PROPOSED SMPTE RECOMMENDED PRACTICE

Serial Digital Interface Checkfield for 10-Bit 4:2:2 Component and 4f_{SC} Composite Digital Signals

RP 178
Revision of RP 178-1993

1 Scope

This practice specifies digital test signals suitable for evaluating the low-frequency response of equipment handling serial digital video signals as defined by ANSI/SMPTE 259M. These test signals are fully valid digital component video as defined in ANSI/SMPTE 125M. They are also useful digital composite video signals suitable for testing serial equipment in an out-of-service mode. Although a range of signals will produce the desired low-frequency effects, two specific signals are defined to test cable equalization and phase locked loop (PLL) lock-in, respectively.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this practice. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this practice are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

ANSI/SMPTE 125M-1995, Television — Component Video Signal 4:2:2 — Bit-Parallel Digital Interface

ANSI/SMPTE 244M-1995, Television — System M/NTSC Composite Video Signals — Bit-Parallel Digital Interface

ANSI/SMPTE 259M-1993, Television — 10-Bit 4:2:2 Component and 4f_{SC} NTSC Composite Digital Signals — Serial Digital Interface

3 General considerations

Stressing of the automatic equalizer is accomplished by using a signal with the maximum number of 1s or 0s with infrequent single clock periods with the other polarity. Stressing of the phase locked loop is accomplished by using a signal with a maximum low-frequency content; that is, with maximum time between transitions.

3.1 Channel coding of the serial digital signal defined by ANSI/SMPTE 259M utilizes scrambling and encoding into NRZI (nonreturn to zero inverted) accomplished by a concatenation of the following two functions:

$$G_1(X) = X^9 + X^4 + 1 \qquad G_2(X) = X + 1$$

As a result of the channel coding, long runs of 0s in the $G_2(X)$ output data can be obtained when the scrambler, $G_1(X)$, is in a certain state at the time when specific data words arrive. That certain state will be present on a regular basis; therefore, continuous application of the specific data words will regularly produce the low-frequency effects (see annex A).

3.2 Although the longest run of parallel data 0s will occur during the EAV/SAV for component signals and the TRS-ID for composite signals, the probability of coincidence with the required scrambler state to permit either stressing condition to occur is small. The amount of low-frequency effect is so time limited that equalizers and phase locked loops are not maximally stressed.

3.3 In the data portions of digital video signals (that is, all samples except the EAV, SAV, and ANC data flags), the sample values are restricted to exclude data levels 0 to 3 and 1020 to 1023 (000_h to 003_h and $3FC_h$ to $3FF_h$ in the hexadecimal representation). The result of this restriction is that the longest run of 0s, at the input to the scrambler is 16, occurring when a sample of value 200_h is followed by a sample of value between 004_h and 007_h. This situation can produce up to 26 0s at the encoder output which is also not a maximally stressed case.

3.4 Other specific data words in combination with specific scrambler states can produce a repetitive low-frequency serial output signal until the next EAV or TRS-ID is received to change the state of the scrambler. It is these combinations of data words that form the basis for the test signals defined by this practice.

3.5 Because of the Y/C interleaved nature of the component digital signal, it is possible to obtain nearly all combinations of sample word pairs by defining a particular flat color field in a noise-free environment. It is such sample word pairs which will produce the desired low-frequency effect. Because of the noninterleaved nature of the composite digital signal, such pairs will produce a square wave at one-half the clock frequency which is not a valid signal, but may be used for out-of-service testing.

4 Checkfield data

4.1 Receiver equalizer testing is accomplished by producing a serial digital signal with maximum dc content. Applying the sequence 300_h, 198_h continuously during the active line will produce a signal of 19 high (low) states followed by 1 low (high) state in a repetitive manner once the scrambler attains the required starting condition. Either polarity of the signal will be obtained depending on whether the 19 bits are high or low. By producing at least one-half field of such active lines, the required starting condition, and hence desired signal, will be produced on several lines.

4.2 Since an ITU-R BT.601-4 color black frame has an even number of 1s with no other data present, the polarity of the equalizer stress signal would be the same on each occurrence. The stressing sequence also has an even number of 1s in each word. Therefore, in component systems, it is necessary to add an odd number of 1s to each frame. For the equalizer test signal defined in this practice, the last active sample in line 20 for 525-line systems and the last sample in line 23 for 625-line systems should be 80_h instead of the 198_h which would otherwise be present. Designers are cautioned to ensure there is an odd number of 1s in a majority of frames if other data has been added to the signal in order to ensure both polarities of the test signal are produced.

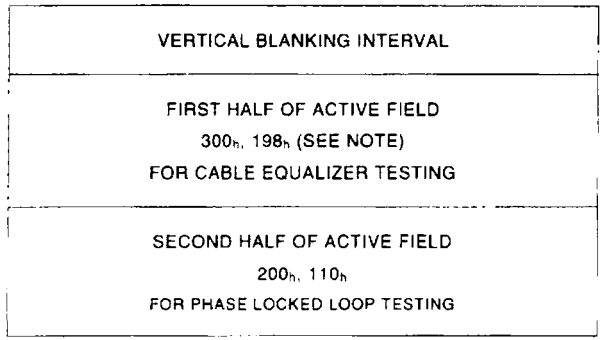
4.3 Receiver phase locked loop testing is accomplished by producing a serial digital signal with maximum low-frequency content and minimum number of zero crossings. Applying the sequence 200_h, 110_h continuously during the active line will produce a signal of 20 high (low) states followed by 20 low (high) states in a repetitive manner once the scrambler attains the required starting condition. By producing at least one-half field of such active lines, the required starting condition, hence desired signal, will be produced on several lines.

4.4 The signals defined in 4.1 and 4.2 are valid component digital signals as defined by ANSI/SMPTE 125M due to the interleaved nature of component digital signals. The same values (except for the single 80_h of 4.2) are used for composite digital testing of systems using the serial form of ANSI/SMPTE 244M; however, the signals are not valid as they represent a square wave at one-half the clock frequency. There are sequences which represent a small amplitude signal for composite digital which could be considered valid by virtue of their amplitude being more than 20 dB below video peak signal level. However, such signals have an average value more negative than blanking level which could disturb some systems under test. Therefore, for convenience of generation and ease of test operation, the composite digital test signals are chosen to be the same as the component digital test signals.

5 Serial digital interface (SDI) checkfield

5.1 Distribution of data in the SDI check field is shown in figure 1 for the signal standards referenced in clause 2. Specific distribution of sample values are shown in tables 1, 2, 3, and 4 for component 525/625 and composite 525/625, respectively. In each field, the line where the signal changes from one test signal to the other is defined as a range rather than a specific line.

NOTE – The alternate distribution of sample values (that is, interchanging the Y values with the C_b/C_r values) produces the same test signal result in the serial digital signal. Use of the alternate distribution is in compliance with this recommendation and may be preferred in some applications as picture disturbances are more visible in the resulting green display colors.



NOTE – For component signals, the last sample in the first active line of the first field is 80_h (see 4.2).

Figure 1 – Serial digital interface checkfield

Table 1 – 525-component SDI checkfield sample values

Line number	Word numbers	Sample	Word value
20, 283	0 ... 1436	C_b	300_h
	1 ... 1437	Y	198_h
	2 ... 1438	C_r	300_h
	4 ... 1439	Y	198_h
	⋮		
140 to 148 400 to 408 (approx mid-field)	0 ... 1436	C_b	200_h
	1 ... 1437	Y	110_h
	2 ... 1438	C_r	200_h
	4 ... 1439	Y	110_h
	⋮		
263, 525	0 ... 1436	C_b	200_h
	1 ... 1437	Y	110_h
	2 ... 1438	C_r	200_h
	4 ... 1439	Y	110_h

NOTE – Sample 1439 in line 20 is 80_h (see 4.2).

Table 2 – 625-component SDI checkfield sample values

Line number	Word numbers	Sample	Word value
23, 336	C_B0 ... C_B359	C_b	300_h
	Y 1 ... Y 718	Y	198_h
	C_R0 ... C_R359	C_r	300_h
	Y 2 ... Y 719	Y	198_h
	⋮		
160 to 168 470 to 478 (approx mid-field)	C_B0 ... C_B359	C_b	200_h
	Y 1 ... Y 718	Y	110_h
	C_R0 ... C_R359	C_r	200_h
	Y 2 ... Y 719	Y	110_h
	⋮		
310, 623	C_B0 ... C_B359	C_b	200_h
	Y 1 ... Y 718	Y	110_h
	C_R0 ... C_R359	C_r	200_h
	Y 2 ... Y 719	Y	110_h

NOTE – Sample Y 719 in line 23 is 80_h (see 4.2).

Table 3 – 525-composite SDI checkfield sample values

Line number	Word numbers	Word value
21, 284	0 ... 764	300_h
	1 ... 765	198_h
	2 ... 766	300_h
	4 ... 767	198_h
	⋮	
140 to 148 400 to 408 (approx mid-field)	0 ... 764	200_h
	1 ... 765	110_h
	2 ... 766	200_h
	4 ... 767	110_h
	⋮	
262, 525	0 ... 764	200_h
	1 ... 765	110_h
	2 ... 766	200_h
	4 ... 767	110_h

Table 4 – 625-composite SDI checkfield sample values

Line number	Word numbers	Word value
23, 336	0 ... 944 1 ... 945 2 ... 946 4 ... 947 ⋮	300_h 198_h 300_h 198_h
160 to 168 470 to 478 (approx mid-field)	0 ... 944 1 ... 945 2 ... 946 4 ... 947 ⋮	200_h 110_h 200_h 110_h
310, 623	0 ... 944 1 ... 945 2 ... 946 4 ... 947	200_h 110_h 200_h 110_h

Annex A (informative)
Bibliography

ITU-R BT.601-4, Encoding Parameters of Digital Television for Studios

Eguchi, Takeo. Pathological check codes for serial digital interface systems. SMPTE Journal 101(8): 553-558; August 1992.

PROPOSED SMPTE RECOMMENDED PRACTICE

Organization of DPX Files on TAR Tapes

1 Scope

1.1 This practice describes a method (TAR) by which DPX files are organized on tape storage devices (and any other type of sequential block-oriented device.)

1.2 This practice describes the tape file format basic concept, physical and logical layout, the directory file format, file naming conventions, and examples of usage.

1.3 This practice describes a recommended method for tape labels that are attached to the tape shell or reel.

1.4 This practice is intended for interchange only. It has not been optimized for other uses.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this practice. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this practice are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

ANSI X3.4-1986 (R1992), Information Processing — Coded Character Set — 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)

ANSI/IEEE 1003.1-1990, Portable Operating System for Computer Environments, Section 10, Data Interchange Format (TAR)

ANSI/SMPTE 268-1994, File Format for Digital Moving-Picture Exchange (DPX)

3 Tape file format

3.1 Basic concept

The tape file format is based on the TAR (tape archive) file format in common use in UNIX systems and available in the public domain for most operating systems. However, there are many ways to organize and store files on tape using TAR. This practice describes a method for organizing DPX files on tape that is optimized for a DPX user's typical needs. Specifically, it is optimized to support efficient pseudo-random access to any file contained on a tape, minimize the need for large hard-drive storage in the case when the entire contents of the tape are not needed, and be efficient with respect to tape usage and overhead.

TAR header blocks are used to create a header for each file on the tape. Additionally, file marks are used as delimiters between each file. The mechanics of creating TAR header blocks and file marks is usually hidden from the user by the TAR utility or other equivalent tape management software.

3.2 Physical layout

The physical layout is:

File 1	LFM	File 2	LFM	....

File N	LFM	(EOR) .. LEOT ... PEOT

LFM – Long file mark;
EOR – End of recording;
LEOT – Logical end of tape;
PEOT – Physical end of tape.

The long file mark is necessary for most tape-based devices to implement append or search/skip

operations and is used above to illustrate the general case. This physical layout is not required as long as the device can support search/skip operations. Tapes are generated using the TAR utility or by an application program capable of producing TAR compatible files. Each file on the tape is a TAR volume (i.e., wrapped with a tar header block at the beginning and a TAR EOF at the end [see table 1] except for use of the frame grouping factor [see 3.4.2.2]) and is followed by a file mark. The reason that there is only one file per tape archive record (TAR file) is because DPX files are usually very large and if a user does not have much hard drive space, it becomes difficult to easily extract the image or images desired. DPX files are one picture or frame of image per file so sequences of images do not result in large files. File marks are implemented differently by different device vendors, but they serve to provide improved efficiency for pseudo-random access to sequential devices. On UNIX machines, the "mt" command is used to write file marks and space over file marks, or an SCSI driver is used that sends the appropriate SCSI commands to create and use file marks.

NOTE – A long file mark (or some other equivalent device specific marking method) at the end of the TAR file is required to support appending files to a partially filled tape.

Table 1 – TAR volume

TAR header (512 bytes)
File Contents
TAR EOF (1024 bytes of zeroes)

It is recommended when reading or writing TAR files that the blocking factor be configured for an 8k block size. This is supported for most tape devices and provides adequate data transfer throughput rates. For example, if an application is using the UNIX TAR command to create image files, the following command (where 16 represents 512 * 16 = 8k) should be used:

TAR -cvf /dev/tape -b 16 filename

where:

TAR is the command;

- is a flag-to-follow indicator:

c is a flag meaning create a TAR file;

v is a flag meaning verbose reporting of relevant information;

f is a flag meaning the destination device or file is explicitly specified;

/dev/tape is an example of an explicitly specified destination device;

b is a flag meaning the blocking factor is explicitly specified;

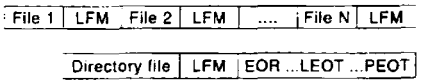
16 is an example of an explicitly specified blocking factor;

filename indicates the file to be TARed.

Blocking factors other than 16 are supported using the blocking factor comment field in the directory file (see 3.4.2.2 for details). The directory file should always be blocked at 16 so that most machines can at least read it.

3.3 Logical file layout

The physical layout is:



At the end of each tape, there will be a directory file, used to describe the contents of the tape.

3.3.1 Appending to a tape

The recommended method of appending files to a tape is to position the tape past the last TAR file (of the previous set of images) which would be the directory file, then back up to the beginning of the directory file, read and save the directory file, back up to the beginning of the directory file and start writing the tape. The new directory file that will appear after all the files must include all of the files on the tape as if the tape had been written all at the same time. This is why the above procedure read and saved the directory file. The "appending" process is transparent; it should not be possible to tell that the tape has been appended to if it is done correctly. When appending to a tape, it should be ensured that any affected comment lines are updated appropriately (like Total_Size).

3.4 Directory file format

The directory file takes the form:

TAR header:

Comment line(s)
File1name size date/time
File2name size date/time
.
.
.
FileNname size date/time
DPXDIR size date/time
(this is the directory file)

TAR EOF: The contents of the directory file are fully human-readable ASCII with the exception of the TAR header and TAR EOF (which get stripped off during the "deTARing" procedure).

3.4.1 File name entry format

Each file entry in the directory file has the following format:

Filename size date/time additional user info

where the fields are:

Filename – name of the film (alphanumeric);

Size – size of the file in bytes (numeric);

Date/time – the date and time the file was created in the same format that is used in ANSI/SMPTE 268M (DPX) (YYYY:MM:DD:HH:MM:SS:LTZ). This field is optional.

Additional user information – additional information the user finds valuable to be included in a directory.

For ease of reading the directory structure, each field within a file entry is separated by white space and the file entry is line feed terminated. The line feed character is hex 0A (decimal 10). Note to DOS and Mac users: DOS naturally terminates lines with a carriage return/line feed (CR/LF) pair (hex 0D followed by hex 0A). Max naturally terminates lines with a carriage return (CR – hex 0D). To be compatible with this practice, DOS and Mac systems must convert these character pairs or characters to line feeds. If for some reason it is not possible to properly terminate a line, make a note on the tape label as to what was done (see clause 5).

Nonprinting characters are not allowed except for the end-of-line terminator.

The following defines the maximum size for each field:

Field	Absolute maximum size (characters)	Recommended maximum size (characters)
Name:	100	46
Size:	10	10
Date/time:	23	23
User specific information:	Up to a total entry length of 255 characters	0

For ease of readability, it is recommended that no directory entry exceed 80 characters in length. In no case shall a directory entry exceed 255 characters.

3.4.2 Comment line format and conventions

Comment lines may be interspersed freely in the directory file. Usually, comment lines will appear at the beginning of the directory file before the first-listed file because they usually apply to all of the following files. Comment fields are entirely optional.

3.4.2.1 Comment line syntax

Comment lines always start with a '#' character in the first character position on a line. All following characters can be anything human-readable. Embedding nonprinting characters is not allowed. Comment fields are terminated the same way as all other directory file entries, with a line feed.

3.4.2.2 Comment line reserved words and meanings

The following words and word sequences are reserved and should not appear directly after the comment line character ('#') unless they are meant to be a reserved comment line word. Usually the lines using reserved comment line words should also be noted on the tape label as described in clause 5.

Project_Name ttttti ttttttti tt ttt ttttt tttt
Job_Name jjjj jjjjjjjj jjj j jjjjjj
Blocking_Factor xxxxx
Frame_Grouping_Factor xxx
Total_Size xxxxxxxxxxxx
Tape xxxxx of YYYYY

The keywords *Project_Name* followed by free ASCII text indicates the project name.

The keywords Job_Name followed by free ASCII text indicates a task or job within a project.

The keywords Blocking_Factor followed by a five-digit number indicates the blocking factor of the tape before the directory file. The directory file should always be blocked using a blocking factor of 16.

The keywords Frame_Grouping_Factor followed by a three-digit number indicates the number of DPX files in one TAR file. This mechanism has been provided to support large sequences of frames and is not recommended for generic interchange due to the possibility that a group of frames is larger than the user's local storage capacity. The frame grouping factor must be constant for all files (except for the directory file) following this designation. The last TAR file before the directory file need not contain the Frame_Grouping_Factor number of frames.

The keywords Total_Size followed by a 13-digit number indicates the total size of all files described by the directory file in bytes and including the directory file. This number is the actual recovered number of bytes, not the tape usage. The tape usage is typically significantly greater due to recoverable errors and tape marks.

The keyword phrase Tape xxxxx of YYYYY describes the sequence number of this tape in the total number of sequence-related tapes.

3.5 Naming convention

3.5.1 Directory file

The directory file name is DPXDIR in upper case characters as shown.

3.5.2 Image files

Each file name must be less than or equal to 100 characters in length. This is a constraint of TAR. Each image file name generated is of the form:

jobName/scene/elementName/resolution/
abbrevframeNumber.DPX

Example:

klingonEncounter/battle/moonOnWires/
3656x2664/mw000001.DPX

Subsets of the above path are allowed. That is, jobName, scene, elementName, and resolution can be left out of the file name. This chopping of the tree should be done as far down the directory tree as possible to make it easier for the receiver to integrate the files into his file system. This is because when extracting files with TAR, TAR wants to reconstruct the directory structure as stored on the tape onto the receiving system.

The first frame in a sequence is number 000001 unless some later edition forces negative or zero frame numbers. Resolution is the number of pixels per line, "x", number of lines per frame. Abbrev is up to two characters long. Frame number is six characters in length and is followed by a .dpx extension. No names should contain more than one "." character. This is so that a name parser can determine the file type if a type other than .dpx is contained on the tape and makes compatibility between differing operating systems easier to attain.

3.5.3 Negative sequence numbers

In the event that subsequent work creates frames which occurred before frame 1, zero or negative frame numbers are allowed. It is recommended that to accommodate negative frame numbers (indicated with an "-") that the abbrev field be limited to one character. It should be noted that a minimally compliant reader need recognize only positive sequence numbers and that this exceptional case be noted on the tape label. (see clause 5 for recommended methods for tape labeling).

4 Example

4.1 Creating a tape

This clause is a walk-through of the procedures required to create a tape compatible with the recommendations made in this practice. This example will be illustrated for UNIX, MS-DOS (DOS), and Mac operating systems. It is recommended that the tape be read before distributing it to make sure some fatal or silly error has not been made.

Assume we have three DPX files in the current directory (folder) and we want to make a directory file and a tape. Also assume that our files are at the bottom of the directory tree and do not require any higher levels of the directory tree. These procedures would need to be modified slightly to include multiple branches of a directory tree.

Our files are named credit1, credit2, and credit3.

4.1.1 UNIX example

Step 1: Rename the files to have a sequence number and to have an eight-character name using the "mv" command:

```
mv credit1 cd000001.dpx
mv credit2 cd000002.dpx
mv credit3 cd000003.dpx
```

(Instead of renaming, they could have been copied using the "cp" command.)

Step 2: View the current directory using the "ls -l" command. Now create a DPXDIR file using the "cat > DPXDIR" command and manually type in the directory information in the proper format referring to the output of the "ls -l" command previously executed. If you are familiar with a test editor, you need not go through the contortions listed below; just edit the file instead of cating it:

```
ls -l
```

(Screen will look as follows)

```
total 3
-rw-rw-r— 1 scotts  358 Sep  1 15:15 cd000001.dpx
-rw-rw-r— 1 scotts  358 Sep  1 15:15 cd000002.dpx
-rw-rw-r— 1 scotts  358 Sep  1 15:15 cd000003.dpx
```

```
cat > DPXDIR
cd000001.dpx    358     1994:09:01:15:15:00:PST
cd000002.dpx    358     1994:09:01:15:15:00:PST
cd000003.dpx    358     1994:09:01:15:15:00:PST
DPXDIR          100     1994:09:01:15:17:00:PST
^d (control-d)
```

Look in this newly created directory file (more DPXDIR). It should look like:

```
cd000001.dpx  358   1994:09:01:15:15:00:PST
cd000002.dpx  358   1994:09:01:15:15:00:PST
cd000003.dpx  358   1994:09:01:15:15:00:PST
DPXDIR        100   1994:09:01:15:17:00:PST
```

Now look in at the directory (ls -l). It should look like:

```
total 4
-rw-rw-r— 1 scotts  164 Sep  1 15:18 DPXDIR
-rw-rw-r— 1 scotts  358 Sep  1 15:15 cd000001.dpx
-rw-rw-r— 1 scotts  358 Sep  1 15:15 cd000002.dpx
-rw-rw-r— 1 scotts  358 Sep  1 15:15 cd000003.dpx
```

It was shown to replace the size of the DPXDIR file (0) with 100 knowing this was probably incorrect. Now that we know what the size is (164), we can correct the DPXDIR file:

```
sed 's/s100/164/' DPXDIR > temp
```

(The first character in the above quoted string is a 'one' character.)

Now view the new file we just created (more temp) to make sure it looks like DPXDIR except that the file size has been changed from 100 to 164. Once this is verified, overwrite the old DPXDIR with the new one (temp):

```
mv temp DPXDIR
```

You may wish to verify this step by doing one more "ls -l" (screen will look as follows):

```
total 4
-rw-rw-r— 1 scotts  164 Sep  1 15:20 DPXDIR
-rw-rw-r— 1 scotts  358 Sep  1 15:15 cd000001.dpx
-rw-rw-r— 1 scotts  358 Sep  1 15:15 cd000002.dpx
-rw-rw-r— 1 scotts  358 Sep  1 15:15 cd000003.dpx
```

Step 3: TAR up the files. Assume the tape drive is /dev/tape:

```
tar -cvf /dev/tape -b 16 cd000001.dpx
tar -cvf /dev/tape -b 16 cd000002.dpx
tar -cvf /dev/tape -b 16 cd000003.dpx
tar -cvf /dev/tape -b 16 DPXDIR
```

NOTES

1 The above procedure assumes two things: that the tape drive/TAR utility/tape drive driver does not automatically rewind after operations, and that the tape drive/TAR utility/tape drive driver writes a long file mark after the end of a tape write operation (so that appends are possible). It also assumes that all the files fit on the tape so the directory file was accurate and that this job was not appended onto another one.

2 If you had a lot of files, this method becomes tedious; writing a script file to automate this action is recommended. The reason such a script was not included here is that there are many UNIX date/time formats and it could not be guaranteed that the script would work properly on your machine.

4.1.2 MS-DOS example

Step 1: Rename the files to have a sequence number and to have an eight-character name using the rename command:

```
rename credit1 cd000001.dpx
rename credit2 cd000002.dpx
rename credit3 cd000003.dpx
```

(Instead of renaming, we could have copied them using the copy command.)

Step 2: View the current directory using the dir command. Now create a DPXDIR file by executing edit dpxdir and manually type in the directory information in the proper format referring to the output of the dir command previously executed.

```
dir
```

(Screen will look as follows:)

Volume in drive C is MS-DOS_6. Serial number is 1AE2:4C08. Directory of c:\trial*.*

```
.            <DIR>      1-14-94   3:54p
..           <DIR>      1-14-94   3:54p
cd000001.dpx       397  9-01-94   4:38p
cd000002.dpx       397  9-01-94   4:38p
cd000003.dpx       397  9-01-94   4:38p
    1,191 bytes in 5 file(s)      20,480 bytes allocated
                                  65,404,928 bytes free
```



```
edit dpxdir
cd000001.dpx   397   1994:09:01:16:38:00:PST
cd000002.dpx   397   1994:09:01:16:38:00:PST
cd000003.dpx   397   1994:09:01:16:38:00:PST
DPXDIR         100   1994:09:01:16:40:00:PST
```

Exit the editor (Alt-f x y).

Now look at the directory (dir). It should look like:

Volume in drive C is MS-DOS_6. Serial number is 1AE2:4C08. Directory of c:\trial*.*

```
.            <DIR>      1-14-94   3:54p
..           <DIR>      1-14-94   3:54p
cd000001.dpx       397  9-01-94   4:38p
cd000002.dpx       397  9-01-94   4:38p
cd000003.dpx       397  9-01-94   4:38p
dpxdir             194  9-01-94   4:42p
    1,385 bytes in 6 file(s)      32,768 bytes allocated
                                  65,396,736 bytes free
```

Now that we know what the size of DPXDIR is (194), we can correct the DPXDIR file by re-editing the file:

```
edit dpxdir
(change the size of the dpxdir file from 100 to 194).
```

Exit the editor (Alt-f x y).

You may wish to verify this step by doing one more dir (screen will look as follows):

Volume in drive C is MS-DOS_6. Serial number is 1AE2:4C08. Directory of c:\trial*.*

```
.            <DIR>      1-14-94   3:54p
..           <DIR>      1-14-94   3:54p
cd000001.dpx       397  9-01-94   4:38p
cd000002.dpx       397  9-01-94   4:38p
cd000003.dpx       397  9-01-94   4:38p
dpxdir             194  9-01-94   4:52p
    1,385 bytes in 6 file(s)      32,768 bytes allocated
                                  65,396,736 bytes free
```

Now, to adhere to the letter of this practice, you must use some utility to convert the DOS end-of-line characters (CR/LF) into a UNIX style end-of-line character (LF). Note that a conversion program which substitutes LF for CR/LF will change the size of this file. The file-size entry in the directory file must be corrected accordingly. For interchange of tapes between like operating systems, this requirement may be overlooked, but should be noted on the tape label (see clause 5).

Step 3: TAR up the files.

NOTE – MS-DOS does not come with the TAR utility. You must have installed this previously. You must also have installed any tape drivers required to support your tape drive and any application software required to write the files to the tape. The following procedure will not tell you how to get the TAR format files onto tape.

We will assume that you must first form a TAR format file then copy this file to your tape drive.

```
tar -cvf tarfile1 -b 16 cd000001.dpx
tar -cvf tarfile2 -b 16 cd000002.dpx
tar -cvf tarfile3 -b 16 cd000003.dpx
tar -cvf tarfile4 -b 16 dpxdir
```

This will have formed four TAR format files in your current directory. Now you must copy them to your tape drive using whatever application software came with your tape drive.

NOTE – The above procedure assumes two things: that the tape drive/TAR utility/tape drive driver does not automatically rewind after operations, and that the tape drive/TAR utility/tape drive driver writes a long file mark after the end of a tape write operation (so that appends are possible).

4.1.3 Mac example

Step 1: Rename the files to have a sequence number and to have an eight-character name by clicking on each file and typing the new name.

Step 2: Create a file directory. To do this, first select each file and get the file size and time/date information by selecting *File:Info* from the main menu bar and recording this information. Record the actual number of bytes used, not the number allocated on the hard drive. Then create the directory file by selecting the *TeachText* editor and the *File:New* menu choice from the menu bar. Now manually type in the information as shown below:

```
cd000001.dpx   1013   1994:11:15:09:01:00:PST
cd000002.dpx   1013   1994:11:15:09:01:00:PST
cd000003.dpx   1013   1994:11:15:09:01:00:PST
DPXDIR          100   1994:11:15:09:03:00:PST
```

Now select the *File:Save As* from the menu bar and save the file as DPXDIR.

Now close the editor and select the DPXDIR file you just made in the folder and get the file size and the time/date by again selecting the *File:Info* from the menu bar. Record this information. Close the *File:Info* box. Drag the DPXDIR file onto the *TeachText* icon to edit it. Edit in the correct file size and resave the file.

Now to adhere to the letter of this practice, you must use some utility to convert the Mac end-of-line character (CR) into a UNIX style end-of-line character (LF). For interchange of tapes between like operating systems, this requirement may be overlooked, but should be noted on the tape label (see clause 5).

Step 3: TAR up the files.

NOTE – Macs do not come with the TAR utility. You must have installed this previously. You must also have installed any tape drivers required to support your tape drive and any application software required to write the files to the tape. The following procedure will not tell you how to get the TAR format files onto tape.

We will assume that you must first form a TAR format file then copy this file to your tape drive.

```
tar -cvf tarfile1 -b 16 cd000001.dpx
tar -cvf tarfile2 -b 16 cd000002.dpx
tar -cvf tarfile3 -b 16 cd000003.dpx
tar -cvf tarfile4 -b 16 DPXDIR
```

This will have formed four TAR format files in your current directory. Now you must copy them to your tape drive using whatever application software came with your tape drive.

4.2 Reading a tape

4.2.1 UNIX example

4.2.1.1 Reading a file

To read an entire tape is easy (and even easier if you have a script file).

```
tar -xvf /dev/tape (to recover the file)
mt -f /dev/tape fsf (to skip forward to the next archive)
```

This command will restore the file from device /dev/tape into the current directory. In this case, it will have recovered the first file on the tape cd000001.dpx. If we want to recover all of them, we just repeat the above TAR and mt command for each file to be recovered. However, if you want to recover only

certain files in the middle of the tape, you will need to skip to it by counting how many files need to be skipped and then using the 'mt' command. For example, if we want to extract only cd000003.dpx, we need to skip two files. The command is:

```
mt -f /dev/tape fsf 2
tar -xvf /dev/tape
```

4.2.1.2 Getting the directory

There are many methods of getting to the directory file at the end of the tape. The easiest method is to space the tape forward to the end of the tape and then back up past the last file (which should be the directory file). The commands to cause this to happen are:

mt -f /dev/tape eom (drive the tape to the end)
mt -f /dev/tape nbsf 2 (back up past the last file and EOF mark and position at the beginning of the file)

Now the file can be deTARed normally. After the directory is deTARed, one can use the "mt -f /dev/tape rewind" command to return to the beginning of the tape.

The above method can take a long time and has been known to have compatibility problems between different tape drives and operating system versions.

Another method which is more tedious but faster and more reliable between platforms (and can be scripted easily) is to start at the beginning of the tape and, using the method in 4.2.1.1, move file by file through the tape until the DPXDIR file is found.

4.2.2 MS-DOS example

There is no MS-DOS example because the example is completely dependent on the TAR software in use.

NOTE – If a tape was made on a UNIX machine and conforms to this practice regarding end-of-line terminators, the chances are that DOS will have no problem displaying the directory file since most DOS applications do the right thing if they see a line feed without a carriage return (remember that DOS expects a carriage return and line-feed pair).

4.2.3 Max example

There is no Mac example because the example is completely dependent on the TAR software in use.

NOTE – If a tape was made on a UNIX machine and conforms to this practice regarding end-of-line terminators, the Mac will not be able to display the directory file without first converting the UNIX style end-of-line terminator (0A hex) into a Mac style end-of-line terminator (0D hex). This must be done by a file converter or some other utility. If this is not done, the file will probably display as one very long line with vertical jogs (line feeds).

5 Tape label

A DPX tape shall have a label attached to the outside of the tape reel or shell. This tape label should note any exceptional information.

A DPX tape will have as a minimum on the tape label the following:

1) DPX TAR format

2) "Tape x of y" where x and y are the tape and the total number of associated tapes to form a tape sequence number.

Helpful other information (which should be included in the directory file as a comment line) are things like blocking factor for the tape if it is different from the directory file, project name, and job name.

In the case that something on the tape does not conform to this practice (like end-of-line terminator), this should be noted on the tape label such as: "End-of-line is Mac style; sorry!"