoked in conversational (question and answer) mode. Alternately, after you gain experience, all the options may be specified on a single command line. Any command line options entered will result in the corresponding questions not being asked in the console dialogue. Conversely, if you forget to enter one or more required command line options, D2T will ask you one or more questions before processing the input file.

Typing ? on the command line causes D2T to display a list of all valid command line options.

CONVERSATIONAL MODE CONSOLE DIALOGUE:

Name of .DBF File to Transfer to Tape:

Simply input the name of the .DBF file and hit <ENTER>. The filename extension ".DBF" is assumed and may be omitted. The name of the .DBF file may be preceded by a pathname as required. This question is skipped if the filename is specified on the command line with the "DFI=" (Disk File Input) option.

Enter tape file number or 'EOD' for End-Of-Data [first file on tape is 0]:

Simply input the tape file number and hit <ENTER>. If you specify any number greater than 0, D2T will space past that many file marks before writing the disk file to tape. If this is a virgin tape you must answer 0. Enter the characters "EOD" (End Of Data) to cause D2T to add the tape file at the end of a string of one or more previously written files. In this case, D2T will tell you which tape file number it is writing. End-Of-Data (a standard convention) is indicated by two adjacent filemarks at the end of the last good file on a tape. All of Electrovalue's programs that write tape terminate the tape with a double filemark. The tape file number question is skipped if the command line option "TFO=" (Tape File Output) is used.

Convert ASCII to EBCDIC (Y/N) ?

Enter "Y" to have D2T convert the .DBF data to EBCDIC. This question is not asked if "A" (ASCII) or "E" (EBCDIC) is specified as a command line option.

NOTE: There are several ASCII characters that have no EBCDIC equivalent. They are not likely to be found in your .DBF file. If encountered, they are translated into an EBCDIC BACKSLASH (224, E0). These characters are:

ENQ (5,05) DC3 (19,13) GS (29, 1D) US (31, 1F) `(96, 60) | (124,7C)

Logical Record Length on Tape Will Be: nnnn

Enter Tape Blocking Factor (number of logical records per block) [1-bbb]:

D2T analyzes the .DBF file and displays the logical record length it will use when writing to tape. The logical record length will be the same as the sum of the field widths in your .DBF file unless it contains NUMERIC fields with decimal points or you are writing Packed Decimal fields. At any rate, the record layout report that is produced will fully document each field. Further information on field lengths and the record layout report is presented on the following pages.

D2T then displays a range of possible blocking factors. Tape data is written in physical blocks. The 'Blocking Factor' is the number of logical records per physical block on tape. characters. Minicomputers and mainframes generally do not welcome blocks larger than 32,767 bytes. If you don't know what blocking factor to use, discuss it with the technical personnel who will be processing the tape. The blocking factor question is skipped if the blocking factor is specified on the command line with the "BF=" option.

(P)rint, (V)iew, (L)og or (N)o Record Layout Report? Answer See full discussion below.

Automatic Rewind ? (Y)es, (N)o, (U)nload? Tell D2T what to do after the file has been transferred to tape.

PROCESSING As soon as the "Automatic Rewind" question is answered, D2T starts the disk to tape process. When the file has been completely transferred, D2T returns to DOS.

DELETED RECORDS Records in the .DBF file that are marked as DELETED are by default not written to tape. The number of deleted records that were skipped is displayed on the screen and shown in the record layout report. If you want to write DELETED records to tape, use the command-line-only option "IDR (Include Deleted Records). If the IDR option is chosen, the number of included DELETED records is displayed on the screen and shown in the record layout report.

FORCING UPPER CASE Some older mainframes can not handle lower case characters. If your .DBF file contains lower case data that the mainframe finds offensive, use the command-line-only option FUC (Force Upper Case). Any lower case data in the .DBF file will be written to tape as upper case.

ABORTING To abort a transfer in progress, hit CTRL-A. D2T will write two adjacent filemarks, and return to DOS. The amount of data transferred to tape before the abort is shown on the screen and in the record layout report.

PHYSICAL PARAMETERS: The default physical parameters (Host adapter card, Tape drive SCSI ID, and Write density) used by D2T are displayed on the top line of the screen when the program is first run. They may be changed by using the TACO program. See the INSTALLATION chapter of this manual.

END-OF-TAPE: Large disk files may be transferred to any number of tapes. If the end-of-tape reflective marker is hit, D2T closes the current tape by writing two adjacent filemarks, then rewinds and unloads the tape, and outputs the following message:

End of Tape Encountered. Do you wish to mount another tape and continue the transfer? (Y/N)

Mount another tape and input Y for yes to continue.

SUMMARY OF D2T COMMAND LINE OPTIONS

This is an example of the list of options which are displayed if you type: D2T ? <ENTER>

```
DFI=filename       - Name of DBF file to write to tape
   TFO=n|EOD       - Tape file n (0 is first file) or EOD for End-Of-Data.
        BF=n       - Blocking factor is n logical records per physical block.
           A       - Write tape in ASCII.  No conversion.
           E       - Write tape in EBCDIC.
           F       - Write tape at high velocity.
          NR       - Don't rewind tape at beginning or end of run.
           R       - Rewind tape at end of run.
           U       - Unload tape at end of run.
         FUC       - Force Upper Case when writing character fields to tape.
         YMD       - Set date format to YYMMDD (default is MMDDYY).
         IDR       - Include Deleted Records (default is to skip Deleted records).
       PC=xy       - Redefine Packed Decimal +/- characters, "x" for +, "y" for -.
       ZC=xy       - Redefine Zoned Decimal +/- characters, "x" for +, "y" for -.
           P       - Print record layout information at end of run.
          PO       - Print Only.  Don't transfer data to tape.
 LF=filename       - Log record layout report to filename.
LFO=filename       - Log record layout to filename only, don't write data to tape.
           V       - View record layout information at end of run.
          VO       - View Only.  Don't transfer.
           ?       - Print this message.
```

DOCUMENTING THE TAPE'S CODE, RECORD LAYOUT and FILE STRUCTURE

The options discussed here serve the purpose of fully documenting the contents of a magnetic tape produced by D2T.

In conversational mode, D2T asks: (P)rint, (V)iew, (L)og or (N)o Record Layout Report ? Select P to route the report to your PC's printer. Select V to see it on the screen. Select L to Log the report to a disk file (useful on networks where the printer isn't local to your PC or when you want to include a copy of the layout in a report). In this case you'll be asked the name of the disk file to which D2T is to write the report. Select N for No report.

You should select P or L to create printed documentation to accompany any transmittal of the tape. If you select L, you may import the log file into your word processor and add or modify it in any way you like.

The F command, Write tape at high velocity, is only applicable if you have a dual speed drive such as a Cipher F880.

Note that the field lengths on tape will differ from those in the .DBF structure for numeric fields with decimal points and all Packed Decimal fields.

Command Line Equivalents: You can select the destination of the report using command line options. Note that to suppress the report and prevent D2T from asking you if you want one, you can use "LF=NUL" to route it to DOS's NUL "NON-device".

P (Print) causes D2T to print a record layout after completing the disk to tape transfer.

PO (Print Only) causes D2T to print a record layout without performing an actual data transfer to tape.

V (View) causes D2T to display a record layout (on the screen) after completing the disk to tape transfer.

VO (View Only) causes D2T to display a record layout without performing an actual data transfer to tape.

LF=filename causes D2T to Log the report to disk file "filename".

LFO=filename causes D2T to Log the report to disk file "filename" without writing to tape.

EXAMPLE: D2T DFI=DECDATE TFO=0 BF=50 E P U <ENTER>

NOTE: THE .DBF FILE CONTAINS 5 "DELETED" RECORDS

This .DBF Structure: (as listed by dBASE's "LIST STRUCTURE" command)

```
Structure for database: DECDATE.DBF
Number of data records:    1047
Date of last update   : 04/28/99
Field  Field Name  Type       Width    Dec
    1  BLDG_DESC   Character     15
    2  LAND_DESC   Character     20
    3  ACERAGE     Numeric       10      4
    4  ZONING      Character      4
    5  NO_ONRS     Character      2
    6  DEED_DATE   Character      6
    7  SALE_AMT    Numeric        9
    8  LAND_VAL    Numeric        9
    9  IMP_VAL     Numeric        9
   10  LAST_YR_TX  Numeric       10      2
   11  THIS_YR_TX  Numeric       10      2
** Total **                     105


Corresponds to this Record Layout Report:

Magnetic Tape Information for DECDATE:

Tape written in EBCDIC
Tape file number                                       =      0
Logical record length                                  =    101
```

```
Blocking factor                                         =     50
Tape block length                                       =   5050
Number of blocks written                                =     21
Number of records transferred                           =   1042
Number of bytes transferred                             = 105242

     5 "Deleted" records in original disk file skipped.

BLDG_DESC    001-015   L=015   Character.
LAND_DESC    016-035   L=020   Character.
ACERAGE      036-044   L=009   Numeric.   Assume 4 decimal places
ZONING       045-048   L=004   Character.
NO_ONRS      049-050   L=002   Character.
DEED_DATE    051-056   L=006   Character.
SALE_AMT     057-065   L=009   Numeric.
LAND_VAL     066-074   L=009   Numeric.
IMP_VAL      075-083   L=009   Numeric.
LAST_YR_TX   084-092   L=009   Numeric.   Assume 2 decimal places
THIS_YR_TX   093-101   L=009   Numeric.   Assume 2 decimal places
```

NOTES ON DATA TYPES

.DBF Numeric Fields: Internally (from left-to-right), .DBF numeric fields consist of an optional minus sign ("-"), the digits to the left of the decimal point, the decimal point itself and the digits to the right of the decimal point. The maximum number of positive digits and the decimal point add up to the Field Width shown by your database program's 'DISPLAY STRUCTURE' command. Here's what D2T does to .DBF numeric data:

Decimal points are discarded. This means that numeric fields that contain a decimal point on disk occupy one less byte on tape. For example, if the field width for a field containing a decimal point in the dbf file is 7, D2T will write the field with a width of 6. D2T's Record Layout Report tells the tape recipient where to re-insert the decimal point.

D2T left fills numeric fields with 0's.

Example:

The Numeric Field " 1234.56" (Field Width 8) in a .DBF file is written to tape as "0123456" (Field Width 7). D2T's layout report tells the recipient where to insert the decimal.

Negative numeric values are represented with a minus sign ("-") in the leftmost position.

The Numeric Field "   -123.45" (Field Width 9) in a .DBF file is written to tape as "-0012345" (Field Width 8). Again, D2T's layout report tells the recipient where to insert the decimal.

.DBF Date Fields: .DBF date fields are 8 characters long. When displayed (e.g. by the dBASE 'EDIT' and BROWSE commands), two / (slash) characters are used as separators, e.g. 11/25/89.

Internally, in the .DBF file, there are no slashes - instead the most significant digits of the year are present. For example, the 8-character .DBF date (as shown by the dBASE 'EDIT' command) 11/22/88 is represented internally as the 8 characters 19881122.

Year 2000 concerns caused us to update our date output format. Mainframe tapes commonly represented dates as MMDDYY, sometimes as YYMMDD. Most mainframe users changed date formats as we approached the new millennium. To make our software year 2000 compliant, our date format was revised to output a four digit year.

The default is for D2T to write .DBF date fields to tape in MMDDYYYY format. Use the command-line-only option YMD to switch this to YYYYMMDD when necessary. The Record Layout Report generated by D2T documents the format that has been used.

Thus, if the default date format is in force, 11/22/98 becomes 11221998 on tape.

.DBF Logical Fields are one byte long and contain either the character T (for True) or F (for False). It's not likely that you'll want to write logical fields to tape but if you do, you'll get one of the above characters.

Packed decimal and zoned decimal numeric fields

D2T can (optionally) write .DBF numeric fields as Packed Decimal and Zoned Decimal fields on tape. The rest of the record can be either EBCDIC or ASCII.

To write Packed or Zoned Decimal, the field type in the .DBF file must be NUMERIC.

PACKED: To write a numeric field in Packed Decimal format the Field Name in your .DBF file must start with the two characters "P_".

ZONED: To write a numeric field in Zoned Decimal format the Field Name in your .DBF file must start with the two characters "Z_".

How the .DBF data is processed:

Decimal points are discarded if present, but their position is noted in the record layout report.

Minus signs are discarded if present, but properly represented in the Packed or Zoned tape field's low-order byte.

Fields are left-filled with ZEROs.

The resulting data is translated to Packed or Zoned format and written to tape.

That's all there is to it. The record layout report identifies Packed and Zoned fields as such.

NOTES ABOUT THE SIZE OF FIELDS WRITTEN

The record layout report displays the length of Packed and Zoned fields on tape.

Zoned: Zoned fields on tape are the same length as the field width in the .DBF file for integer fields; one less than the .DBF width if the .DBF field has a decimal position.

Packed: The internal structure of .DBF numeric fields is less than elegant. Negative numbers must carry a minus sign but positive numbers are unsigned. Thus a .DBF numeric field of any given width can contain a positive value ten times greater in magnitude than the largest negative number. Additionally, packed fields on tape must always represent an even number of unpacked digits.

These rules result in the following formula for calculating the size of Packed fields on tape from the width of .DBF fields (as shown by the data base language's DISPLAY STRUCTURE command):

P = INT [D/2] + 1   Where P is the size of the Packed field in bytes and D is the width of the .DBF field after any decimal point is discarded.

Some Examples:

	DBF FIELD WIDTH	DECIMAL DBF CONTENTS	HEX TAPE CONTENTS	TAPE FIELD WIDTH
PACKED	4	1234	01 23 4C	3
	4	-987	00 98 7D	3
	5	123	00 12 3C	3
	5	12345	12 34 5C	3
	5	-9876	09 87 6D	3
	6	123456	01 23 45 6C	4
	6	-98765	00 98 76 5D	4
ZONED	4	1234	F1 F2 F3 C4	4
	4	-987	F0 F9 F8 D7	4
	5	123	F0 F0 F1 F2 C3	5
	5	12345	F1 F2 F3 F4 C5	5
	5	-9876	F0 F9 F8 F7 D6	5
	6	123456	F1 F2 F3 F4 F5 C6	6
	6	-98765	F0 F9 F8 F7 F6 D5	6

```
CHANGING THE SPECIAL CHARACTERS
The default HEX characters used on tape are "C"
for + and "D" for -.  You can change both of
them with the command line options "PC=xy" and
"ZC=xy" if necessary where x is the substitute
character for positive numbers and y is the sub-
stitute character for negative numbers. For ex-
example to use A and B HEX respectively for
Zoned + and -, execute D2T as follows:
D2T ZC=AB    [Other options] ... <ENTER>.
```

FIELD WIDTHS CHANGE DURING TAPE-TO-DISK TOO: Note that when translating Packed and Zoned from tape to disk the FXED program reserves an extra byte for the sign. Thus, if you use D2T to write Packed or Zoned data on tape, you can not bring it back into the same .DBF file without first modifying the structure to increase the DBF field width.

See the last page of the FXED chapter for additional information on Packed and Zoned decimal data.

Writing multiple files to tape

To use a batch file to automatically write multiple disk files to tape you should make use of the NR and EOD command line options. When D2T is called, it will rewind the tape if the tape drive is online. If you are writing multiple files to tape this behavior can be unwanted. The NR (No Rewind) option should be used in this case.

For example, a batch file containing:

```
d2t DFI=A1  BF=100 E  V
d2t DFI=B1  BF=100 E  V NR          (there is an implied TFO=EOD here)
d2t DFI=C1  BF=100 E  V NR          (there is an implied TFO=EOD here)
d2t DFI=D1  BF=100 E  V NR          (there is an implied TFO=EOD here)
d2t DFI=E1  BF=100 E  V NR U        (there is an implied TFO=EOD here)
```

will write out 5 tape files. You can not use file numbers after the first line if you are using NR, D2T doesn't know where the tape is positioned if you tell it not to rewind. However, D2T does know how to successfully find the End-Of-Data location (2 adjacent file marks at the end of all the recently written data). Note that the last line uses both NR and R. Because the R in on the command line, it overrides the NR so as to rewind the tape to the beginning at the end of the job. You could have used U to unload the tape instead. If you send someone a stacked tape such as this, it would be a good idea to have D2T print file layout reports for each file and have CUBS A print a log report for the entire tape.

The NR option is to be used only to position a data file at the end of data. It should not be used on a blank tape. If it is used on a blank tape, D2T will search forward looking for 2 filemarks. It will not find them because the tape is blank.

NR can be used from command line mode as when D2T is called from another program or a batch file. NR can also be used in conversational mode. You must start D2T by typing in “D2T NR” from the command prompt if you are using NR in conversational mode. If you don't type in NR at the same time as D2T is invoked, D2T will automatically rewind the tape.

The NR option is used to automatically put the file you are writing out at the end of data. You can not specify a file number when you use NR. If you specify a file number at the same time you use NR the file number is ignored. Using this command allows you to put multiple files on tape without rewinding after each file is written.

However, you may use R or U to automatically rewind the tape or unload it after your file is written. If you answer R or U these commands will override the NR at the end of the job.

Dealing With Mystery Tapes

It shouldn't happen, but it's common that you'll receive "mystery" tapes. A mystery tape is one for which the paperwork is missing, or worse, one with erroneous paperwork. You have to accurately know a number of parameters before you can successfully import typical mainframe data.

Most of this chapter assumes you are dealing with data in a fixed record length format.

Before you run your tape to disk job, spend a few minutes determining or confirming these important parameters with electrovalue's programs. Six important parameters are listed below along with the names of the Electrovalue utility programs you can use to determine them.

1. WHICH TAPE FILE IS THE DATA IN? (CUBS' "A" and "L" modes, KEYS "R" & "C" commands).

2. THE LENGTH OF EACH DATA FILE IN BYTES (CUBS' "A" and "L" modes).

3. LOGICAL RECORD LENGTH: KEYS "R", "C", "A", "E", "F5", and "F8". LRL, after running CUBS' "A" or "L" modes. CUBS "L" mode.

4. FIELD DEFINITIONS WITHIN EACH LOGICAL RECORD (DITTO & KEYS).

5. DENSITY (KEYS' "R" command, "C" command).

6. CODE [e.g. ASCII, EBCDIC, PACKED/ZONED DECIMAL] (KEYS' "R", "D", "A", "E" "N", "C", and "CTRL-N" commands); (CUBS "L" mode, see pg. -M2a-).

NOTE: KEYS, CUBS and DITTO can do a lot more than the few features that are described below. Each is fully documented in its own chapter, read about them in detail sometime.

CAUTION

CUBS AND KEYS CAN EASILY WRITE ON TAPE IF YOU HIT THE WRONG KEY.
WRITE PROTECT THE TAPE FIRST BY REMOVING THE WRITE RING OR PUSHING THE THUMBWHEEL TO THE LOCKED POSITION.

Which tape file is the data in?

Unlike disk files, tape files aren't listed in a directory. A tape file is a "physical" thing, it's 0 or more bytes of data followed by a hardware detectable thing called either an "End-Of-File" mark, EOF mark, filemark (Electrovalue's term of choice) or a "tape mark" (term sometimes used by mainframe programmers). By convention, the first file on tape is file 0. Subsequent files are just counted sequentially as filemarks are encountered.

Before you process the tape data in any way, you need to know which tape file contains the data. It's about a 50-50 bet whether it's going to be in file 0 or file 1.

Sometimes, you will encounter circumstances where there are multiple data files on a tape. These are sometimes called 'stacked files'. In this circumstance you'll find CUBS to be an invaluable tool.

You can use the KEYS program's "R" command several times in a row to read the first 4 or 5 blocks and display their sizes. Several (usually only three) 80 byte records in file 0 are the hallmark of a labeled tape. This means that the data is in file 1. You can use KEYS ‘C’ command to do a brief check on the tape label and display whatever relevant information there exists.

However, it's likely you'll also want to learn how many bytes of data are in each file and you’ll want help determining the logical record length you'll be dealing with, so instead of KEYS, you can use the CUBS program, as described below, to kill 3 birds with one stone.

How many bytes in a tape file?

When you're going to import data, it's good to be sure you have sufficient space on your hard disk. You can use the CUBS program to quickly determine the number of bytes in each tape file. On a typical 9 track drive running at 125 inches per second, CUBS can analyze a full 2400-foot, 1600 BPI reel of tape in less than 4 minutes. The hp 88780/7980 drives can analyze a reel in approximately 3.8 minutes.

For example: CUBS L P A H <ENTER> will analyze a full tape (displaying Label info if any), output the results to the Printer, Automatically end the analysis and rewind when two adjacent filemarks (conventionally used to indicate "End-Of-Data") are encountered. In this example, we've also told CUBS to Halt on any unrecoverable read error.

Finding the logical record length

Mainframe tapes usually consist of fixed-length logical records, end-to-end within each block of data on tape, without delimiters.

One of the things you usually want to do to make such data "PC-FRIENDLY" is to import the data record-for-record into a database program or add CR/LF (Carriage Return/Line Feed) at the end of each fixed-length record.

You can import data directly into a .DBF file or add the CR/LF (don't do both) with the FX program during the tape-to-disk transfer, but in either case, you need to know the length of each fixed-length logical record in the tape file you're importing.

The LRL program

The LRL program suggests possible logical record sizes based on the assumption that the physical block lengths used within a tape file are integer multiples of the logical record length. That is to say that each physical block of data consists of one or more logical records and that no logical records span from the end of one physical block to the beginning of the next. This is generally the case with mainframe tapes.

In the interests of tape use efficiency, the programmer writing a mainframe tape normally appends an integer number of logical records into each tape block. The number of records per physical block is called the BLOCKING FACTOR. For example, if the logical record length is 131 bytes, the physical tape blocks might be written as 1310 bytes (10:1 blocking factor), or 13,100 bytes (100:1 blocking factor). All the blocks in a tape file are usually the same length, with the frequent exception that the last block contains only the last "N" records from the disk file. Usually the last block is written as such a "short" block; less frequently it is padded out to full size with SPACE, NUL or ^^^^^^ characters.

When you tell the LRL program the size of the first and last blocks of a tape file, it presents you with a list of all possible logical record lengths.

Use the CUBS program's A or L modes to quickly determine the size of the first and last blocks in each file on any tape. For example: CUBS A <ENTER>. See the CUBS chapter for further information.

The numbers LRL outputs are those numbers which divide evenly (no fractional result) into both the first and the last physical block lengths. All possible blocking factors from 1:1 through 10,000:1 are considered. When the last block of a tape file is smaller than the first, the number of choices will be small (LRL has more information on which to base its guesses) - if the first and last blocks are the same size, LRL will present more possibilities.

Usage: LRL [x] [y] <ENTER>

The arguments x and y are optional. If you forget to enter them, LRL will ask for them. They are the sizes of the first and last blocks of the tape file, in either order. If only one number is entered, LRL assumes that the first and last block sizes are the same.

```
Sample run:
LRL 14300 11011 <ENTER>
FIRST BLOCK IS 14300 BYTES LONG
LAST BLOCK IS 11011 BYTES LONG
Trying blocking factors from 1 through 1200
POSSIBLE LOGICAL RECORD SIZES GREATER THAN 12 ARE:
 143  13
```

Frequently the tapes you receive will be mainframe "labeled" tapes. Each data file in a labeled tape is preceded and followed by small "label" files. A typical labeled tape has labels in files 0 and 2, the data is in file 1. Most of the time, but not always, the RECORD LENGTH is provided in one of the fields in the first label file [file 0]. Repeated here, for easy reference are some good words about this from the CUBS chapter.

CUBS L mode

"L" mode is a special variation of the "A" analyze mode which displays additional information gleaned from the "LABEL" files on so-called "LABELED" tapes. The label files contain only a few informative fields, CUBS adds the industry standard field names and outputs the information in an easy-to-read format.

Remember to skip around the label files when using FX to import the data file you're interested in.

If you run CUBS in "L" mode on an unlabeled tape, you'll get exactly the same results as the "A" analyze mode would produce. By selecting L mode, you're asking CUBS to look for labels (80 byte blocks with certain ID words in certain positions). If labels are found, CUBS determines if the tape is written in ASCII or EBCDIC (and tells you), then outputs short, canned label-field-descriptions followed by the contents of the label fields.

A sample follows. The information that is most interesting is in BOLD.

Mount the tape, put the drive online and input: "CUBS L [options] <ENTER>"

```
ANALYZING TAPE....

TAPE RECORDED IN EBCDIC
TAPE FILE 000:                         LABEL ID: VOL1
VOLUME ID: PEF001                      ACCESSIBILITY:
OWNER:                                 LABEL STANDARD VERSION:

TAPE FILE 000:                          LABEL ID: HDR1
FILE ID: PY.PROD.PEF.TAPE               FILE SET ID: PEF001
FILE SECTION NUMBER: 0001               SEQUENCE NUMBER:0001
GENERATION NUMBER:                      GENERATION VERSION NUMBER:
CREATION DATE (YYDDD):  97025           EXPIRATION DATE (YYDDD):  98055
ACCESS: 0                               BLOCK COUNT: 000000
SYSTEM CODE: IBM OS/VS 370

TAPE FILE 000:                          LABEL ID: HDR2
RECORD FORMAT: FIXED LENGTH
```

```
BLOCK LENGTH: 04050                     RECORD LENGTH: 00135
BUFFER-OFFSET LENGTH:

TAPE FILE 000: 003 BLOCKS, 240 BYTES.
  BLOCK   1; 080 BYTES
  BLOCK 003; 080 BYTES                 240 BYTES THIS TAPE.

TAPE FILE 001: 103 BLOCKS, 415,530 BYTES.
  BLOCK   1; 4,050 BYTES
  BLOCK 103; 2,430 BYTES                 415,770 BYTES THIS TAPE.
```

Figuring out the field definitions

Each mainframe tape you receive for processing should be accompanied by paperwork containing the field definitions for fixed-length record data. Sometimes, people will send you tapes with no documentation. Occasionally, the documentation will be wrong. Other times the documentation gets lost.

The DITTO program.

Mainframes don't use field separators. Thus you have a puzzle on your hands if the field definition paperwork is missing. Records containing undocumented, adjacent numerical fields are just about hopeless but alphanumeric fields containing data such as names, addresses and phone numbers are easily scoped out with help of the DITTO program. Once you find out which file the data is in, you can read in a few blocks with FX to your hard disk and can leisurely use DITTO to print out a report.

PURPOSE: DITTO helps the user to analyze the contents of files comprised of fixed-length records. DITTO displays the contents of one or more sequential records along with "column numbers" which indicate the byte position of each character.

FEATURES: DITTO contains logic to create and print up to 9999 column numbers and the associated data records, without overflowing the right margin of the screen or printer. Input may be from tape or disk. Output may be to printer, screen, or disk. EBCDIC to ASCII conversion is supported. The data may be displayed, below the column numbers, as printed characters, in HEX, or both.

SAMPLE DITTO DIALOGUE AND PRINTOUT:

```
Type the Logical Record Length (maximum column number) and hit <ENTER>: 146
(T)ape or (D)isk input? T
Enter tape file number: 0
Convert EBCDIC to ASCII? Y
Enter range of records (e.g. 100-150) or <ENTER> to start with first: 1-9
Output as (C)haracters, (H)ex digits, or (B)oth? C
How many records should be displayed per header ? 9
Output to (S)creen, (P)rinter, Printer with e(X)tra PrtSc or (D)isk File? P
Type the Maximum number of columns/page (Page Width) and hit <ENTER>: 122
```

```
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011111111111111111111111
00000000011111111112222222222333333333344444444445555555555666666666677777777778888888888999999999900000000001111111111222
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012

401608949 86279 085A101 ADKINS         GEORGE ALTON        MWR   SUNBEAM TRAILER PARK  LOT 9           42754 0041645011586
401669562 86279 085A101 ADKINS         MARY E              FWD   SUNBEAM TRAILER CT    LOT 9           42754 0022046010176
394589230 88284 085A101 ALEXANDER      MARGIE HELENA       FWR   685 JOHNSON ST.                       42754 0100152010188
401981017 86087 085A101 ALLEN          DANIEL BRYANT       MWR   406 BRANDENBURG RD                    42754 0011768061070
403625119 79124 085A101 ALLEN          FRANCES E           FWR   209 CONKLIN DR                        42754 0041347000077
403626900 73176 085A101AALLEN          JAMES MELVIN        MWR   406 BRANDENBURG RD                    42754 0101745101745
407085077 88097 085A101 ALLEN          JAMES TODD          MWR   406 BRANDENBURG RD                    42754 0021470061570
219489830 73222 085A101AALLEN          JUDY                FWR   406 BRANDENBURG RD                    42754 0071145060070


111111111111111111111111
222222233333333334444444
345678901234567890123456

100486 00600700800900000
012279 00400501600701801
090188 00800900000100200
012686 00601701811900000
040279 00401511600701801
032673 00411511611711811
021288 00801900000100200
062573 00401511601711811
```

Wrong density?

Imagine this scenario:

(a) You're getting ‘Hard Errors’, Unrecoverable Read Errors or HER’s from each block when reading a new tape.

(b) Previously read tapes can still be read without problem.

Item (b) rules out dirty heads and other hardware problems.

Maybe, even though you asked for a particular density they gave you a different density tape, one that your drive can not read.

If the tape you're trying to read is the right density for your tape drive, chances are almost 100% that you'll be able to read it without problem. Sometimes there are problems with old tapes or tapes written out of skew but this is rarely seen with modern drives.

You can use the KEYS program to test for characteristics that are indicative of a wrong-density tape.

The following scenario is strongly suggestive of wrong density:

a) You can read known good tapes without error. This rules out other problems such as dirty heads or hardware configuration problems.

b) Using KEYS, enter "5B" to go forward 5 blocks (skipping past any possible label file).

Now enter a series of "R" commands in an attempt to read a bunch of successive, equal-length data blocks. If it's a "wrong-density" tape, you'll get "Hard Error" (HER) on each block and the block lengths displayed will vary by 10% or 20% but sort of hover around some value.

If you try to display the data using "D", neither the ASCII nor EBCDIC will make any sense.

Another scenario suggestive of a wrong density tape or a backup tape is as follows:

You can read in the label information fine, but you get hard errors every time you try to read in a data block. This can happen when you are sent a tape in a backup data format from a mainframe system. The data in the label can be read, but the blocks may be too large or may be compressed with a software program.

What code (EBCDIC, packed or Zoned Decimal) ?

You probably want all the data you're importing to end up as ASCII on your PC. Use the KEYS program's "R" command to read the first block of the data file on tape [if it is a "labeled" tape, first type "1F" to skip past the label file]. Then display that data, side-by-side, in DECIMAL, ASCII and EBCDIC converted to ASCII, by hitting "D" or "-D". The data type can be identified by the heading under the section of data that you can read. You can fit more data on the screen by re-displaying the data with the "E" command (if EBCDIC) or with the "A" command (if ASCII) . The numbers and the other code will not be displayed with the E and A commands. Other ways to determine code, if the tape is a labeled tape, is to use the CUBS program's "L" mode or KEYS "C" command

If the data is pure EBCDIC, select EBCDIC to ASCII translation when importing the data with FX.

KEYS displays "un-printable" characters as decimal points. If your data contains PACKED DECIMAL or ZONED DECIMAL fields you'll see bunches of decimal points where (in context) they don't make any sense.

CAUTION - KEYS can easily write on tape if you hit the wrong key. WRITE PROTECT THE TAPE FIRST by REMOVING THE WRITE RING or PUSHING THE THUMBWHEEL TO THE LOCKED POSITION.

The procedure for dealing with packed or zoned decimal is to use the FXED program to perform a mix of different types of translation, as necessary, on a field-by-field basis. To do that, you have to first identify the code type. To identify these various codes, display the data in HEX either with the KEYS program (hit CTRL-N), then "D" or "N", or with the DITTO program. Compare the bytes corresponding to the unprintable characters to the examples of various codes on the last page of the FXED chapter. Then define the Packed or Zoned fields as such during the FXED program's "Translation Definition" phase.

Basic Information: Tape Data Formats

PURPOSE: The purpose of this document is to introduce and/or review the format of data on industry standard 9 track magnetic tapes and cartridges. This information should be of interest to most users of the tape drive subsystem. Cartridge in this manual refers to 3480, 3490, and 3490e tape cartridges.

No need to erase nor format a 9 track tape or a cartridge before you use it.

Unlike disks and cassette backup devices, you neither format nor erase a 9-track tape or a cartridge before you write on it. 9-track tape and cartridges are like audiocassettes or videocassettes at home: you just write on the virgin tape or over-write the data on a previously used tape.

If you want to erase an entire tape quickly, for security (secrecy) purposes, see the description of the CUBS program's E command in the CUBS chapter or the KEYS program's Z command in the KEYS chapter

DATA FORMAT ON TAPE

"9-TRACK" EXPLAINED:

Data is recorded on tape in 9-bit "characters". Each 9-bit character consists of an 8-bit byte of memory data plus a 9th parity bit that the hardware uses for error detection/error correction. The parity bit is automatically added by the hardware while writing tape and automatically tested (and removed) by the hardware when reading tape. Furthermore, the beginning of each data block contains a preamble. The preambles for PE, GCR and NRZI formats differ, but they in general contain synchronization and deskewing data that allows the tape head circuitry to locate the data.

The data is recorded between the preamble and the postamble. GCR encoded data also contains resynchronization bursts.

The end of each block contains a postamble.

After the postamble comes the inter-block gap.

You didn't need to know that; it's transparent to the user. If you ever wondered where the "9" in "9-track" comes from, now you know.

18 and 36 track

Simply put, data is recorded across 18 or 36 tracks on tape frames with check bytes. It's not as simple as 9 track tape. Give us a call or check the ECMA standards below if you are interested in this.

Check the ECMA standard # 120 for a detailed analysis of 18 track tape formats. You may check the ECMA standard # 196 for a detailed analysis of 36 track tape formats.

DATA CHARACTER SETS - EBCDIC & ASCII

Back in the early days of ½ " computer tape, data was often stored as Binary Coded Decimal (BCD). This six bit data corresponded to the keypunch code of early IBM keypunch machines. Other computer manufactures used similar if not identical six bit representations.

IBM expanded the BCD code to include upper and lower case characters and some other control characters. IBM called this the Extended Binary Coded Decimal Interchange Code (EBCDIC).

In the Telegraph and Teletype industry, a seven-bit code called ASCII was beginning to replace a five-bit baudot code. Because ASCII could represent letters and upper and lowercase characters, it was also used in computer systems. Today, we know the first 128 characters as the ASCII character set. As an 8 bit representation the most significant bit is zero. When the most significant bit is one, we have what is sometimes called the extended ASCII code.

Data is stored on all magnetic tapes as magnetized areas recorded in one of two ways. Therefore binary values from 00000000 to 11111111 plus one 'parity bit' can be recorded across 9 tracks.

This data often represents ASCII or EBCDIC data. For instance: 001000001 = 'A' in ASCII. The decimal (base 10) value of 001000001 is 65. This is how data is represented on tape.

PHYSICAL TAPE BLOCKS and INTER-BLOCK-GAPS:

A collection of characters that the hardware writes or reads as a unit is called a physical block, or simply a block. Blocks are separated by sections of erased tape called Inter- Block-Gaps (IBGs), which are created automatically by the tape drive hardware. The length of the IBG for 1600 BPI 9-track tape is nominally 0.6 inches; its maximum legal length is 25 feet under ANSI standards. The design of the hardware assures that when the tape drive stops, the read/write head is always positioned in an inter-block-gap.

Why have inter-block gaps at all? The inter-block gap gives the computer time to process the blocks it just read so the tape drive can move at a continuous speed without stopping and starting. This is continuous movement is called 'streaming operation'. Of course, this assumes that the computer is fast enough to process the data during the time the drive is in the inter-block gap.

BLOCK LENGTH (also known as BLOCK SIZE):

The length of the data blocks on a tape is not standard and varies with the application. Block lengths from 1 byte up to 65535 bytes (64K less 1) may be read and written by most Electrovalue tape systems. Most minicomputers and mainframes are limited to a block size of 32,767 bytes. Whenever you write tapes for export to another computer, you should ask the intended recipient of the tape to specify the block size (or maximum block size) to be written. If there is no opportunity for such a discussion, write blocks close to 32,000 bytes, no larger, and make sure that that the logical records are not split across blocks. For example, if your logical record length is 100, a block size of 30,000 would be a good choice. Small block sizes are wasteful of tape and for this reason are contraindicated.

A simplified example of how block size helps to increase capacity is as follows:

Imagine a record size of 120 bytes and a block size of 120 bytes.

The data block when written at 1600 BPI takes up 120/1600 (0.075) of an inch of space on the tape. Each block is separated by an InterBlock Gap. In this case it is probably .6" In approximately six inches of tape you will have 10 times 120 = 1200 bytes of data.

Now, imagine a record size of 120 bytes and a block size of 1200 bytes.

The data block when written at 1600 BPI takes up 1200/1600 (0.75) of an inch of space on the tape. Each block is separated by an InterBlock Gap. In this case it is probably .6" In approximately six inches of tape you will have perhaps 5 times 1200 = 6000 bytes of data.

FIXED-LENGTH LOGICAL RECORDS, BLOCKING FACTOR

To facilitate processing of tape data, it is expected that when you write a tape you will group an integer number of fixed- length "LOGICAL" records into one PHYSICAL tape block. The ratio of physical block length to logical record length is called the BLOCKING FACTOR, i.e. **(record length) x (blocking factor) = block length.**

Filemarks

For convenience in processing and for organizational purposes, physical tape blocks are grouped into tape files. Tapes do not start out with a directory of files although some mainframe operating systems use extra files (called 'labels') to provide information about the data files on the tape. Sticky labels or accompanying paperwork are used to tell you what file(s) contain what data. It is customary to begin file numbering with 0, i.e. the first file on a tape is file zero.

The end of a tape file is indicated by a filemark (frequently also called a "TAPE-MARK" or an EOF MARK) which is a special record generated by and detectable by the hardware.

END-OF-FILE mark = TAPE-MARK = FILEMARK = EOF mark

It is an industry standard that two adjacent filemarks indicate the end of the valid data on a reel of tape. The Electrovalue software follows this convention.

DOUBLE EOF, DOUBLE TAPE-MARK, DOUBLE FILEMARK

Labeled tapes

Tapes from mainframes often contain extra small files, which are useful in the mainframe environment, but of little use to us. These labels are actually 80 byte long blocks of information containing such things as system codes, system names, volume numbers and user information. They usually contain the record length and block size for each file on the tape. Both the CUBS program and the KEYS program can read this information from the tape label.

Don't be fooled trying to import "label" files into your application. Labeled tapes are discussed, with examples, in the CUBS and the KEYS chapters of this book.

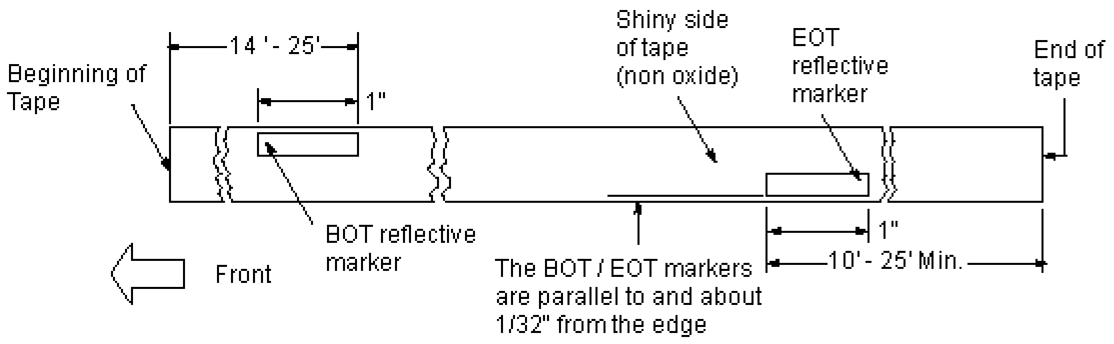
Reflective markers:

BEGINNING-OF-TAPE (LOAD POINT)

Near the beginning and end of each reel of tape are reflective markers that are optically sensed by the tape drive. All tape before the Beginning-Of-Tape (BOT) marker is considered leader. Data recording begins just beyond the BOT marker (this is the load point).

Replacing the BOT and EOT reflective markers

9 track tape tends to wear first at the beginning of the reel near the BOT marker. This can cause hard errors when you try to write on a tape. If you are getting hard errors near the beginning of the tape and you suspect tape wear is the cause, you can pull off 75’ to 125’ or so of tape and replace the BOT reflective marker. Place the new BOT marker between 14 and 25 feet from the physical beginning of the tape, parallel to and approximately 1/32” from the top of the tape.



Placement of BOT and EOT reflective markers

Caution: The above procedure will destroy the existing data on the magnetic tape. It should only be used when you are going to overwrite data on the tape.

END-OF-TAPE:

The End-Of-Tape reflective marker sets a status bit to warn the computer that the end of physical tape is near. The tape drive normally stops automatically when rewound or backspaced to the BOT marker. The tape drive does not automatically stop at EOT to allow reading or writing the last block(s). Electrovalue software handles EOT situations for you, with informative messages as necessary.

WRITE RINGS

Each reel of tape has a groove on the back into which a soft plastic ring may be inserted. This ring is called a "WRITE RING". If it is not installed in the reel of tape the drive will not write on the tape and the tape is said to be write protected.

THUMB WHEELS

Each cartridge has a thumb wheel on the back that turns. When the flat area of the wheel is showing, the cartridge can not be written to. The cartridge is said to be write protected. Often there is a picture of a lock on the flat surface of the wheel to emphasize this. The lock shows when the cartridge is write protected.

Re-writing Blocks within a file or files within a group of files:

Note that tape drive procedures (due to tolerance buildup) do not permit one to replace a block on tape with a new block and to expect that the following block or file to remain intact. For example, if you have ten files on tape you can not replace file 5 (even with a file of the same size) and expect file 6 to remain intact.

TAPE ERRORS

Tape read errors are rare if the tape is not old and the tape drive read/write head is kept clean. Most errors that are detected are due to a fleck of dust or dirt on the tape or dirty heads and are recoverable.

READ RETRIES

If after a read operation, the status indicates a read error, the Electrovalue software will backspace and attempt to re-read the block several times before posting any error conditions on the screen for operator action. Most tape drives contain a tape cleaning head which is designed to remove flecks of dust and dirt as the tape passes over it.

Tape capacity

The approximate capacity of various reels of tape is presented here for reference. The actual capacity of a reel of tape falls off quickly for small block sizes. Each block written to tape is followed by a 0.3 to 0.6 inch inter-block-gap (IBG) of blank tape. The figures below are for blocks of 3200 bytes. At a block size of 32K the capacity increases by approximately 13%.

Approximate 9 track tape capacity

REEL DIAMETER	FEET OF TAPE	CAPACITY 800 BPI	CAPACITY 1600 BPI	CAPACITY 3200 BPI	CAPACITY 6250 BPI
7"	600	5 MB	10 MB	20 MB	38 MB
8.5"	1200	10 MB	20 MB	40 MB	76 MB
10.5"	2400	20 MB	40 MB	80 MB	152 MB
10.5"	3600	30 MB	60 MB	120 MB	228 MB

Approximate cartridge capacity

Cartridge	Capacity uncompressed	With IDRC compression
3480	200 MB	800 MB
3490	650 MB	2.2 Gb
3490e	800 MB	2.4Gb

Network Notes

It is our recommendation that you install the tape drive on a client machine, not on a server. When the tape drive is running, it makes heavy use of the hard disk to transfer data. So do many servers in going about their serving business. One would slow the other down.

Installation on a client machine shouldn't be a disadvantage. Anytime any of our software asks for a disk file name the name may be preceded by an optional drive and path. The disk file may be anywhere on the network.

Note that this software will only run under Windows 95, Windows 98 and DOS. However, you may install Windows 95/98 on a client which runs the tape drive and have Windows 95/98, Windows NT, Windows 2000, Linux, or Novel as a server operating system. Your data files may be on the server or a client machine.

You can check Windows 95/98 documentation to see how to map server disk drives to disk drives on your machine.

Exporting Database Disk Data To Tape

WITH SPECIFIC EXAMPLES FOR dBASE III+

IMPORTANT NOTE: Because dBASE and its workalikes (FoxBASE, FoxPro, Clipper, Quicksilver, dBXL etc.) utilize the most popular internal database file format (.DBF files), Electrovalue's software provides the ability to export to tape directly from .DBF files. See the separate D2T chapter for full details.

The discussion below ignores using our D2T program to export tape data directly from .DBF files. The dBASE language is used to give examples. We hope these examples and the accompanying explanations will be helpful to users of non- dBASE databases. The actual commands used for any specific database language may, of course, be different.

If your database uses .DBF files read the D2T chapter to see how to directly export .dbf data. If you want all of the .DBF fields transferred to tape, the following discussion is moot. Using D2T saves time and space on your hard disk because you would not need to use one of the two-step procedures discussed below. D2T creates mainframe friendly tape files directly from dBASE compatible internal format .DBF files

Structure of data on tape

Typical mainframe tape data, as used for data interchange between computers, consists of fixed length data records, one after the other, end-to-end, without any extra record separator characters such as commas or quotes. Thus, your goal is to create a fixed field fixed record length database with no delimiters and transfer this database to tape.

Each record is built up from individual, fixed length fields. There are no field separators. There are no record separators either on magnetic tape. Because each record is fixed length and there are an integral number of records in each block, no record delimiters are needed. The software can figure out where each field and each record start and end.

In most PC type databases (not dbase) there exist record delimiters in addition to field delimiters. As an example an ASCII comma delimited database usually has double quotes surrounding each field, with commas as field separators and a carriage return and a line feed delimiting each record. Each field usually has leading and trailing spaces removed.

When transferring this ASCII comma delimited file to tape, you want to make each field a fixed length thus causing each record to be fixed length. You also want to remove the quotes and commas delimiting each field. Lastly, you will want to remove the carriage return and line feed from each record.

You can import ASCII comma delimited data into a matching dBASE file structure and output it as an SDF file. You could use FX to transfer the SDF file to tape truncating the carriage return & linefeed as you export the data.

The data on tape may be in the ASCII code, or more frequently, in EBCDIC (an IBM mainframe code). The tape itself doesn't contain the field names, lengths etc. It's your job to send that information, on paper, with the tape.

Internal Structure of data in a database .DBF disk file:

In addition to the data that you should send to the mainframe, a .DBF file contains a lot of extra information (for use only by dBASE and its workalikes) in the form of a variable length header and "flag" characters mixed with the data records. It would make no sense to transfer a complete .DBF file to tape and send it to computer shop using a mainframe computer.

Some programs such as Visual FoxPro create files ending in ".dbf" which are not truly compliant with dBASE III+ standards. Visual FoxPro native dbf files are 'enhanced' dbf files. They contain data types not found in dBASE III+ compatible files. Of course, Visual FoxPro will also open and work with standard .dbf files. To create files that are dBASE III+ compatible for use with D2T from Visual FoxPro .dbf files, use the Visual FoxPro command "copy to <filename> type FOX2X".

Creating Mainframe Friendly Data Tapes

Again, the discussion below ignores using our D2T program to export tape data directly from .DBF files. The dBASE language is used to give examples. We hope that these examples and the accompanying explanations will be helpful to users of non- dBASE databases. The actual commands used for any specific database language may, of course, be different.

If your database uses .DBF files you should read the D2T chapter to see how to directly export tape data. If you want all of the .DBF fields transferred to tape, the following discussion is obsolete. D2T creates mainframe friendly tape files directly from dBASE compatible internal format .DBF files.

STEP 1/ From dBASE's dot prompt, select the .DBF file of interest: e.g. USE DEMO <ENTER>

This is a good time to create a hardcopy of the structure of your database, you'll need the field definitions eventually:

```
e.g. LIST STRUCTURE TO PRINTER <ENTER>
An example of some of what you might see is:
  Field   Field Name      Type           Width
      1   FNAME           Character       15
      2   LNAME           Character       20
      3   CNAME           Character       32      PLEASE READ THE NOTE BELOW !
      4   STREET          Character       36
      5   CITY            Character       25      DON'T BE FOOLED BY 'Total'
      6   STATE           Character        2
      7   ZIP             Character        8      Note:  The "** Total **" displayed by dBASE is 1
      8   PHONE           Character       16      more than the sum of the number of characters in all
the
      9   DATE            Character        8      fields.  The record lengths we'll deal with are
either
     10   NOTES           Character       50      1 less than this number (i.e. the logical record
lengths
     11   SAW_AD_IN       Character       15      on tape) or 1 more than this number (i.e. the length
of
     12   CUST_NMBR       Character        6      a record plus the <CR/LF> in  "SDF" files.)
** Total **                               234
```

STEP 2/ Next, we use a dBASE command to create a disk file which is a generic, ASCII, "SDF" type file. An SDF file is almost in mainframe-friendly format. It consists of fixed length data records, built up from fixed length fields, with a CR/LF (the pair of ASCII characters carriage-return/line-feed) at the end of each record, e.g. from the dot prompt type:

```
COPY TO DEMO TYPE SDF <ENTER>
```

Record Length: The record length on disk will be the sum of the field lengths plus 2 characters for the CR/LF. If you're copying out all the fields, as in the above example, this is one character more than the TOTAL shown by 'LIST STRUCTURE'. See the note to the right of the sample structure on the first page.

CHARACTER FIELDS are right-filled (with ASCII SPACE characters) to the full size shown by the 'LIST STRUCTURE' command.

NUMERIC FIELDS are left-filled with ZEROes or SPACES. Numeric fields with decimal points are copied out including the decimal point character. For example, if you have a numeric field WIDTH = 10, DEC = 4, you'll get data of the following format: 12345.6789, 00123.4567, 00001.2345.

DATE FIELDS are written in the form YYYYMMDD. Thus a date of 11/22/88 (as shown by dBASE's 'EDIT' command) will be written to disk as 19881122. Mainframes aren't likely to want that format. If you must use date fields, and the SDF date format causes the mainframe grief, you can used Electrovalue's FXED program (after dBASE and before FX) in disk-to- disk mode to shuffle fields around, delete the "19" etc. to create the specific format you need.

Field Delimiters: The 'COPY TO...TYPE SDF' command doesn't create any; this is good, that's the way mainframes like the data. COPY TO does allow you to select just the fields you want, in the order you want - see your dBASE manual.

Filename Extension: Unless you specify a filename extension, the file named in the 'COPY TO' command will be given the extension .TXT. The file resulting from the above command will be called DEMO.TXT.

Control-Z Character: dBASE places a (mainframe offensive) CTRL-Z (IBM-PC End-of-File character) at the end of the disk file; we'll see how to get rid of the CTRL-Z with Electrovalue's FX program in a minute. You can read more about CTRL-Z in the FX chapter of the Electrovalue tape drive manual if interested.

Copy Options: Note that dBASE's 'COPY TO' command allows you to use the <scope>, FOR, WHILE and FIELDS options to export just those records and fields you choose.

STEP 3/ Use the FX program to write the SDF disk file to tape. Besides just transferring the disk data to tape, the necessary features FX will provide are:

Stripping off the CR/LF characters at the end of each record. ("Do you want to truncate...?" in the example below.)

Discarding the CTRL-Z character at the end of the disk file. ("Terminate...CTRL-Z ?" in the example below.)

Other features, which you'll have to pick properly (ask the intended recipient of the tape), are:

Writing the tape in EBCDIC or not.

What physical block size to use. FX will force this to be an integer multiple of the logical record size.

Writing the last tape block as a short block (recommended). To do this, use the /S option after the block size during the FX dialogue.

Sample Console Dialogue: Here's the pertinent part of the FX program's console dialogue dealing with our DEMO file:

DISK TO TAPE

```
Enter disk file name,
or names [e.g. ABC.TXT LILY FILE1.XYZ *.BAS XYZZY.?]: DEMO.TXT
Do you want to truncate or expand the records you're transferring to tape ? Y
Type input (disk) record length or 'V' for variable and hit <ENTER>: 235
Type output (tape) logical record length and hit <ENTER>: 233
Convert ASCII to EBCDIC? Y
Terminate transfer on first CTRL-Z in disk file? Y
Enter block length:  2330/S
Enter tape file # [0-9999] or <ENTER> for 'End-Of-Data': 0
Append disk file(s) to the END of an EXISTING tape file? N
```

Creating Delimited Data On Disk From Fixed Length Data On Tape

THE PURPOSE OF THIS DOCUMENT is to give a cookbook example of how to accomplish a common task. For example, it should be suitable for the boss to follow while the normal user of the tape drive subsystem is on vacation. Hopefully it will, in addition, provide some users with an introduction to the use of the FX program and the power of FXED control files. It is not meant to teach these programs - during the examples of console dialogue you'll find cameo appearances of features and syntax which aren't pertinent to this example and which are not explained here. See the FX and FXED chapters in the Tape Drive User Manual if you're curious to learn more.

OVERVIEW:

You have typical "fixed-length record" mainframe data on tape and want to transfer it to disk in "DELIMITED" format to import it into Paradox or to be processed by a program written in BASIC.

Tape-to-disk transfers are always handled by Electrovalue's FX (File eXchange) program.

Before you run the transfer with FX, you'll use the FXED program to generate a small disk file called a control file. The control file will contain the code conversion and data manipulation definitions that will be followed by FX during the actual tape-to-disk transfer. Control files can be used over and over and may be easily edited by FXED to create similar but new control files.

DEFINITION:

A DELIMITED DATA database consists of a collection of data records.

Each record has the same number of fields, and each ends in the following 2 characters: carriage return and line feed (shown as <CR> and <LF> in the following examples).

From one record to the next, corresponding fields may vary in length - and they usually will since one can save disk space by deleting trailing blanks (rightmost SPACE characters) from each fixed length tape field.

Fields are separated by commas <,>. Additionally, each field which is not a pure numeric field (i.e. it contains character data) is preceded with and followed by a double quote <"> character.

EXAMPLE:

A delimited record containing four fields [first name, age, middle initial, and last name] might look like the following:

"JEREMIAH",7,"X","BULLFROG"<CR><LF>

The following table contains the record layout information that came with the tape.

FIELD #	CONTENTS	START	END	LENGTH
1	LAST NAME	1	15	15
2	FIRST NAME	16	25	10
3	MIDDLE INITIAL	26	26	1
4	SOCIAL SECURITY NUMBER	27	35	9
5	SALARY (DDDDDDCC)	36	43	8

```
6            MALE (M) or FEMALE (F)            44      44    1
```

In the example which is unfolding, we're going to show how to import just some of these fields, and in a different order, into industry standard delimited data format.

User input is shown in **bold**.

From the DOS command line we invoke the FXED program. Then we answer 2 questions to pick a name for the control file and tell FXED the logical record length. When the first menu is displayed, we choose "T" to enter the Translation Definition Phase.

```
FXED <ENTER>
FX EDITOR  Rev. 96.31  02/08/94
Type the name of the FXED Control file: DELI <ENTER>   (Use any name you like)
Editing a new FXED Control file: DELI.CF
How many bytes are in each fixed-length input record? 44 <ENTER>

Editing FXED Control file: DELI.CF
Main Menu
  T - Define Types of Translation.
  O - Define Output Record Layout.
  C - Define Conditional Record Processing.
  E - Define Logical End of File (EOF).
  S - Define Secondary Translation Table.
  V - View all definitions.
  P - Print all definitions.
  D - Disk File Processing.
  G - Generate .DBF file.
  L - Change Input Record Length.
  F - File Maintenance.
  Q - Quit, Saving results on disk.
  A - Abort, Don't save results on disk.

Selecting "T"  selects the FXED Translation Definition phase.   You have to define some translation
type for every character position you're going to deal with.  We're assuming that the data on tape is
written in EBCDIC.  Note that your tape data might not be in EBCDIC, it might be in ASCII, or in a mix
of codes (e.g. EBCDIC and Packed Decimal).

FXED -     Translation Definition Commands
<range>E              Translate the bytes in <range> from EBCDIC to ASCII.
<range>e              EBCDIC to ASCII but force lower case ASCII output.
<range>CTRL-E         EBCDIC to ASCII but force UPPER case ASCII output.
<range>P              Translate the bytes in <range> from Packed decimal to ASCII.
<range>U              Undefines the bytes in <range>.
<range>Z              Translate the bytes in <range> from Zoned decimal to ASCII.
<range>=              Defines the bytes in <range> as needing no translation.      .
.etc

Here's where you specify that all 44 bytes of each input record are to get EBCDIC to ASCII translation:
1-44E
Now you hit <ESC> to return to the main FXED menu, and choose "O" to enter the Output Definition Phase.
<ESC>  .....     O
```

This calls up the output definition screen. You'll be inputting a mix of different types of commands. Tape field selections end in "\" to remove trailing blanks or with <ENTER> to bring in a fixed length field. Lines starting and ending with the <INS> key are literal text insertions - the literal characters between the <INS> keystrokes are mixed with the data from tape when the job is run. To input the literal characters <CR><LF>, hit the <ENTER> key while inserting (line 11 below).

```
FXED - Output Record Layout Commands
<range><ENTER>               Add the input bytes specified by <range> to the output.
<range>\              Same as <ENTER> except that trailing SPACEs are deleted.
<range>X              Same as \ but also removes leading SPACEs and 0's.
S                     Adds a SPACE to the output. (Shorthand to insert a SPACE).
<INS>text<INS>        Adds the text specified to output at current position.
xN                    Adds x digits of the input record number to the output.
```

```
xU                     Undefine (delete) output definition line x.
etc

 (01:001) <INS>"<INS>         ; Start with "
 (02:002) 16-25\              ; First Name
 (03:012) <INS>","<INS>       ; Add ","
 (04:015) 26<ENTER>           ; Middle Initial
 (05:016) <INS>","<INS>       ; Add ","
 (06:019) 1-15\               ; Last Name
 (07:034) <INS>",<INS>        ; Add ",
 (08:036) 27-35<ENTER>        ; SSN    (a numeric field doesn't get quotes)
 (09:045) <INS>,"<INS>        ; Add ,"
 (10:047) 44-44<ENTER>        ; Sex
 (11:048) <INS>"<ENTER><INS>  ; End with "<CR><LF>.
```

Now let's hit <V> to View the results so far:

V

For each line, FXED shows the line number (referenced when editing), the range of tape input bytes if any, the range of output bytes resulting from this line, the length of the output string, information on the format of the output string and a semicolon followed by an optional user-generated comment.

```
01:              O = 001-001  L = 001  '"'            ; Start with "
02: I = 016-025  O = 002-011  L = 010  TRUN FLD       ; First Name
03:              O = 012-014  L = 003  '","'          ; Add ","
04: I = 026-026  O = 015-015  L = 001  FIELD          ; Middle Initial
05:              O = 016-018  L = 003  '","'          ; Add ","
06: I = 001-015  O = 019-033  L = 015  TRUN FLD       ; Last Name
07:              O = 034-035  L = 002  '",'           ; Add ",
08: I = 027-035  O = 036-044  L = 009  FIELD          ; SSN
09:              O = 045-046  L = 002  ",""           ; Add ,"
10: I = 044-044  O = 047-047  L = 001  FIELD          ; Sex
11:              O = 048-050  L = 003  '"<CR><LF>'    ; End with "<CR><LF>.
```

now hit

<ESC>

to return to main menu. Then, most importantly, hit "**Q**" to save the control file to disk.

FXED returns control to DOS.

RUNNING THE TAPE-TO-DISK TRANSFER

From here, assuming that you have not made any mistakes, it's time to run the actual tape-to-disk transfer with FX.

FX<ENTER>

From the main menu, select **D** for Tape-to-Disk

FX asks what tape file contains the data (if you don't know, find out - see the MYSTERY TAPE chapter in the User Manual if you don't know how). The first file on tape is known as file 0. Try that if you like. If you get hardly any output or none at all, it's probably a labeled tape; re-run the job with file 1 as input.

Reset 8th bit ?   **N**

If input data processing criteria are defined in an FXED control file, input the <filename>, else press <ENTER> alone for no control file: DELI<ENTER>

Enter a ... disk filename ... **DELIDATA.TXT** (Use any name you like)

The job runs, you follow the prompts to exit FX back to DOS. The delimited data is on your disk, in this example it's in the current directory in a file called DELIDATA.TXT. You could, of course, have called it any filename you like and optionally preceded the filename with a drive and a path.

IMPORTANT: These few pages are intended to be a detailed "cookbook" example pertinent to a common

ASCII Chart

task. Complete user information on the FXED and FX programs is found in their chapters in this manual.

Character Name	Char	Code	Decimal	Binary	Hex
Null	NUL	Ctrl @	0	00000000	00
Start of Heading	SOH	Ctrl A	1	00000001	01
Start of Text	STX	Ctrl B	2	00000010	02
End of Text	ETX	Ctrl C	3	00000011	03
End of Transmit	EOT	Ctrl D	4	00000100	04
Enquiry	ENQ	Ctrl E	5	00000101	05
Acknowledge	ACK	Ctrl F	6	00000110	06
Bell	BEL	Ctrl G	7	00000111	07
Back Space	BS	Ctrl H	8	00001000	08
Horizontal Tab	TAB	Ctrl I	9	00001001	09
Line Feed	LF	Ctrl J	10	00001010	0A
Vertical Tab	VT	Ctrl K	11	00001011	0B
Form Feed	FF	Ctrl L	12	00001100	0C
Carriage Return	CR	Ctrl M	13	00001101	0D
Shift Out	SO	Ctrl N	14	00001110	0E
Shift In	SI	Ctrl O	15	00001111	0F
Data Line Escape	DLE	Ctrl P	16	00010000	10
Device Control 1	DC1	Ctrl Q	17	00010001	11
Device Control 2	DC2	Ctrl R	18	00010010	12
Device Control 3	DC3	Ctrl S	19	00010011	13
Device Control 4	DC4	Ctrl T	20	00010100	14
Negative Acknowledge	NAK	Ctrl U	21	00010101	15
Synchronous Idle	SYN	Ctrl V	22	00010110	16
End of Transmit Block	ETB	Ctrl W	23	00010111	17
Cancel	CAN	Ctrl X	24	00011000	18
End of Medium	EM	Ctrl Y	25	00011001	19
Substitute	SUB	Ctrl Z	26	00011010	1A
Escape	ESC	Ctrl [	27	00011011	1B
File Separator	FS	Ctrl \	28	00011100	1C
Group Separator	GS	Ctrl]	29	00011101	1D
Record Separator	RS	Ctrl ^	30	00011110	1E

Unit Separator	US	Ctrl _	31	00011111	1F
Space			32	00100000	20
Exclamation Point	!	Shift 1	33	00100001	21
Double Quote	"	Shift ‘	34	00100010	22
Pound / Number Sign	#	Shift 3	35	00100011	23
Dollar Sign	$	Shift 4	36	00100100	24
Percent Sign	%	Shift 5	37	00100101	25
Ampersand	&	Shift 7	38	00100110	26
Single Quote	‘	‘	39	00100111	27
Left Parenthesis	(	Shift 9	40	00101000	28
Right Parenthesis	)	Shift 0	41	00101001	29
Asterisk	*	Shift 8	42	00101010	2A
Plus Sign	+	Shift =	43	00101011	2B
Comma	,	,	44	00101100	2C
Hyphen / Minus Sign	-	-	45	00101101	2D
Period	.	.	46	00101110	2E
Forward Slash	/	/	47	00101111	2F
Zero Digit	0	0	48	00110000	30
One Digit	1	1	49	00110001	31
Two Digit	2	2	50	00110010	32
Three Digit	3	3	51	00110011	33
Four Digit	4	4	52	00110100	34
Five Digit	5	5	53	00110101	35
Six Digit	6	6	54	00110110	36
Seven Digit	7	7	55	00110111	37
Eight Digit	8	8	56	00111000	38
Nine Digit	9	9	57	00111001	39
Colon	:	Shift ;	58	00111010	3A
Semicolon	;	;	59	00111011	3B
Less-Than Sign	<	Shift ,	60	00111100	3C
Equals Sign	=	=	61	00111101	3D
Greater-Than Sign	>	Shift .	62	00111110	3E
Question Mark	?	Shift /	63	00111111	3F
At Sign	@	Shift 2	64	01000000	40
Capital A	A	Shift A	65	01000001	41
Capital B	B	Shift B	66	01000010	42
Capital C	C	Shift C	67	01000011	43
Capital D	D	Shift D	68	01000100	44
Capital E	E	Shift E	69	01000101	45
Capital F	F	Shift F	70	01000110	46
Capital G	G	Shift G	71	01000111	47
Capital H	H	Shift H	72	01001000	48
Capital I	I	Shift I	73	01001001	49
Capital J	J	Shift J	74	01001010	4A
Capital K	K	Shift K	75	01001011	4B
Capital L	L	Shift L	76	01001100	4C
Capital M	M	Shift M	77	01001101	4D

Capital N	N	Shift N	78	01001110	4E
Capital O	O	Shift O	79	01001111	4F
Capital P	P	Shift P	80	01010000	50
Capital Q	Q	Shift Q	81	01010001	51
Capital R	R	Shift R	82	01010010	52
Capital S	S	Shift S	83	01010011	53
Capital T	T	Shift T	84	01010100	54
Capital U	U	Shift U	85	01010101	55
Capital V	V	Shift V	86	01010110	56
Capital W	W	Shift W	87	01010111	57
Capital X	X	Shift X	88	01011000	58
Capital Y	Y	Shift Y	89	01011001	59
Capital Z	Z	Shift Z	90	01011010	5A
Left Bracket	[	[	91	01011011	5B
Backward Slash	\	\	92	01011100	5C
Right Bracket	]	]	93	01011101	5D
Caret	^	Shift 6	94	01011110	5E
Underscore	_	Shift -	95	01011111	5F
Back Quote	`	`	96	01100000	60
Lower-case A	a	A	97	01100001	61
Lower-case B	b	B	98	01100010	62
Lower-case C	c	C	99	01100011	63
Lower-case D	d	D	100	01100100	64
Lower-case E	e	E	101	01100101	65
Lower-case F	f	F	102	01100110	66
Lower-case G	g	G	103	01100111	67
Lower-case H	h	H	104	01101000	68
Lower-case I	I	I	105	01101001	69
Lower-case J	j	J	106	01101010	6A
Lower-case K	k	K	107	01101011	6B
Lower-case L	l	L	108	01101100	6C
Lower-case M	m	M	109	01101101	6D
Lower-case N	n	N	110	01101110	6E
Lower-case O	o	O	111	01101111	6F
Lower-case P	p	P	112	01110000	70
Lower-case Q	q	Q	113	01110001	71
Lower-case R	r	R	114	01110010	72
Lower-case S	s	S	115	01110011	73
Lower-case T	t	T	116	01110100	74
Lower-case U	u	U	117	01110101	75
Lower-case V	v	V	118	01110110	76
Lower-case W	w	W	119	01110111	77
Lower-case X	x	X	120	01111000	78
Lower-case Y	y	Y	121	01111001	79
Lower-case Z	z	Z	122	01111010	7A
Left Brace	{	Shift [	123	01111011	7B
Vertical Bar	|	Shift \	124	01111100	7C
Right Brace	}	Shift]	125	01111101	7D
Tilde	~	Shift `	126	01111110	7E

Delta	∆		127	01111111	7F